УДК 004.31

ПРОЕКТИРОВАНИЕ КОМПАКТНОГО РЕКОНФИГУРИРУЕМОГО *PUF*-КОНТРОЛЛЕРА

Трубач К.И., Иванюк А.А.

Белорусский государственный университет информатики и радиоэлектроники, Минск, Республика Беларусь, xenona11x@gmail.com

Аннотация: разработан реконфигурируемый блок АЛУ для *PUF*-контроллеров, обеспечивающих генерацию случайных чисел и идентификацию устройств в *IoT*. Объединение общих вычислительных примитивов (*LFSR*, *XOR*-деревья, мажоритирование) в едином тракте данных позволяет сократить площадь на *FPGA Artix-7* на 57% по сравнению с раздельной реализацией. При этом микропрограммная перенастройка алгоритмов пред- и постобработки позволяет оперативно адаптироваться к новым криптографическим требованиям без изменения схемотехники.

Ключевые слова: физически неклонируемые функции, реконфигурируемый контроллер, *IoT*.

I. ВВЕДЕНИЕ

Физически неклонируемые функции (*Physically Unclonable Function, PUF*) представляют собой цифровые схемы, чрезвычайно чувствительные к неизбежным технологическим разбросам при производстве. Одна и та же копия схемы на двух разных кристаллах (или даже в разных местах одного кристалла) ведёт себя по-разному, порождая уникальную пару «запрос–ответ» (*Challenge–Response*). Математически *PUF* описывается как *Response=PUF(Challenge)*, где *Challenge* – вектор запроса, *Response* – вектор ответа, а *PUF* – нелинейное отображение, определяемое физическими неоднородностями кристалла. Эти функции служат основой для аппаратной привязки к индивидуальным особенностям чипа и находят применение в самых разных задачах безопасности: генерации истинно случайных чисел (*TRNG*) для формирования ключей шифрования, идентификации устройств (*ID*), аппаратной реализации хэш-функций и других криптографических операций.

Однако каждая *PUF*-функция приобретает собственный набор узкоспециализированных блоков (*BCH*-коды, постобработка *TRNG*, хэширование), что увеличивает площадь кристалла и исключает модификацию алгоритмов без перепроектирования [1–4]. В литературе встречаются отдельные попытки динамически переключать одну *PUF*-структуру между режимом *TRNG* и *PUF*-идентификации [5], однако такое переключение реализуется, как правило, мультиплексированием двух составных частей, а не переиспользованием общих вычислительных ресурсов ради минимизации накладных расходов. В то же время в смежных областях криптографии и безопасности уже применяются реконфигурируемые криптопроцессоры, основанные на простейшем блоке АЛУ и микрокоде со специализированными операциями [6]. Применительно к *PUF* работа, вводящая понятие «потока реконфигурации» для контроллера, отсутствует. Под таким потоком понимается смена микрокода и перекоммутация тракта данных, позволяющие переключать устройство между различными приложениями (или их комбинацией), изменять настройки предобработки запроса и постобработки ответа, причём так, чтобы результирующая архитектура занимала ощутимо меньшую площадь, чем совокупность отдельных узкоспециализированных модулей.

В устройствах Интернета вещей (*IoT*), которые часто включают в себя *PUF* и средства управления ими, остро стоят вопросы энергоэффективности и минимизации площади кристалла. Поэтому данная работа ставит своей целью уменьшить аппаратные затраты на управление *PUF*, сохранив при этом гибкость для адаптации к новым протоколам и алгоритмам. В качестве решения предлагается выделить общие вычислительные примитивы, используемые во всех целевых приложениях (линейный сдвиговый регистр с обратной связью *LFSR*, *XOR*-дерево, мажоритарный элемент, сдвиговые регистры и т.п.), и объединить их в единый микропрограммируемый реконфигурируемый блок АЛУ. На примере рисунка 1 a) такими общими вычислительными примитивами являются функции предобработки *Challenge E*, *D* и постобработки *Response A*, *B*, *C*. Видно, что функции используются в блоках в различных комбинациях и при этом встречаются повторно, что и является примером упомянутой ранее избыточности. Потребление ресурсов можно сократить, если объединить эти блоки в общем тракте данных, как показано на рисунке 1 b). Для коммутации функций вводится дополнительный блок *Configuration Microcode*, хранящий настройку всего модуля и позволяющий переключать устройство между различными функциями (или их комбинацией), изменять настройки предобработки запроса и постобработки ответа без изменения аппаратуры.

