Федеральное государственное автономное образовательное учреждение высшего образования

«Московский физико-технический институт (национальный исследовательский университет)»

Защита информации

Эссе на тему:

Реализация алгоритмов создания и проверки цифровой подписи на ПЛИС

Работу выполнил:
Румянцев Дмитрий
группа Б01-110а

г. Долгопрудный
2024 год

1 Введение

В современном цифровом мире обеспечение безопасности данных является приоритетной задачей для множества организаций. Одним из ключевых инструментов защиты информации является цифровая подпись — криптографический метод, который позволяет удостоверять подлинность и целостность данных. В последние годы программируемые логические интегральные схемы (ПЛИС) становятся популярным выбором для реализации различных криптографических алгоритмов. ПЛИС, благодаря своей гибкости и возможности параллельной обработки данных, предлагают ряд преимуществ для выполнения вычислительно сложных операций.

В данном эссе будет рассмотрен процесс реализации алгоритмов создания и проверки цифровой подписи на ПЛИС, выделены основные технические аспекты и преимущества данного подхода.

2 Цифровая подпись и её значимость

Цифровая подпись — это криптографический механизм, который используется для подтверждения подлинности и целостности данных, а также для обеспечения неотрекаемости, то есть гарантии того, что отправитель не сможет отказаться от факта отправки информации. Цифровая подпись играет важную роль в электронной коммерции, юридических документах, программных лицензиях, а также в других сферах, где требуется подтверждение подлинности передаваемой информации.

В данной работе используется стандарт ГОСТ 34.10-2018, который описывает алгоритм электронной цифровой подписи (ЭЦП) на основе эллиптических кривых, обеспечивающий высокую криптостойкость и безопасность. Данный алгоритм широко используется в России для создания и проверки цифровых подписей. Ниже приведены основные этапы создания и проверки подписи.

2.1 Алгоритм создания электронной подписи

Входные данные:

1. Сообщение M — данные, которые нужно подписать
2. Закрытый ключ d, принадлежащий подписанту
3. Параметры эллиптической кривой, такие как P - базовая точка подгруппы, q - порядок циклической подгруппы группы точек эллиптической кривой и т.д.

Алгоритм:

1. Вычислить хэш-код сообщения :
$$\overline{h} = h(M),$$
где h - хэш-функция (ГОСТ Р 34.11-2012).
2. Вычислить целое число α, двоичным представлением которого является вектор $\overline{h}$, и определить:
$$e \equiv \alpha \mod q$$
Если $e = 0$, определить $e = 1$.
3. Сгенерировать случайное (псевдослучайное) число k, удовлетворяющее неравенству:
$$0 < k < q$$

4. Вычислить точку на эллиптической кривой $C = k \cdot P = (x_C, y_C)$ и определить:

$$r \equiv x_C \mod q$$

Если $r = 0$, возвращаемся к шагу 3.

5. Вычислить значение:

$$s \equiv (r \cdot d + k \cdot e) \mod q$$

Если $s = 0$, возвращаемся к шагу 3.

6. Вычислить двоичные векторы $\overline{r}$ и $\overline{s}$, соответствующие r и s, и определить цифровую подпись $\zeta = (\overline{r}||\overline{s})$ как конкатенацию двух двоичных векторов.

Результатом является цифровая подпись ζ.

2.2 Алгоритм проверки электронной подписи

Входные данные:

1. Сообщение M
2. Подпись ζ
3. Открытый ключ Q
4. Параметры эллиптической кривой (такие же, как при создании подписи)

Алгоритм:

1. По полученной подписи ζ вычислить целые числа r и s. Если выполнены неравенства $0 < r < q$, $0 < s < q$, то перейти к следующему шагу. В противном случае подпись неверна.
2. Вычислить хэш-код полученного сообщения M:

$$\overline{h} = h(M),$$

где h - хэш-функция (ГОСТ Р 34.11-2012).

3. Вычислить целое число α, двоичным представлением которого является вектор $\overline{h}$, и определить:

$$e \equiv \alpha \mod q$$

Если $e = 0$, определить $e = 1$.

