

Метод исследования реализации физически неклонированных функций в информационных системах

В.Р. Лебедев, Е.Ф. Певцов, Т.А. Деменкова, М.И. Малето, В.В. Филимонов

Аннотация – Разработан метод исследования реализации физически неклонированных функций (ФНФ) типа «арбитр», предназначенных для идентификации экземпляров интегральных схем программируемой логики, входящих в состав аппаратной части информационных систем для обеспечения их доверенного проектирования. ФНФ типа «арбитр» формируются как совокупность метастабильных состояний триггера в конце из двух идентичных последовательных цепей мультиплексоров, реализованных на основе блоков SLICE, входящих в микроархитектуру схем программируемой логики. При этом путь прохождения импульсного сигнала от входов цепей до триггера задается вектором запроса, подаваемым на входы мультиплексоров обеих цепей. Индикатором реализации ФНФ для данной схемы являются значения векторов ответов, фиксируемых при реализации метастабильного выхода триггера в конце цепей из мультиплексоров при одних и тех же конкретных значениях векторов запроса. Методика проектирования ФНФ и оригинальные программные коды для их реализации и исследования апробированы при проведении экспериментов со схемами программируемой логики, предназначенными, в частности, для применения в информационных системах. Разработаны программы обработки данных типа «запрос-ответ» в виде двумерной карты откликов. Исследованы зависимости повторяемости ФНФ от разрядности вектора запроса, расположения блоков внутри интегральной схемы программируемой логики и температуры окружающей среды.

Ключевые слова – элементы информационных систем, программируемые логические схемы, физически неклонированные функции, программы обработки данных.

Статья получена 17 октября 2023.

Работа выполнена при поддержке Министерства науки и высшего образования РФ (Государственное задание для университетов № ФГФ3-2023-0005).

В.Р. Лебедев – инженер Центра проектирования интегральных схем, устройств наноэлектроники и микросистем РТУ МИРЭА (e-mail: Slava-lebedev@outlook.com).

Е.Ф. Певцов – к.т.н., директор структурного подразделения РТУ МИРЭА «Центр проектирования интегральных схем, устройств наноэлектроники и микросистем» (e-mail: pevtsov@mirea.ru).

Т.А. Деменкова – к.т.н., ведущий научный сотрудник Центра проектирования интегральных схем, устройств наноэлектроники и микросистем, доцент кафедры вычислительной техники Института информационных технологий РТУ МИРЭА (e-mail: demenkova@mirea.ru).

М.И. Малето – к.т.н., ведущий научный сотрудник Центра проектирования интегральных схем, устройств наноэлектроники и микросистем РТУ МИРЭА (e-mail: maleto@mirea.ru).

В.В. Филимонов – ст. преподаватель, руководитель лабораторного практикума кафедры физики РТУ МИРЭА (e-mail: filimonov_v@mirea.ru).

I. ВВЕДЕНИЕ

В современных условиях, при создании проектов, включая аппаратную часть информационных систем, особое значение приобретает доверенное проектирование и производство, в частности проверка того, что применяемые комплектующие не являются контрафактными изделиями, а поставлены от настоящего производителя. Поскольку значительное большинство проектов цифровых устройств выполняется на основе интегральных схем с программируемой логикой, проверка их подлинности становится актуальной задачей.

Одним из подходов, реализующих доверенное проектирование, служит применение физически неклонированных функций [1].

Физически неклонированная функция (ФНФ, PUF – Physical Unclonable Function) – это функция которая обладает следующими свойствами:

- основана на физической системе;
- легко оценивается;
- ведет себя как случайная функция, непредсказуема даже в том случае если имеется физический доступ к системе;
- невозможно клонировать или воспроизвести ее на другой копии той же физической системы, даже если функциональность известна.

ФНФ могут обеспечить недорогую аутентификацию отдельных микросхем и генерировать энергозависимые секретные ключи для криптографических операций [2]. Технология ФНФ позволяет защитить отдельные IP-ядра микросхем с меньшими накладными расходами. Схемотехнические и аппаратные решения для реализации ФНФ рассмотрены в многочисленных публикациях последних лет [3-12].

В частности, в работе [13] рассмотрены вопросы оценки производительности для нескольких конструкций ФНФ на базе программируемой логики и их сравнения, приведены данные о некоторых известных атаках на ФНФ и соответствующих

контрмерах, приведен краткий обзор сценариев применения ФНФ на базе ПЛИС для доверенного проектирования информационных систем и будущих направлений исследований. Новый вариант улучшенный вариант построения ФНФ типа «двойной арбитраж», позволяющий повысить их непредсказуемость описан в работе [14].

II. АППАРАТНАЯ РЕАЛИЗАЦИЯ И МЕТОДИКА ИССЛЕДОВАНИЯ

В данной работе для применяется один из вариантов реализации ФНФ, а именно – схема типа «арбитр», идея которого иллюстрируется на рис.1.