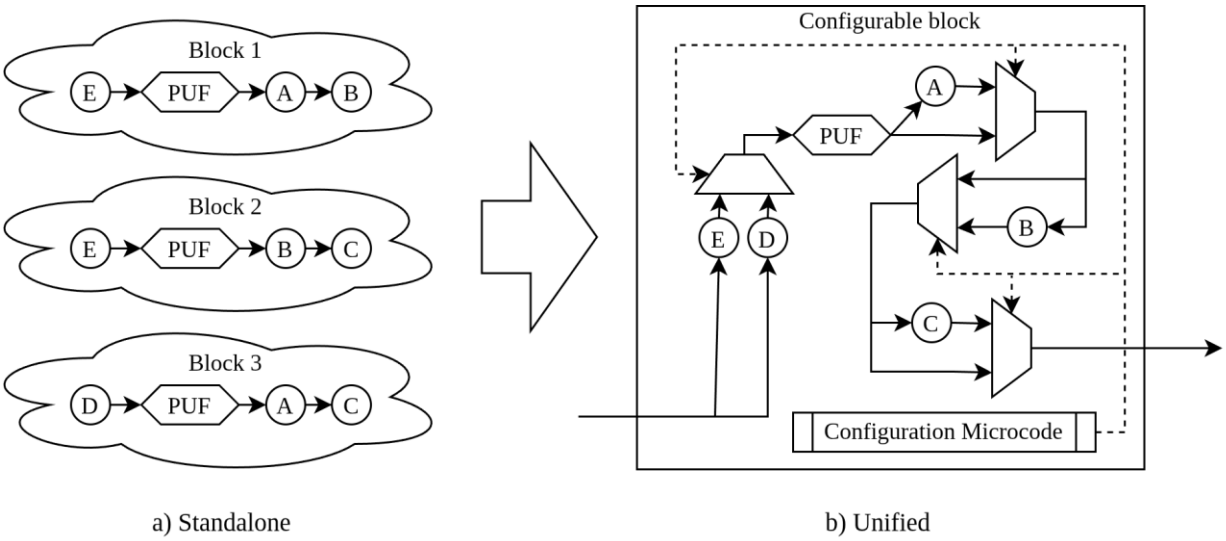


Рисунок 1. Пример перехода от архитектуры с разделяемыми функциями к архитектуре с реконфигурацией аппаратуры и переиспользованием аппаратных блоков

В итоге такой подход даёт возможность адаптироваться к новым требованиям за счёт обновления микропрограммы, а не полного перепроектирования и повторной верификации, что особенно ценно для компактных и энергоэффективных аппаратных корней доверия.

II. ПОСТАНОВКА ЗАДАЧИ

Поскольку физически неклонируемые функции – не одна фиксированная схема, а обширное семейство структур, перед началом проектирования необходимо унифицировать их внешний интерфейс, чтобы впоследствии в надстраиваемое окружение можно было помещать любой вид *PUF* без перестройки всей системы. Как показал анализ имеющихся типов *PUF*, их можно обобщить до схемотехнической модели с минимальным интерфейсом: разрешение работы (*Enable*), сброс (*Reset*), запрос (*Challenge*) и ответ (*Response*). На основе этого интерфейса требуется реализовать как минимум два приложения: идентификацию и генерацию истинно случайных чисел (*TRNG*). Задача – синтезировать микропрограммируемый контроллер, реконфигурирующий предобработку *Challenge* (смена разрядности, алгоритм генерации последовательности) и постобработку *Response* (мажоритарное декодирование, декоррелятор для *TRNG*, *BCH*-кодер или алгоритмы *Helper Data* для *ID*, аддитивный *XOR* для хэша аутентификации). От такого контроллера ожидается сокращение площади по сравнению с суммой отдельных реализаций тех же приложений.

Для обоснования реализуемости и эффективности такого подхода необходимо предварительно исследовать алгоритмическую общность целевых приложений, выделить повторяющиеся вычислительные примитивы (*LFSR*, мажоритарные элементы, деревья *XOR* и т.п.) и количественно оценить потенциальный выигрыш в аппаратных затратах при их совместном использовании. При необходимости адаптации к новому протоколу или уточнённым алгоритмам такой контроллер будет переконфигурирован на уровне микрокода, что открывает путь к более быстрой разработке компактных и энергоэффективных многофункциональных блоков безопасности.

