Реализация криптоалгоритма "Кузнечик" на ПЛИС

Ледерер П.А., Кочубей Д.С
Руководитель - к.п.н, доцент Кафедры интеллектуальных и многопроцессорных систем Механцев Б.Е.
Южный федеральный университет, г. Таганрог

Широкое применение криптографических систем в самых разных областях деятельности делает актуальной аппаратно-программные криптосистемы. Базовым требованием к таким системам является сочетание высокой скорости, не всегда достижимой при реализации на процессоре и комфорте в работе и отладки, трудно обеспечиваемой при чисто аппаратной реализации. Для разрешения этого противоречия в представленных результатах работы система была реализована комплексно. Система включает аппаратный блок, реализующий криптоалгоритмы и процессор для организации обмена данными. Оба блока, а также интерфейсы, небольшой объем оперативной памяти и сопутствующие интерфейсы реализованы на одной FPGA в составе отладочной платы Arty.

Преимущества аппаратной реализации. Основными преимуществами аппаратной реализации криптографических алгоритмов является высокая скорость шифрования по сравнению с программными аналогами. Так же подобные устройства уменьшают нагрузку на компьютер пользователя, так как ресурсоёмкие операции выполняются на специализированной плате расширения. Чаще всего подобные устройства применяются в компаниях и организациях занимающихся работой с большими объёмами конфиденциальной или секретной информации.

Криптоалгоритм кузнечик является блочным алгоритмом шифрования. Длинна блока данных составляет 128 бит, ключа шифрования - 256 бит. Работа алгоритма основана на сетях Фейстеля, благодаря этому "Кузнечика" удобно реализовывать аппаратно:

1. X-функция, побитовое “исключающее или” (далее XOR) блока данных и раундового ключа(генерацию раундового ключа будет разобрана позднее).
2. S-функция, нелинейное преобразование. Побайтовая замена одних значений на другие в соответствии с таблицей констант.
3. L-функция, линейное преобразование. Каждый байт блока умножается на коэффициент в поле Галуа, после чего производится суммирование всех байтов блока с записью результата в конец со сдвигом влево.

Эти шаги повторяются на протяжении 9-и раундов, на 10-ом выполняется только X-функция.

Генерация раундовых ключей. Процесс генерации раундовых так же основан на сети Фейстеля и во многом последовательность действий аналогична таковой в основном алгоритме, но есть ряд отличий.

Сначала мастер-ключ(256 бит) делится пополам, будем называть их правой и левой. Этапы создания раундового ключа:

1. X-функция. Происходит побитовый XOR. Левая часть ключа складывается с итерационными константами (заранее рассчитанными, приведём их расчёт позже).
2. S-функция. Выполняется побайтовая замена с левой частью мастер-ключа.
3. L-функция. Нелинейное преобразование, так же над левой частью.
4. Снова Xm-функция. Последний этап, XOR с правой половиной исходного ключа. Далее правая и левая часть меняются местами.

Далее преобразования X,S,L и Xm повторяются с правой частью 8 раз для получения новой пары ключей (8 раундов сети Фейстеля на каждую пару ключей). На первом раунде генерация раундового ключа не происходит, в качестве раундового ключа используется левая часть master-ключа. На втором раунде мы так же не генерируем ключ, взяв правую часть мастер-ключа, а вот для третьей уже потребуется выполнить 8 итераций сети Фейстеля.

Особенности реализации. Наша реализация алгоритма “Кузнечик” отличается от эталонной описанной в ГОСТ 34.12-2018. Было принято решение, для упрощения отладки и симуляции, уменьшить размеры блока данных со 128 до 16 бит и длину ключа с 256 до 32 бит. Так же было уменьшено количество раундов, у нас их 4 вместо 10. В остальном алгоритм остался неизменным.

Работа проводилась с отладочной платой Arty A7-35 имеющей ПЛИС(модель – XC7A35TICSG324-1L). На плате располагается jtag-программатор, 256 мб оперативной памяти, множество GPIO портов и портов расширения для подключения датчиков и других устройств, а так же поддерживает Microblaze.

Microblaze и ip-ярда. “Мозгом” системы является soft-процессор Microblaze. По сути своей это 32-ух битный процессор с RISC архитектурой. Он способен обеспечивать связь с периферийными устройствами средствами встроенных шин. На процессоре Microblaze можно запустить ряд операционных систем (Linux, FreeRTOS и тд). При этом программист способен выбирать какие функции процессора ему нужны и при необходимости удалять неиспользуемые для увеличения скорости работы системы.

Логика работы устройства при использовании Microblaze описывается на языках C/C++ в специальном SDK. Так же есть возможность подключения к Microblaze специализированных блоков (IP-блоков или IP-ядер), для выполнения определённых операций. Именно таким блоком будет наш криптоалгоритм. Окончательная архитектура системы представлена на Рисунке 1.

