ИНФОРМАТИКА

Вестник Сыктывкарского университета.
Серия 1: Математика. Механика. Информатика.
Выпуск 3 (32). 2019

УДК 004.056.55, 004.3

УСТРОЙСТВО ПОТОЧНОГО ШИФРОВАНИЯ НА ОСНОВЕ ПЛИС

Л. С. Носов, Е. Ю. Пипуныров

Предлагается использование софт-процессора, описанного на языке Verilog, и ПЛИС для создания устройства поточного шифрования, способного к программируемости и быстрой адаптации на аппаратном уровне.

Ключевые слова: защита информации, ПЛИС, шифрование.

1. Введение

Постоянное развитие технологий приводит к повсеместному применению электронных устройств и постоянному обмену информацией между ними. Утечка данной информации может послужить источником угроз приватности и даже общественной безопасности. Важность передаваемой информации стимулирует массовое применение средств защиты. Самым распространённым средством защиты передачи данных является криптографический метод. Разработка систем защиты

© Носов Л. С., Пипуныров Е. Ю., 2019.

для нестандартных решений «с нуля» может отнять много времени, сил, денег и других ресурсов. Помимо того, криптографические методы защиты информации (КМЗИ) хоть и повышают защищённость данных, однако требуют большого количества дополнительных вычислений, нагружающих аппаратуру. Стандартные микропроцессорные устройства имеют общий, неспециализированный набор команд, использование которых снижает скорость обработки данных.

В данной работе на основе практического примера рассматривается возможность создания универсального аппаратного средства (АС) защиты информации (ЗИ), которое можно в короткие сроки сконфигурировать для защиты данных, передаваемых по любому каналу. К разрабатываемому устройству предъявляются следующие требования:

- способность к интеграции с любыми цифровыми интерфейсами;
- возможность изменения структуры устройства на аппаратном уровне, адаптирования устройства к различным факторам;
- желательна возможность более быстрого изменения алгоритма работы устройства посредством его программирования.

В качестве удобного инструмента создания такого устройства предлагается механизм программируемых логических интегральных схем (ПЛИС). Дабы достичь максимальной гибкости конфигурирования устройства, его можно выполнить в виде ядра софт-процессора с набором только самых необходимых команд и специальным языком программирования. Тогда задачу адаптации устройства под новые условия можно будет решать одновременно на двух уровнях:

1) программном – если нужно будет изменить режим работы заданного алгоритма, то в большинстве случаев потребуется изменить небольшое количество строк кода;

2) аппаратном – если возникла необходимость использовать новый алгоритм, можно изменить тракт данных процессора, или изменится канал передачи – интерфейс к нему всегда можно скорректировать.

2. Архитектура и микроархитектура устройства

Итоговое устройство представляет собой софт-процессор со специфичным набором команд. В первую очередь необходимо определить архитектуру ядра устройства. В качестве основы для разрабатываемой архитектуры используется архитектура **MIPS** [1, 2]. Устройство было собрано на базе ПЛИС bf Xilinx с использованием комплекта **Xilinx Spartan-3AN FPGA Starter Kit Board** [3]. При этом сам софт-процессор описан с использованием языка **Verilog** [4].

От архитектуры **MIPS** были унаследованы следующие основные особенности:

- форматы инструкций;
- режимы адресации.

Помимо того, были унаследованы основные команды и дополнены командами, оптимизированными для реализации алгоритма **ГОСТ Р 34.12 — 2015** с размером блока 64 бита [5].

При составлении итогового набора инструкций основными целями являлись минималистичность, простота и максимальная применяемость. Всего в конечном наборе присутствует 26 инструкций: основные

арифметические, логические, условные инструкции, инструкции переходов и ввода-вывода, а также добавленные инструкции для оптимизации криптографического алгоритма.

На рис. 1 показана схема микроархитектуры разработанного устройства. В ней можно выделить 6 основных модулей:

1. Регистровый файл (**reg_file**).
2. Арифметико-логическое устройство (**ALU**).
3. Контроллер (**controller**).
4. Память инструкций (**instr_mem**).
5. Память данных (**data_mem**).
6. Счётчик команд (**PC**).

Счётчик команд и регистровый файл, содержащий 32 видимых программисту регистра, определяют архитектурное состояние. Зная текущее архитектурное состояние, процессор точно определяет, какую операцию и над какими данными надо выполнить для получения нового архитектурного состояния. Помимо того, в регистровый файл отображаются состояния периферийных устройств [2].