III. АЛГОРИТМИЧЕСКИЙ АНАЛИЗ ЦЕЛЕВЫХ ПРИЛОЖЕНИЙ И ВЫДЕЛЕНИЕ ОБЩИХ ВЫЧИСЛИТЕЛЬНЫХ ПРИМИТИВОВ

Зафиксировав единый интерфейс блока *PUF* (E*nable*, *Reset*, *Challenge*, R*esponse*), перейдём к рассмотрению алгоритмических надстроек, необходимых для реализации двух и более приложений. Минимально достаточный набор функций, покрывающий ключевые потребности безопасности *IoT*, включает два аспекта. Первый касается алгоритмической природы формирования *Challenge*: это может быть линейный перебор с заданным шагом (положительным или отрицательным) либо псевдослучайная генерация на основе *M*-последовательности или простейших некриптографических хэш-функций (например, функция *djb2*), цель которых – генерировать значения, относительно равномерно распределенные по всему пространству. Второй аспект – постобработка *Response*, которая включает восстановление ответа мажоритарным декодированием, кодами Хэмминга и *BCH*, сжатие с потерей для получения компактной сигнатуры с помощью одно- или многовходового сигнатурного анализатора (*SISR* и *MISR* соответственно), перемешивание битов через *LFSR*.

Следует отметить, что некоторые из перечисленных блоков и алгоритмов имеют структурные пересечения как на низком уровне декомпозиции (например, *LFSR* выступает не только самостоятельной функцией, но и составной частью *SISR* и *MISR*), так и на более высоком (*XOR*-дерево как простейший аппаратный примитив многократно встречается в структуре многих функций. Выбор уровня декомпозиции АЛУ представляет собой компромисс между производительностью и аппаратными затратами. В связи с этим представляет интерес поиск компромиссного решения, объединяющего рассматриваемые функции в рамках единой архитектуры.

Ключевым соображением при таком объединении является то, что вынесение любой логики во внешний интерфейс превращает её в отдельную инструкцию, и чем мельче гранулярность таких инструкций, тем больший объём микрокода потребуется. Поэтому целесообразно начинать с наиболее крупных пересечений между приложениями. В рамках предложенного подхода были реализованы модификации: объединение *LFSR*, *SISR* и *MISR* в общий регистр с мультиплексированием входов; использование этого же *LFSR* в тракте хэш-функции (*djb2*) и *BCH*-кодера вместо собственных сдвиговых регистров; выделение *XOR*-деревьев в отдельную микрооперацию АЛУ. Была разработана программная модель, имитирующая работу объединённого блока АЛУ, что позволило на ранних этапах проектирования получить нижнюю границу ожидаемого выигрыша без необходимости реализации каждой комбинации на аппаратном уровне. Моделирование показало, что описанные ранее модификации дают соответственно 8%, 14% и 1% сокращения ресурсов, а введение общего регистрового файла добавляет ещё 20%, что в сумме даёт 43% консервативной оценки. Таблица 1 указывает, какие аппаратные блоки могут быть переиспользованы разными функциями.

Таблица 1. Соответствие «функция – требуемые вычислительные примитивы»

Функция	*LFSR*	Сумматоры	*XOR*-дерево	Элементы памяти
Линейный выбор с шагом		+		+
Хэш-функция	+	+	+	+
Мажоритарное декодирование				
Сжатие сигнатурным регистром	+		+	
Перестановка бит	+			+
Коды *BCH*	+			
Коды Хэмминга			+	

Исключение таблицы составляет лишь мажоритарное декодирование, которое имеет компактную реализацию и не пересекается в этом с другими функциями. Таким образом, алгоритмический анализ уже на верхнем уровне демонстрирует потенциальную возможность сокращения аппаратных затрат за счёт мультиплексирования вычислительных ресурсов. Численная оценка сокращения, полученная на основе результатов синтеза для *FPGA*, и будет представлена в следующем разделе.

IV. ЭКСПЕРИМЕНТАЛЬНАЯ ОЦЕНКА ЭФФЕКТИВНОСТИ ПРЕДЛОЖЕННОГО ПОДХОДА

Для проверки гипотезы проведён синтез двух вариантов на *FPGA Artix-7*. Первый вариант (далее – *Standalone*) включает все исходные специализированные модули, перечисленные в разделе III, инстанциированные независимо друг от друга, как это принято в традиционных подходах [5]. Второй вариант (далее – *Unified*) представляет собой разработанный микропрограммируемый блок АЛУ, объединяющий все элементарные вычислительные примитивы в одном блоке с помощью микрокода. В таблице 2 приведены результаты синтеза по основным типам ресурсов для обоих вариантов.

Таблица 2. Сравнение аппаратных затрат на *FPGA Artix-7*

Ресурс	*Standalone*	*Unified*	Сокращение затрат ресурсов
LUT	596	309	48%
Триггеры	250	128	49%
Мультиплексоры	132	28	79%
Общее число блоков	1596	682	57%

Из таблицы видно, что унифицированный блок АЛУ позволяет сократить число логических элементов более чем в два раза. Наиболее значительная экономия достигается за счёт устранения дублирования триггеров и *LUT*. Особенно показательна экономия мультиплексоров (79%), что объясняется заменой множества локальных коммутаторов на единый управляемый тракт данных.