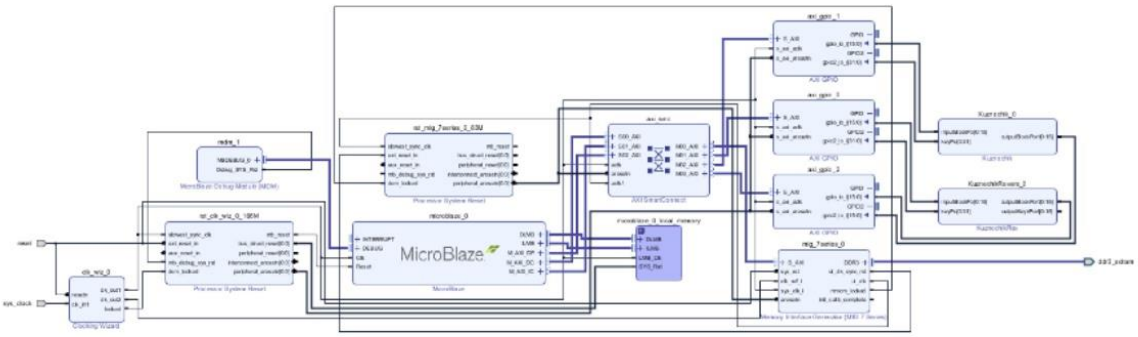

Рисунок 1 - Окончательная архитектура системы

Симуляция работы алгоритма. Реализация криптосистемы на FPGA осуществлялась на языке VHDL. Использованы IP ядра для организации процессора Microblaze и типовых интерфейсных блоков. Проектирование и симуляция системы проводилась в IDE Vivado. Результат симмуляционных тестов представлены на рисунке 2.

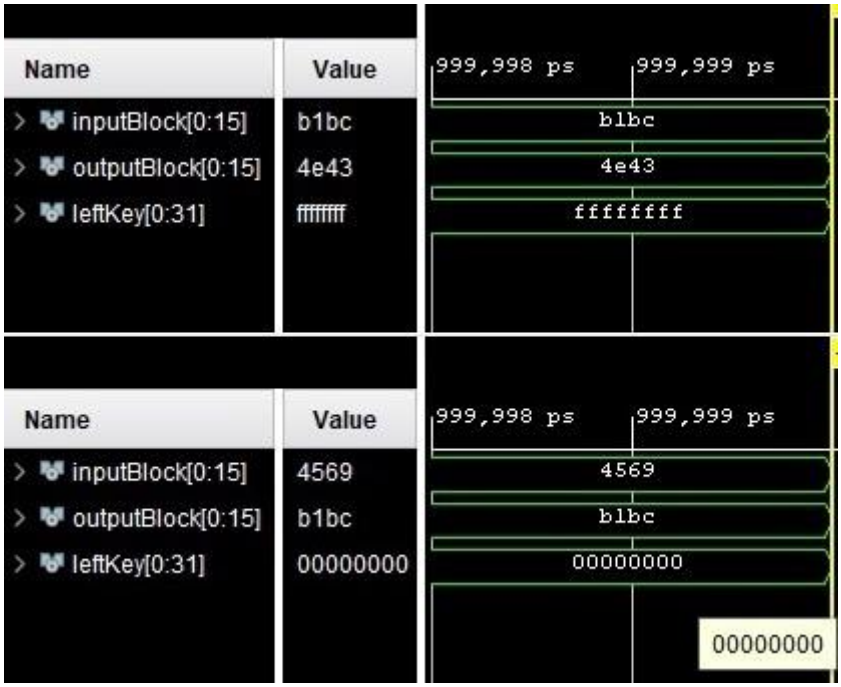

Рисунок 2 - Результаты тестов

С исходным кодом проекта можно ознакомиться по ссылке: https://github.com/FriZIk/FPGA.

Дальнейшее развитие. Одним из путей дальнейшего улучшения является использование конвейерных вычислений для распараллеливания однотипных задач. Так же в будущем возможно взять ПЛИС семейства Zynq, в них аппаратно присутствует ядро ARM Cortex A9, на которое возможно переложить функции, сейчас возложенные на Microblaze, тем самым дополнительно увеличив скорость обработки данных.

Список литературы.

1. Криптографический алгоритм «Кузнечик»: просто о сложном // Habr URL: https://habr.com/ru/post/45900
2. Разработка процессорной системы на базе софт-процессора MicroBlaze в среде Xilinx Vivado IDE // fpga-systems URL: https://fpgasystems.ru/publ/xilinx/microblaze/razrabotka_processornoj_sistemy_na_baze_soft_processora_microblaze_v_srede_xilinx_vivado_ide_hlx_chast_2/10-1-0-7
3. Creating a custom IP block in Vivado // fpga-developer URL: http://www.fpgadeveloper.com/2014/08/creating-a-custom-ip-block-invivado.html
4. П. Н. Бибило Основы языка VHDL. Либроком, 2016.
5. И.И. Левин, Б.Е. Механцев Лабораторные работы по дисциплине "ПЛИС-технологии и методы создания эффективных прикладных программ для ПЛИС". Таганрог: Издательство Южного федерального университета, 2017.
6. Xilinx [Электронный ресурс]. – URL:https://www.xilinx.com/products/silicon-devices.fpga.html (дата обращения: 31.04.2020).