4. Вычислить значение:

$$v \equiv e^{-1} \mod q$$

5. Вычислить значения:

$$z_1 \equiv (s \cdot v) \mod q,$$
$$z_2 \equiv (-r \cdot v) \mod q$$

6. Вычислить точку на эллиптической кривой $C = z_1 \cdot P + z_2 \cdot Q = (x_C, y_C)$ и определить:

$$R = x_C \mod q$$

7. Если выполнено равенство $R = r$, то подпись считается действительной. В противном случае — недействительной.

Результатом является свидетельство о достоверности или ошибочности подписи.

3 Программируемые логические интегральные схемы как средство реализации алгоритмов

Программируемые логические интегральные схемы (ПЛИС) представляют собой универсальные электронные устройства, которые могут быть запрограммированы для выполнения специфических логических операций. В отличие от ASIC (специализированных интегральных схем), которые разрабатываются для выполнения конкретных функций, ПЛИС могут быть перенастроены для выполнения различных задач, что делает их более гибкими и подходящими для применения в ряде областей, включая криптографию. Архитектура ПЛИС состоит из большого числа логических блоков, соединённых программируемыми соединениями. Это позволяет параллельно обрабатывать большие объемы данных, что делает ПЛИС идеальными для реализации вычислительно сложных алгоритмов, таких как цифровая подпись. Одним из ключевых преимуществ ПЛИС является возможность оптимизировать алгоритмы под конкретные задачи, что увеличивает производительность и снижает задержки при обработке данных.

В криптографических приложениях, таких как создание и проверка цифровых подписей, ПЛИС позволяют значительно ускорить выполнение операций за счет их параллельной архитектуры. Это особенно важно в системах реального времени, где требуется быстрая обработка данных и высокая надёжность.

4 Реализация алгоритмов создания и проверки цифровой подписи на ПЛИС

4.1 Архитектура

Одним из ключевых этапов разработки цифровых устройств является создание их архитектуры. Оптимально спроектированная архитектура упрощает процесс верификации логики устройства, повышает его производительность, а также снижает энергопотребление и тепловыделение.

Рассмотрим разделение устройства на логические блоки. Ниже представлен список блоков с их кратким описанием:

- *dsig_top* - блок верхнего уровня, объединяющий более маленькие блоки
- *dsig_gen* - блок, создающий электронную подпись или вычисляющий публичный ключ
- *dsig_ver* - блок, проверяющий электронную подпись
- *data_ctrl* - блок, осуществляющий обмен данными между внешним миром и устройством

