Решение задачи идентификации проектов, реализованных на ПЛИС

Прощеряков А.А.; Иванюк А.А.
Кафедра вычислительных методов и программирования
Белорусский государственный университет информатики и радиоэлектроники
Минск, Республика Беларусь
e-mail: {proshcheryakov, ivaniuk}@bsuir.by

***Аннотация*—Представлены результаты эксперимента по идентификации проекта, реализованного для программируемой логической интегральной схемы с помощью физически неклонируемой функции типа арбитр. Подтверждается улучшение качества идентификации цифрового устройства при использовании модифицированного арбитра.**

Ключевые слова: программируемая логическая интегральная схема, физически неклонируемая функция, модифицированный арбитр.

I. Введение

Применение программируемых логических интегральных схем (ПЛИС) в качестве аппаратной базы для цифровых устройств ограничивается проблемами защиты интеллектуальной собственности разработчиков. Проблема защиты от несанкционированных действий пользователя приобретает больший масштаб, поскольку особенность ПЛИС заключается в возможности получения доступа к конфигурационному файлу проекта и его изменению [1].

Производство большинства серийно выпускаемых ПЛИС не предусматривает включение в чип регистра уникального идентификатора (ID), их пользовательская реализация не может быть защищена от изменения и клонирования. Поэтому в качестве ID предлагается использовать физически неклонируемые функции (Physical Unclonable Function, PUF), которые регистрируют мельчайшие особенности прохождения сигналов по структурным элементам ПЛИС, возникающих из-за физической уникальности последних, обусловленной технологическим процессом производства интегральных схем [1].

II. Физически неклонируемая функция типа арбитр

В качестве PUF, регистрирующей незначительные вариации элементов ПЛИС, предлагается использовать PUF типа арбитр.

В общем случае PUF типа арбитр реализуется в виде множества последовательно соединённых конфигурируемых блоков (последовательности мультиплексоров), направление сигналов в которых определяется настроечными значениями C_i. На вход конфигуриемого пути подаются два идентичных одиночных импульса. На выходе устанавливается арбитр, построенный на D-триггере и регистрирующий опережение одного импульса другим [1].

Предлагается использование модифицированного арбитра, состоящего из четырёх триггеров, регистрирующих не только опережение фронтов одного сигнала другим, но и взаимное расположение спадов импульсов. Схема модифицированного арбитра представлена на рис.1

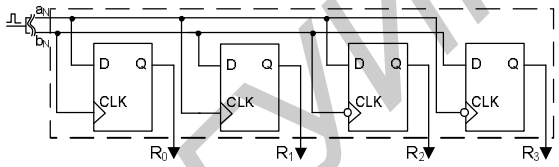

Рис. 1. Схема модифицированного арбитра

Предлагаемый арбитр позволяет получать множество ниблов R (Response), при различном взаимном расположении импульсов a и b (рис. 2).

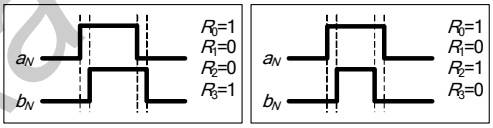

Рис. 2. Изменение R при различных формах импульсов a и b

Для проверки гипотезы о возможности идентификации ПЛИС с помощью предложенной PUF был проведён эксперимент.

III. Эксперимент по идентификации ПЛИС

A. Схемная реализация PUF

Для постановки эксперимента в качестве аппаратной базы была выбрана отладочная платформа Digilent Basys2, на основе ПЛИС Xilinx Spartan-3E.

Для генерации множества идентификационных ниблов R, хранения и передачи его в PC, на языке описания аппаратуры VHDL был реализован проект в САПР Xilinx ISE WebPACK, схема которого представлена на рисунке 3.

Память устройства RegMemory реализована на базе 256-и 8-ми разрядных регистров. В данные регистры записываются ответы R с PUF по адресам, совпадающим с соответствующими значениями C, генерируемые управляющим конечным автоматом Control FSM.

Управление автоматом Control FSM и чтение массива ответов R осуществляется посредством асинхронного параллельного интерфейса Digilent (DEPP), реализованного на базе USB интерфейса. Управление DEPP выполняется C++ приложением посредством Adept API-функций [2].

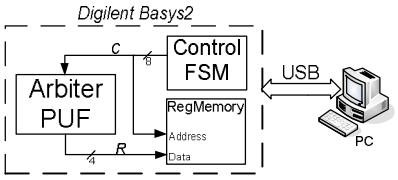

Рис. 3. Схема экспериментального проекта

B. Генерация экспериментальных множеств R

Для проведения эксперимента первоначально был сгенерирован конфигурационный bit-файл с автоматическим распределением логических элементов по слайсам в ПЛИС. Затем выполнена генерация 100 полных множеств *R*: по 256 ответов для различных настроечных значений *C*. Для каждой последовательности *C* PUF-схема генерировала новое множеств ответов *R*.