Следует отметить, что в обоих вариантах функциональное поведение осталось эквивалентным: все целевые операции могут быть выполнены последовательностью микроопераций в *Unified*-версии, тогда как в *Standalone* каждая функция реализована жёстко и независимо и реализуется одной

инструкцией. Приведённые результаты подтверждают целесообразность предложенного подхода для устройств с ограничениями по площади и энергопотреблению, характерных для *IoT*-устройств.

V. ЗАКЛЮЧЕНИЕ

Полученные в ходе синтеза данные демонстрируют, что микропрограммируемый реконфигурируемый блок АЛУ способен обеспечить значительное (до 57%) сокращение аппаратных ресурсов по сравнению с традиционной реализацией набора специализированных блоков, что напрямую влияет на стоимость кристалла и на статическое энергопотребление. Кроме того, предложенная архитектура обладает свойством масштабируемости: добавление новой функции не требует изменения аппаратной части, а сводится к обновлению содержимого управляющей памяти (микрокода) контроллера без модификации его аппаратного ядра. Дальнейшие направления работы включают реализацию полного контроллера с интеграцией реального *PUF*-ядра и внешних интерфейсов, оценку энергопотребления на основе синтеза в *ASIC* с использованием низкопотребляющих библиотек примитивов, а также разработку автоматизированной среды генерации микрокода на основе высокоуровневого описания алгоритма.

Таким образом, представленные результаты доказывают состоятельность идеи реконфигурируемого микропрограммируемого контроллера для *PUF*-приложений и создают основу для построения компактных и энергоэффективных аппаратных корней доверия для устройств *IoT*. Работа вносит вклад в развитие методов проектирования специализированных вычислителей в области *PUF*, объединяющих в себе гибкость программного управления и эффективность аппаратной реализации.

БЛАГОДАРНОСТЬ

Авторы выражают искреннюю благодарность резиденту ПВТ компании «Инженерный Центр Ядро», которая является одним из центров разработки YADRO, за предоставленное оборудование для проведения экспериментов в рамках работы совместной учебной лаборатории с Белорусским государственным университетом информатики и радиоэлектроники.

ЛИТЕРАТУРА

[1] Intrinsic-ID [Электронный ресурс] // Cybersecurity Intelligence. – URL: https://www.cybersecurityintelligence.com/intrinsic-id-1409.html (дата обращения: 17.08.2026).

[2] Baturone I., Román R., Corbacho Á. A unified multibit PUF and TRNG based on ring oscillators for secure IoT devices //IEEE Internet of Things Journal. – 2022. – Т. 10. – №. 7. – С. 6182-6192.

[3] Synopsys DesignWare TRNG Security IP [Электронный ресурс] // Synopsys. – URL: https://www.synopsys.com/designware-ip/security-ip/cryptography-ip/random-number-generators/trng.html (дата обращения: 17.08.2026).

[4] Huang Z., Wang Q. A PUF-based unified identity verification framework for secure IoT hardware via device authentication //World Wide Web. – 2020. – Т. 23. – №. 2. – С. 1057-1088.

[5] Taneja S., Rajanna V. K., Alioto M. In-memory unified TRNG and multi-bit PUF for ubiquitous hardware security //IEEE Journal of Solid-State Circuits. – 2021. – Т. 57. – №. 1. – С. 153-166.

[6] Grand M. et al. A reconfigurable multi-core cryptoprocessor for multi-channel communication systems //2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum. – IEEE, 2011. – С. 204-211.

ARCHITECTURE DESIGN OF A COMPACT RECONFIGURABLE PUF CONTROLLER

K.I. Trubach, A.A. Ivaniuk

Belarusian State University of Informatics and Radioelectronics, Minsk, Republic of Belarus,
xenona11x@gmail.com

Abstract: this paper presents a reconfigurable ALU for PUF controllers designed to provide random number generation and device identification in IoT systems. By combining common computational primitives (LFSR, XOR trees, and majority logic) into a unified data path, the proposed approach reduces the area overhead on the Artix‑7 FPGA by 57% compared with a separate implementation of the same functions. Furthermore, microprogrammable reconfiguration of pre‑ and post‑processing algorithms enables rapid adaptation to new cryptographic requirements without hardware redesign.

Keywords: physically unclonable functions, reconfigurable controller, IoT.