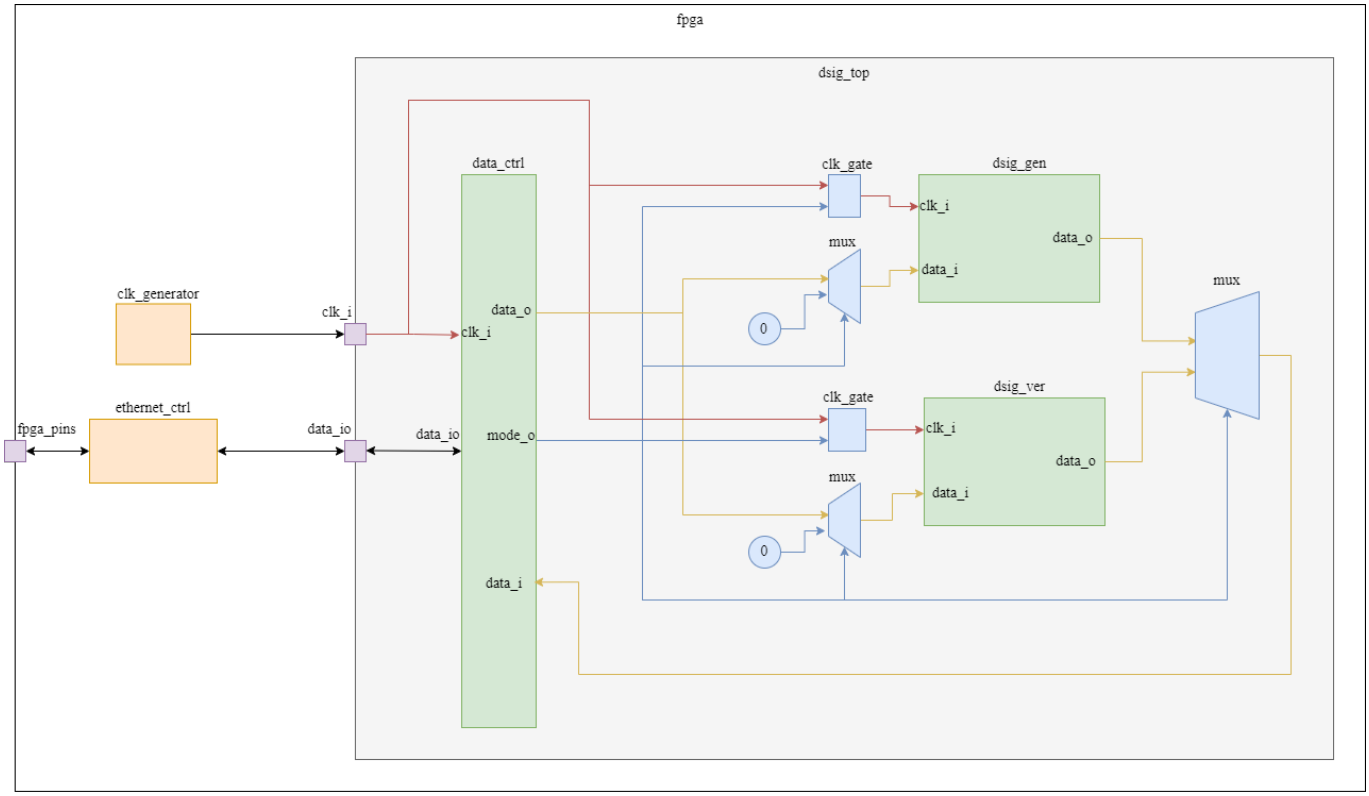

Рис. 1: Верхнеуровневое разбиение на логические блоки

На рис. 1 представлено верхнеуровневое разбиение устройства на логические блоки, а также логика для их взаимодействия. На *dsig_top* приходит тактовый сигнал с генератора (в нашем случае частота этого сигнала 50 МГц), а также двунаправленная шина данных, соединяющая вычислительный блок с Ethernet-модулем. Тактовый сигнал идет на все блоки, но перед блоками *dsig_gen* и *dsig_ver* стоят ячейки (*clk_gate*), блокирующие его распространение для неактивного блока (например, если в определенный момент поступил запрос на создание электронной подписи, и блок *dsig_ver* находится в неактивном состояние, его тактовый сигнал будет заблокирован (подсоединен к константе 0)). Данные с внешнего мира поступают в виде Ethernet-фреймов на Ethernet-контроллер (*ethernet_ctrl*), где они преобразуются в необходимое представление (удаляется ненужная информация, связанная с передачей данных по Ethernet). Далее эти данные поступают на *data_ctrl* модуль, где они разделяются на две шины: режим работы и даннные, необходимые для работы вычислительных модулей (сообщение (подписанное или неподписанное), идентификатор пользователя). Потом эта информация подается на вычислительные блоки, проходя через мультиплексоры, которые блокируют ее (подают 0) для неактивного модуля. Результат вычислений подается на *data_ctrl* и далее на *ethernet_ctrl*, в котором преобразуется в Ethernet-фрейм и отправляется пользователю.

Теперь рассмотрим строение вычислительных блоков *dsig_gen* и *dsig_ver*. Основные элементы *dsig_gen* (рис. 2):

- *hash_streebog* - блок, считающий хэш от сообщения с помощью функции Стрибога
- *rand_num_gen* - блок, генерирующий псевдослучайные числа
- *key_storage* - блок, хранящий секретные ключи пользователей
- *elliptic_curves_gen* - блок, создающий электронную подпись