Затем с помощью FPGA Editor было изменено расположение одного элемента PUF, смоделировав этим небольшое изменение проекта. После чего выполнено повторение генерации 100 последовательностей.

В третьем опыте генерация множества *R* проводилась для проекта с полностью изменённым расположением элементов PUF, сгруппированных более компактно.

C. Анализ результатов

В ходе вышеописанных действий получено три таблицы выходных значений PUF размером 256x100 значений, для одной и той же аппаратуры и практически одинаковых проектов.

Для каждого значения настроечных *C* найден наиболее часто встречающийся вариант ответа.

Сравнив данные каждого из опытов, получено, что ответы PUF первого и второго опытов не совпадают в 151 случае (58,9%), первого и третьего — в 233 случаях (91,0%), второго и третьего — в 240 случаях (93,8%). Полученный результат может свидетельствовать о возможности определения подлинности проекта.

Проведён анализ получения различных значений ответов PUF. Результаты представлены на гистограмме (рис. 4).

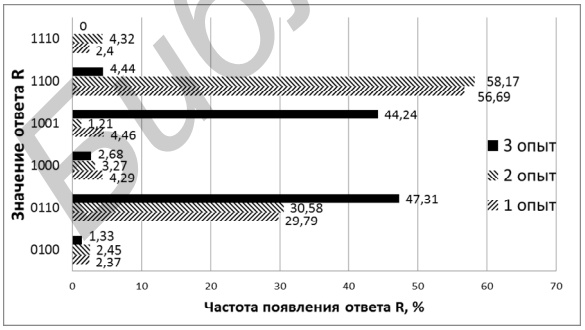

Рис. 4. Гистограмма частот появления различных R

Наличие в результате эксперимента таких несимметричных ответов как 0100, 1000, 1100, 1110 свидетельствует об изменении не только относительного положения сигналов, но и об изменении их скважности, но не значительного, поскольку ответ 1010 не появлялся. Использование пар триггеров с инверсным подключением сигналов позволяет отследить незначительные смещения сигналов, поскольку сама реализация триггера в ПЛИС предполагает проведение сигнала установки и сброса триггера по различным путям, через мультиплексоры с различной задержкой сигналов.

Также эксперимент подтверждает улучшение качества идентификации четырьмя триггерами в сравнении с одним. Второй триггер позволил различить случаи 0100 и 0110; третий — 1000 и 1100; четвёртый — 0100 и 1100. Для первого опыта добавление трёх триггеров позволяет идентифицировать в три раза больше вариантов проектов (рис. 5).

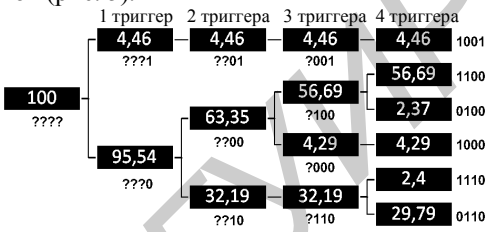

Рис. 5. Диаграмма процентного разделения ответов *R* на идентификационные группы с увеличением числа триггеров в арбитре

Подобный эксперимент был проведён и с использованием двух однотипных платформ Digilent Nexys2, которые были сконфигурированы одним bit-файлом. В результате эксперимента множества ответов *R* отличались в 36 позициях, что говорит о возможности идентификации не только проектов, но и самих ПЛИС.

Стоит также отметить небольшое число появлений нестационарных ответов — значений *R* отличных от чаще встречающихся. Однако их наличие заставляет выполнять не один цикл генерации ответа, а несколько, с вычислением моды по каждому значению *C*. По результатам опытов можно говорить и о стабильности появления таких нестационарных ответов при каких-то определённых значениях *C*, меняющихся для каждой из реализаций проекта. Эту особенность можно также рассматривать как идентификационный признак, и строить идентификатор не на базе анализа генеральной совокупности *R*, а на выборочном анализе *R* в конкретных значениях *C*.

IV. Вывод

Проведённый эксперимент подтверждает возможность идентификации проектов для ПЛИС и самих ПЛИС при помощи PUF типа арбитр. Появление в результатах несимметричных ответов модифицированного арбитра указывает на обоснованность выбора четырёх триггеров для построения арбитра. Выдвинуто предположение анализа не всей выборки ответов PUF, а только тех, где возникают нестационарные ответы.

[1] Иванюк, А. А. Проектирование встраиваемых цифровых устройств и систем : монография / А. А. Иванюк. – Минск : Бестпринт, 2012. – 337 с.

[2] Adept 2.1.1 SDK, 32/64-bit Windows [Electronic resource]. – Mode of access: http://www.digilentinc.com/Data/Products/adept2/digilent.adept.sdk_v2.1.1.zip – Date of access: 10.10.2012