Арифметико-логическое устройство осуществляет выполнение всех логических и арифметических операций над входными данными. Память инструкций содержит загруженную программу, а память данных – полученные с интерфейсов и промежуточные данные. Итоговое устройство имеет гарвардскую архитектуру, а потому эти два вида памяти разделены друг от друга. Контроллер преобразует код операции и функции

в управляющие сигналы. Основной упор в процессе оптимизации был сделан на максимальное снижение взаимодействия с памятью на время выполнения алгоритма шифрования (табл. 1). Причиной этому будет то, что взаимодействие с памятью является затратным и ресурсоёмким процессом [1].

Таблица 1

Инструкции оптимизации криптоалгоритма

Инструкция	**Значение**
slc	инструкция циклического сдвига влево
src	инструкция циклического сдвига вправо
kxor	инструкция ИСКЛЮЧАЮЩЕГО ИЛИ с ключом
cs	инструкция подстановки из таблицы подстановок

В случае с алгоритмом «МАГМА» были оптимизированы 3 операции, входящие в него с помощью соответствующих модулей:

1. Выработка раундовых ключей (**key_gen**).
2. Операция подстановки (**subst**).
3. Сдвиговые операции (**shifter**).

Модуль генерации раундовых ключей хранит 256-битный ключ, реализованный в виде блока **ROM**. На вход модуля поступает номер раунда. Для вычисления номера подключа из номера раунда использовалось одно замеченное свойство: если начать нумерацию раундов не

с 1, а с 0, то номером ключей для первых 24 $(0-23)$ раундов является значение последних 3-х бит номера раунда. Для последних же 8-ми раундов $(24-31)$ для получения номера раундового ключа достаточно произвести инвертацию последних 3-х бит. Данная операция описана в