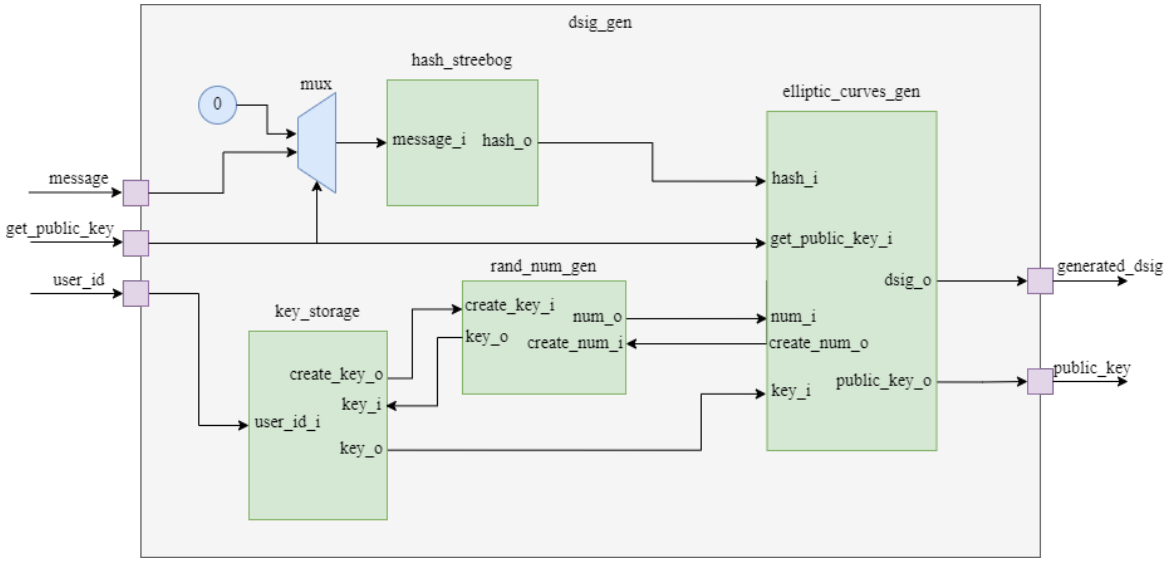

Рис. 2: Строение блока *dsig_gen*

У блока *dsig_gen* есть два режима: создание подписи и вычисление публичного ключа пользователя, режимы управляются специальным сигналом *get_public_key*, если он равен 1, то включается режим нахождения публичного ключа, если равен 0 - режим создания подписи. На вход *hash_streebog* поступает сообщение (если включен режим создания подписи), иначе подается 0, на выходе получается хэш этого сообщения (или 0). Блок *rand_num_gen* генерирует псевдослучайные числа на основе заданных параметров. В качестве алгоритма для генерации псевдослучайных чисел используется регистр сдвига с линейной обратной связью, или LFSR. На вход *key_storage* поступает идентификатор пользователя, на выходе получаем его секретный ключ. Если пользователя нет в базе, то *key_storage* делает запрос *rand_num_gen* для генерации ключа, *rand_num_gen* генерирует ключ и отправляет его обратно на *key_storage*, который в свою очередь заносит пользователя и сгенерированный ключ в базу. На вход *elliptic_curves_gen* поступают хэш сообщения (или 0), секретный ключ и псевдослучайное число (если режим создания подписи), на выходе получается электронная подпись (если режим создания подписи) или публичный ключ пользователя (если режим нахождения публичного ключа).

Основные элементы *dsig_ver* (рис. 3):

- *primitive_ver* - блок, проверяющий принадлежность двух частей подписи (r и s) нужным промежуткам
- *hash_streebog* - блок, считающий хэш от сообщения с помощью функции Стрибога
- *elliptic_curves_ver* - блок, проверяющий электронную подпись

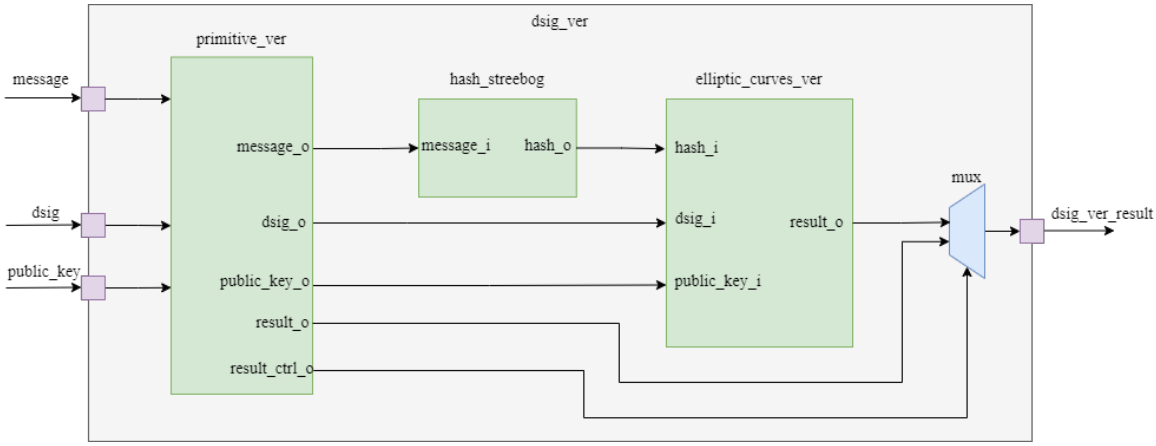

Рис. 3: Строение блока *dsig_ver*

На вход *primitive_ver* поступают сообщение, подпись и публичный ключ. Этот блок делает примитивную проверку подписи (проверяет, находятся ли r и s в нужных промежутках, если нет - отправляет на выход результат о некорректности подписи, иначе отправляет входные данные на остальные блоки). На вход *hash_streebog* поступает сообщение, на выходе получается хэш этого сообщения. На вход *elliptic_curves_gen* поступают хэш сообщения, электронная подпись и открытый ключ, на выходе получается результат проверки подписи.

Стоит отметить, что представленная выше архитектура может меняться в процессе ее реализации.

На рис. 4 приведена логика работы устройства.

Для взаимодействия с пользователем планируется создать web-site. С помощью графического интерфейса пользователь сможет авторизоваться, выбрать режим работы и загрузить сообщение. Далее эти данные будут отправлены на ПЛИС по Ethernet. Также на этот сайт будут приходить данные с ПЛИС.

Проверку корректности данных планируется реализовать только на стороне сайта, т.е. на ПЛИС будут отправляться корректные данные. Это упростит и ускорит обработку данных при получении.

Таким образом, нам не требуется производить обработку ошибок и исключений на ПЛИС.