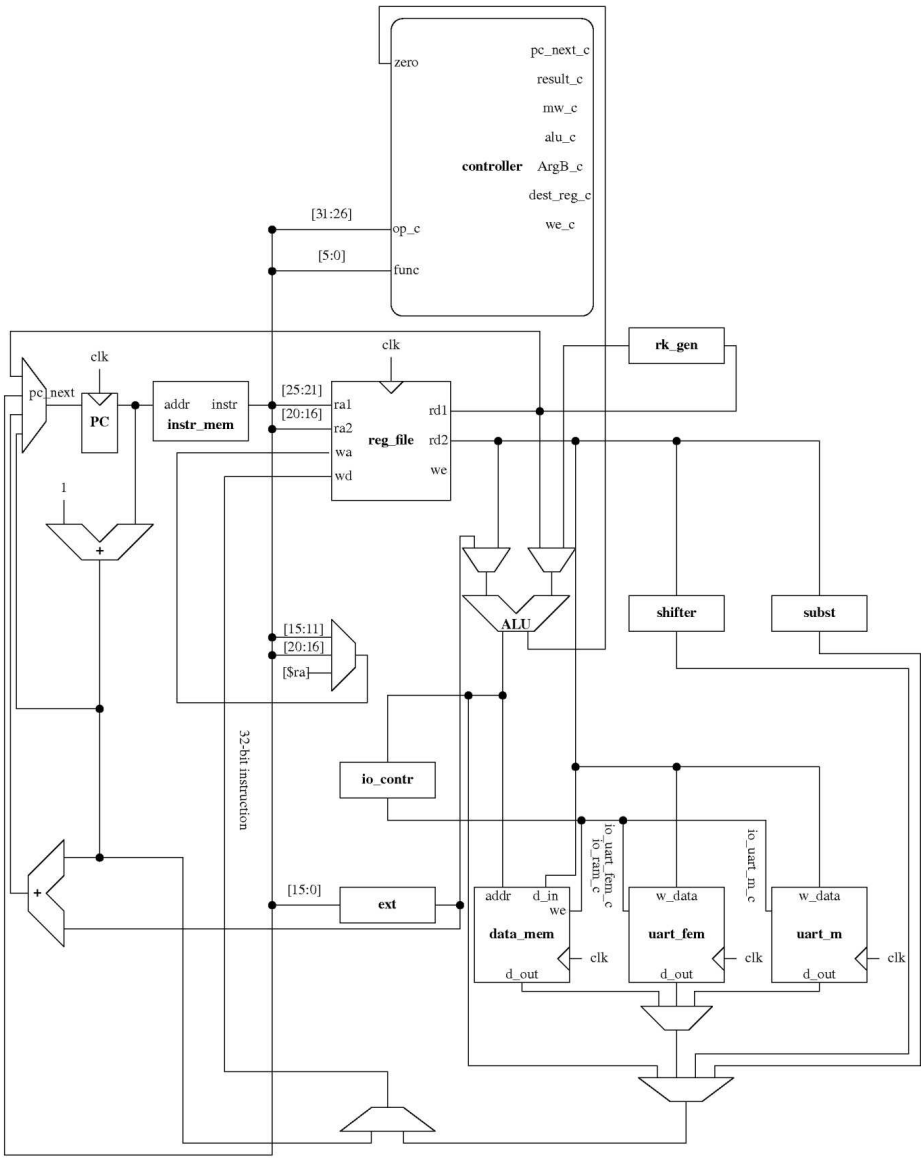

Рис. 1. Схема микроархитектуры устройства

следующей строке на языке **Verilog**:

```
assign keynum = (raund[4:3] == 2'b11)?raund[2:0]:~raund[2:0].
```

Выход данного модуля через мультиплексор подключён к первому входу **ALU**. На второй вход подаётся значение 32-битного блока. **ALU** производит операцию сложения данных значений по модулю 2. Таким образом, полностью реализовано сложение с раундовым ключом за 1 рабочий такт процессора.

Модуль, осуществляющий операцию подстановки на аппаратном уровне, реализован в виде 8-ми блоков **ROM**. На вход подаётся значение блока, для которого производится операция замены. Данное значение разбивается на 4-битные блоки, которые в дальнейшем используются в качестве адреса соответствующего им блока **ROM**. Полученные значения от каждого блока конкатенируются и подаются на выход [5].

Было расширено устройство сдвига посредством добавления в него возможности циклического сдвига. Данный вид сдвига не входит в базовую архитектуру **MIPS**-процессора. Таким образом, доля инструкций обмена данными с памятью во время выполнения основного алгоритма шифрования сведена к минимуму. Полная реализация одного раунда основного алгоритма «МАГМА» занимает всего 6 рабочих тактов процессора.

В итоговое устройство для тестирования была включена поддержка двух периферийных интерфейсов: **RS-232 DB9** и **DE9**. Передача данных поверх этого интерфейса происходит по протоколу **UART**. Конечное устройство производит шифрование и расшифрование данных, передаваемых по данным интерфейсам. Данные интерфейсы были вы-

браны по причине их изначального присутствия на тестовой плате и относительной простоты их реализации.

Оба интерфейса связаны с регистром **$us** (**uart state**) в памяти. Данный регистр содержит информацию о наличии принятых данных в регистре приёмника и заполнении регистра передатчика. В случае переполнения регистра приёмника происходит соответствующее прерывание. Оба интерфейса отображаются в память данных. К ним можно обратиться на чтение или запись с помощью команд чтения и записи в память по определенным адресам [1].

3. Программируемость устройства

Последней желаемой характеристикой устройства является его программируемость. В данной работе программируемость реализована несколько нестандартным методом. Причины тому описаны ниже.

Во-первых, созданное процессорное ядро имеет гарвардскую архитектуру, это означает, что память команд и память данных физически разделены друг от друга и память команд не изменяется в процессе работы устройства.

Во-вторых, устройство реализовано на основе ПЛИС, это значит, что его структуру можно изменить в любой момент.

В результате память команд в данном случае можно реализовать в виде блока **ROM**. Программируемость же означает реконфигурацию памяти команд в соответствии с неким программным кодом во время реконфигурации всей ПЛИС. Первым преимуществом данного подхода является простота его реализации. Вторым преимуществом является то, что при синтезе нетлиста происходит оптимизация тракта данных

синтезатором, который отбрасывает незадействованную часть логики. Так, например, при реализации шифратора в конечном нетлисте не будет присутствовать таймер, описание которого, однако, присутствует в составе проекта.

Для программирования устройства выбран язык ассемблера **MIPS**. Для преобразования языка ассемблера в коды инструкций был написан скрипт на языке **AWK**.

4. Тестирование работы устройства

Производилось шифрование данных, передаваемых по тестовому каналу передачи данных с интерфейсом **RS-232**. Для реализации потокового шифратора основной алгоритм использовался в режиме гаммирования с обратной связью по выходу [6]. Данный режим позволяет использовать идентичные устройства с одинаковым алгоритмом работы как для шифрования, так и для дешифрования, необходимо лишь синхронизировать их. В общем случае защищённый канал между двумя участниками представлен на рис. 2. Результаты тестирования (исходное и зашифрованное сообщение) представлены на рис. 3.