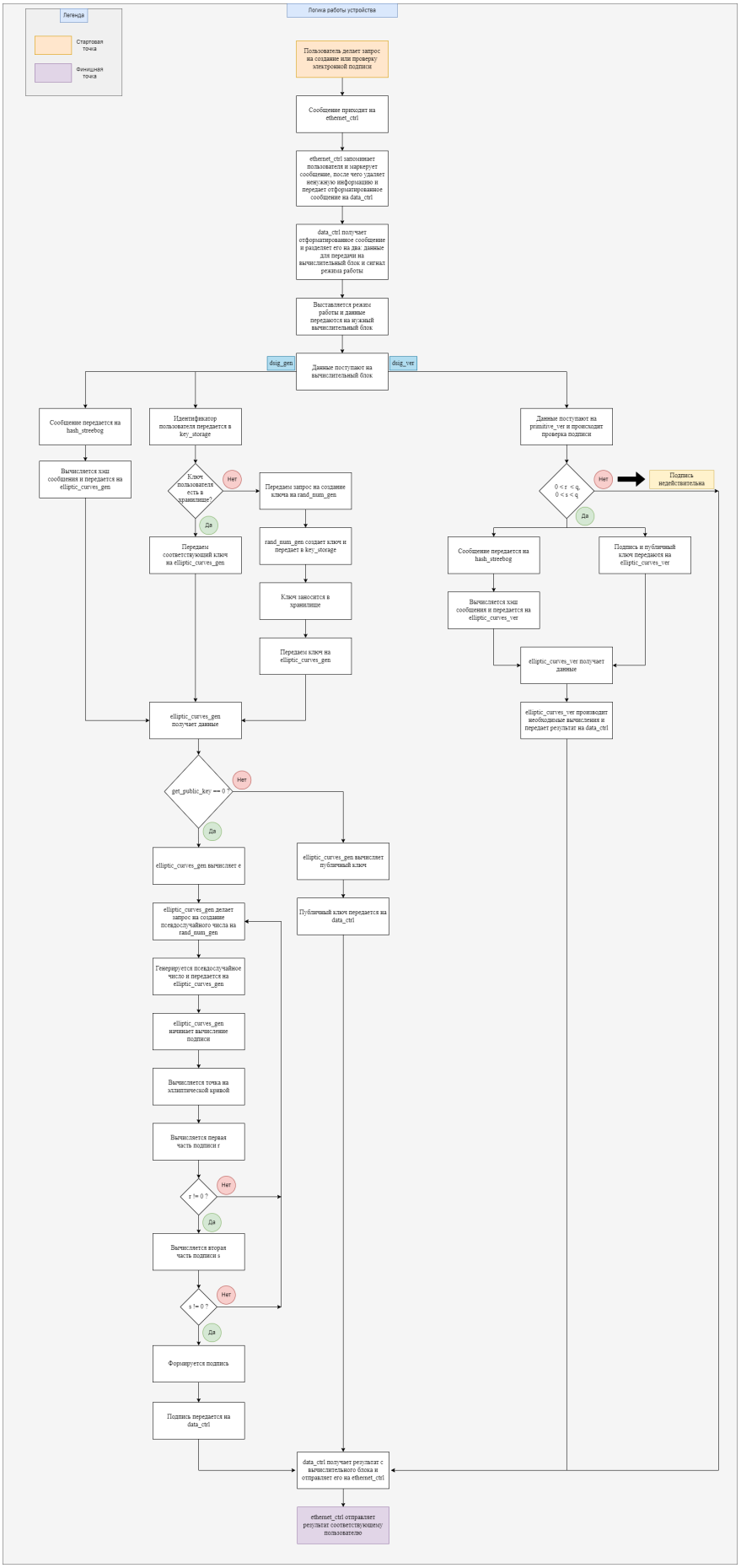

Рис. 4: Логика устройства

4.2 Написание RTL и его верификация

После создания архитектуры устройства начинается этап RTL-разработки. На этом этапе пишется код на языках описания аппаратуры (Verilog и SystemVerilog). Также важно проводить верификацию работы модулей с помощью тестовых программ. Весь код будет размещаться в git-репозитории по ссылке.

Из-за большого количества блоков процесс верификации устройства достаточно сложен. Чтобы упростить и ускорить его, сначала каждый блок будет тестироваться отдельно. После успешной верификации всех блоков их можно будет соединить и проверить работу сначала групп блоков, а затем и всего устройства в целом.

4.3 Написание временных ограничений

Следующим важным этапом является задание временных ограничений для модулей, включая определение тактовых сигналов, их задержек и других параметров. Это необходимо для корректной работы любого цифрового устройства. Ограничения помогают программе оптимально расположить логические элементы на ПЛИС, обеспечивая стабильную работу устройства при различных температурах и уровнях загруженности, которые влияют на распространение сигналов, переключение транзисторов и их характеристики. Без таких ограничений фронты сигналов могут «поплыть», из-за чего устройство либо не запустится, либо будет работать нестабильно.

4.4 Имплементация RTL на ПЛИС и верификация устройства

После написания логики и временных ограничений устройства необходимо их имплементировать на ПЛИС. В качестве рабочей платы будет использоваться RZ301 EP4CE6 development board от китайского производителя микроэлектроники OMDAZZ. Основной чип платы — Cyclone IV EP4CE6E22C8N, содержащий более 6000 логических элементов. К плате также можно подключить Ethernet-модуль и карту памяти.

Заключительный этап разработки — верификация работы готового устройства. Важно отметить, что фактический процесс реализации проекта гораздо сложнее, и описанные выше этапы представляют собой только основные шаги, порядок которых может варьироваться.

4.5 Возможные проблемы и их решение

Во-первых, могут возникнуть трудности с реализацией запланированной архитектуры, и, возможно, потребуется добавить новые блоки или, наоборот, интегрировать некоторые блоки в состав других.

Во-вторых, сложности могут возникнуть при реализации определённого функционала; в таком случае потребуется исследовать альтернативные способы решения поставленных задач.