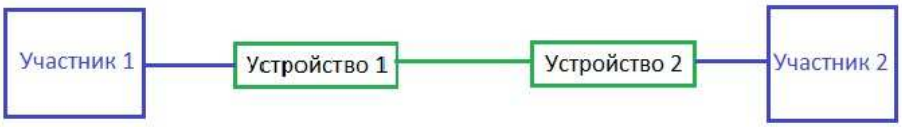

Рис. 2. Защищённый канал передачи

Общие характеристики конечного устройства приведены в табл. 2.

На основании вышеизложенных характеристик можно точно измерить возможные скорости шифрования устройства. Количество шиф-

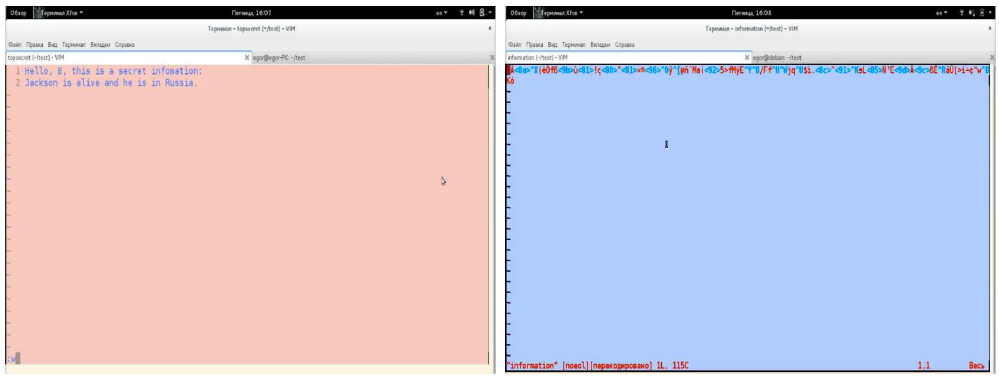

Рис. 3. Исходное сообщение (слева) и защифрованное сообщение (справа), передаваемые по защищенному каналу

руемых в секунду блоков $V_{\text{ш}}$ можно определить как

$$V_{\text{ш}} = \frac{f}{N_{\text{и}}} = \frac{25000000}{302} = 82781 \frac{\text{Блок}}{\text{с}},$$

Таблица 2

Характеристики полученного устройства

Характеристика	**Значение**
Тактовая частота	25 МГц (при реализации на Spartan3AN)
Тактов на инструкцию	1
Разрядность	32 бита
Алгоритм шифрования	ГОСТ 34.12 с длиной блока 64 бита «МАГМА»
Таймер	16-битный, конфигурируемый
Интерфейсы	LED x8, RS-232-DE9, RS232-DB9

где f – тактовая частота работы устройства при данной реализации, $N_{\text{и}}$ – количество инструкций для шифрования одного блока. Если учесть, что размер одного блока в битах $B \leq 64$, тогда максимальная скорость работы устройства:

$$V_{\text{ш}} \cdot B \leq 662251 \frac{\text{байт}}{\text{с}}.$$

Дальнейшие изменения скорости работы могут быть связаны со скоростью работы интерфейса. Устройство показало стабильную передачу зашифрованных данных по **RS-232** при передаче данных со скоростью 38400 бод. При дальнейшем увеличении скорости передачи появлялись помехи, которые «лавинно» искажали весь канал. Данные помехи связаны с несовершенством канала **RS-232**. Увеличение скорости передачи возможно при использовании порта **USB**.

5. Заключение

В работе на практике был показан метод для разработки универсального устройства поточного шифрования. Было получено устройство, представляющее собой софт-процессор с оптимизированными модулями для криптографических вычислений, описанный на языке **Verilog**. Если сравнивать полученное устройство с готовыми криптоконтроллерами, то представители данной группы обладают значительно большей производительностью по сравнению с софт-процессорами, так как являются готовыми оптимизированными аппаратными блоками, а софт-процессоры — по факту, лишь совокупностью запрограммированных ячеек. Так, например, контроллеры серии **STM32F4xx** компании **STMicroelectronics** способны работать на частоте до 168 МГц,

используя 32-битный тракт данных, что в результате приводит к скорости шифрования в разы большей, чем у получившегося устройства [7]. Однако софт-процессоры имеют гораздо большую гибкость по сравнению с готовыми аппаратными блоками. Так, например, при необходимости реализовать алгоритм «МАГМА» на базе контроллера из серии **STM32F4xx** пришлось бы реализовывать его на программном уровне, что заняло бы не меньше времени, чем описание оптимизированного модуля на **Verilog** или **VHDL** для софт-процессора, а работало бы, возможно, несколько хуже и с затратой несколько больших ресурсов. При этом намного большие проблемы возникли бы при необходимости реализовать шифрование данных, передаваемых через специфичный канал передачи.

Учитывая вышесказанное, можно сделать вывод о том, что представленный подход является отличным решением для защиты специфичных каналов передачи данных в случаях, не требующих большой скорости передачи.

Список литературы

1. **P. Pal Chaudhuri.** Computer organisation and design. Delhi: PHI Learning, 2014. 897 с.

2. **David M. Harris and Sarah L. Harris.** Digital Design and Computer Architecture. Boston: Morgan Kaufman, 2007. 570 с.

3. **ГОСТ Р 34.12-2015** Информационная технология. Криптографическая защита информации. Блочные шифры. М.: Стандартин-

форм, 2015. 25 с.

4. **IEEE 1364-2001** IEEE Standard Verilog Hardware Description Language. USA: The Institute of Electrical and Electronics Engineers, 2001 778 c.

5. **Pong P. Chu.** FPGA prototyping by Verilog examples Xilinx Spartan-3 Version. New Jersey:John Wiley & Sons, 2008. 488 c.

6. Spartan-3A/3AN FPGA Starter Kit Board User Guidei. v. 1.1. XILINX, 2008. 140 c.

7. **Самоделов А.** Криптография в отдельном блоке: криптографический сопроцессор семейства STM32F4xx. Официальный сайт компании «Компэл». URL: http://www.compel.ru/lib/ne/2012/6/4-kriptografiya-v-otdelnom-bloke-kriptograficheskiy-so-protsessor-semeystva-stm32f4xx. (дата обращения 03.12.2016).

Summary

Nosov L. S., Pipunyrov E. Y. Stream encryption based on FPGA

It is proposed to use a soft-processor, which described in Verilog language, and FPGA to create a universal stream cipher, which could be programmed and quickly adapted at the hardware level.

Keywords: information security, FPGA, stream cipher.

References

1. **P. Pal Chaudhuri.** *Computer organisation and design*, Delhi: PHI Learning, 2014, 897 p.

2. **David M. Harris and Sarah L. Harris.** *Digital Design and Computer Architecture*, Boston: Morgan Kaufman, 2007, 570 p.

3. **GOST R 34.12-2015** *Informatsionnaya tekhnologiya. Kriptograficheskaya zashchita informatsii. Blochnyye shifry* (Information technology. Cryptographic information security. Block ciphers), M.: Standartinform, 2015, 25 p.

4. **IEEE 1364-2001** IEEE Standard Verilog Hardware Description Language. USA: The Institute of Electrical and Electronics Engineers, 2001, 778 p.

5. **Pong P. Chu.** FPGA prototyping by Verilog examples Xilinx Spartan-3 Version. New Jersey:John Wiley & Sons, 2008, 488 p.

6. Spartan-3A/3AN FPGA Starter Kit Board User Guidei. v. 1.1. XILINX, 2008, 140 p.

7. **Samodelov A.** Kriptografiya v otdel'nom bloke: kriptograficheskiy soprotsessor semeystva STM32F4xx. Ofitsial'nyy sayt kompanii «Kompel» (Cryptography in a separate block: cryptographic coprocessor STM32F4xx family. The official website of the company «Kompel»), URL: http: //www.compel.ru/lib/ne/2012/6/4-kriptografiya-v-otdelnom-bloke-kriptograficheskiy-so-protsessor-semeystva-stm32f4xx. (date of the application: 03.12.2016).

Для цитирования: Носов Л. С., Пипуныров Е. Ю. Устройство поточного шифрования на основе ПЛИС // *Вестник Сыктывкарского*

университета. Сер. 1: Математика. Механика. Информатика. 2019. Вып. 3 (32). С. 62–76.

For citation: Nosov L. S., Pipunyrov E. Y. Stream encryption based on FPGA, *Bulletin of Syktyvkar University. Series 1: Mathematics. Mechanics. Informatics*, 2019, 3 (32), pp. 62–76.

СГУ им. Питирима Сорокина *Поступила 21.10.2019*