В-третьих, возможны трудности с имплементацией логики на плату: например, может не хватать логических элементов на чипе или необходимых интерфейсов и памяти. В этом случае оптимальным решением станет покупка более продвинутой платы.

4.6 Возможные покупатели

Прежде всего, такое устройство будет полезно для малых и средних компаний, где ошибки сотрудников могут привести к серьезному ущербу. Чтобы повысить ответственность сотрудников за выполненную работу, можно внедрить систему подписания документов, кода и других материалов с помощью электронной подписи. Подписанный сотрудником документ означает его готовность взять на себя ответственность за его содержание.

Почему наше устройство хорошо подходит для этой цели? Во-первых, оно может находиться в защищенном месте в офисе компании. Во-вторых, можно настроить доступ к устройству толь-

ко из корпоративной сети. В-третьих, для использования подписей только внутри компании не требуется квалифицированная подпись. В-четвертых, список сотрудников компании меняется нечасто, поэтому реализованная система хранения ключей будет достаточно эффективной.

5 Преимущества и вызовы при использовании ПЛИС для цифровой подписи

Использование ПЛИС для реализации алгоритмов создания и проверки цифровых подписей предоставляет множество преимуществ. Одним из главных является высокая производительность, обусловленная параллельной архитектурой. В отличие от процессоров общего назначения, которые выполняют инструкции последовательно, ПЛИС могут одновременно обрабатывать несколько потоков данных, что существенно ускоряет выполнение криптографических операций. Это особенно полезно при работе с большими объёмами данных или в условиях ограниченного времени. Гибкость ПЛИС позволяет адаптировать их под различные задачи и быстро внедрять новые алгоритмы, что даёт значительное преимущество в изменяющихся условиях, где требования к безопасности могут меняться. Например, в случае обнаружения уязвимости в каком-либо алгоритме цифровой подписи, ПЛИС можно перепрограммировать для использования другого, более безопасного алгоритма без необходимости замены аппаратного обеспечения.

Однако использование ПЛИС также связано с рядом вызовов. Одним из них является сложность разработки. В отличие от стандартных процессоров, программирование ПЛИС требует специфических навыков и глубокого понимания аппаратных средств. Разработка схемы на ПЛИС включает проектирование на уровне логики и требует использования специализированных языков описания аппаратуры. Другим вызовом является ограниченность ресурсов ПЛИС. В зависимости от конкретной модели, ПЛИС могут иметь ограниченное количество логических блоков и памяти, что может стать ограничением при реализации сложных криптографических алгоритмов. Это требует от разработчиков тщательной оптимизации схемы для наиболее эффективного использования доступных ресурсов. Кроме того, стоимость ПЛИС может быть выше по сравнению с традиционными процессорами, особенно если требуется большое количество таких устройств для массового производства. Однако это компенсируется их гибкостью и возможностью адаптироваться к изменяющимся условиям.

6 Заключение

Реализация алгоритмов создания и проверки цифровой подписи на ПЛИС представляет собой мощное и перспективное решение для задач, связанных с криптографией и обеспечением безопасности данных. Благодаря высокой производительности, параллельной обработке и гибкости, ПЛИС предоставляют значительные преимущества перед традиционными процессорами, особенно в условиях, где критически важны скорость и надёжность. Тем не менее, успешная реализация подобных алгоритмов требует преодоления ряда вызовов, включая сложность разработки и ограниченные ресурсы устройств. Оптимизация под конкретные задачи и эффективное использование ресурсов ПЛИС позволяет достичь максимальных показателей производительности и безопасности. Перспективы использования ПЛИС для реализации цифровой подписи весьма широки, и с развитием технологий эта область может стать ещё более востребованной, особенно в контексте новых угроз безопасности и требований к защите информации.

7 Список литературы

1. ГОСТ 34.10-2018. Информационная технология. Криптографическая защита информации. Процессы формирования и проверки электронной цифровой подписи

2. Cyclone IV Device Handbook
3. Cyclone IV Device Datasheet
4. https://www.mdpi.com/2076-3417/12/1/378
5. https://www.mdpi.com/2079-9292/6/2/46
6. https://www.mdpi.com/2410-387X/6/2/25
7. https://arxiv.org/pdf/2112.02229