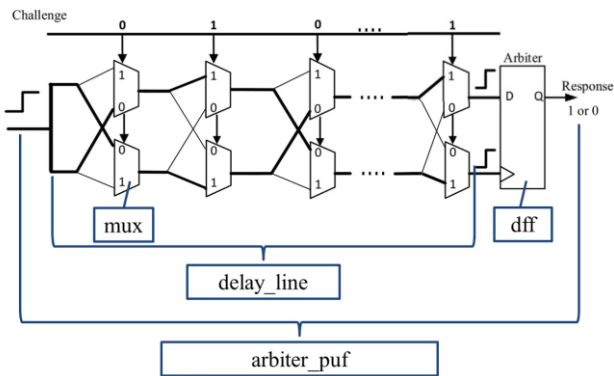


Рис.1. Иерархия модуля PUF типа «арбитр».

Схема ФНФ типа «арбитр», построена на мультиплексах. Обязательным условием ФНФ этого типа является прохождение сигнала через симметричные пути цифрового устройства. В этом случае ФНФ описывается парой "запрос-отклик".

Запрос (challenge) представляет собой N-битный вектор, который определяет физический путь, по

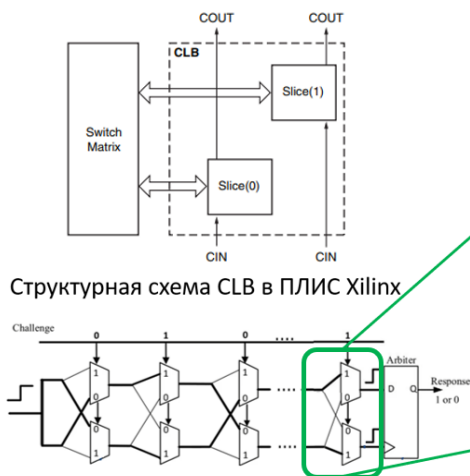
которому пройдет входной импульс (pulse). Ответом (response) будет являться сигнал, который установился на выходе D-триггера в результате прохождения импульса. Основным фактор, характеризующий ФНФ в данном виде – это различие задержек сигналов на элементах цепи, поскольку влияние на задержки со стороны производителя затруднено и экономически нецелесообразно. Результатом этого является индивидуальный отклик системы.

Индивидуальность отклика пар «запрос-отклик» (CRP – challenge-response pair), является ключевой характеристикой, которая позволяет использовать ФНФ для генерации уникальных идентификаторов.

На текущий момент основными производителями ПЛИС являются компании Altera (Intel) и Xilinx (AMD). Основным фактором, обусловившем выбор производителя ПЛИС Xilinx, в данном проекте является возможность задействовать минимальное количество элементов в стандартных блоках ПЛИС.

Основным ресурсом для реализации последовательной и комбинационной логики ПЛИС Xilinx являются конфигурируемые логические блоки CLB (Configurable Logic Block) [15]. Каждый CLB подключен к матрице коммутаторов для доступа к общей матрице маршрутизации. CLB содержит в себе два модуля типа SLICE. SLICE — это базовая логическая единица, разработанная компанией Xilinx, которая содержит: четыре таблицы истинности, восемь элементов хранения, многофункциональные мультиплексы и цепь переноса. Это позволяет реализовывать логические, арифметические функции и функцию ПЗУ. Для реализации PUF типа «арбитр» в данном проекте используются аппаратные мультиплексы. Это позволяет минимизировать путь прохождения сигнала и количество элементов в цепи. На рис.2 продемонстрировано прохождение двух сигналов в SLICE.

- CLB – Configurable Logic Block



Структурная схема CLB в ПЛИС Xilinx

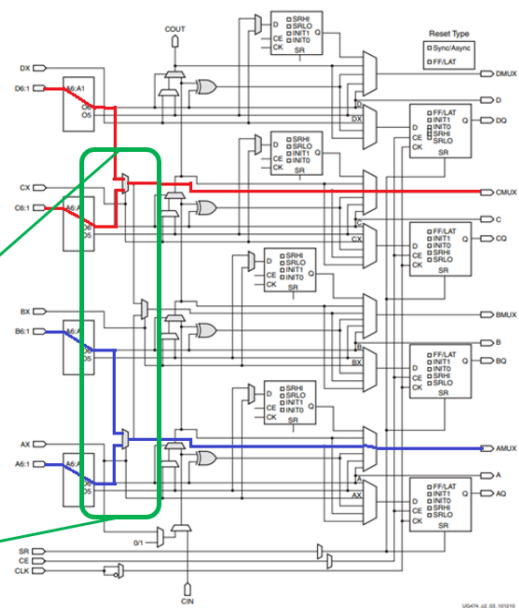


Рис.2. Архитектура ПЛИС 7-го поколения Xilinx – CLB и реализация симметричных путей прохождения

сигнала в модулях SLICE ПЛИС Xilinx.

III. ПРОГРАММНЫЕ МОДУЛИ ДЛЯ РЕАЛИЗАЦИИ ПРОЕКТА

Как показано на рис.2. в данной работе используются аппаратные мультиплексоры MUXF7. Код поведенческого описания блока мультиплексора:

```
(* DONT_TOUCH = "yes" *)
module mux(
  input a,
  input b,
  input sel,
  output out
);
(* DONT_TOUCH = "yes" *) MUXF7 MUXF7_inst (
  .O(out), // Output of MUX to general routing
  .I0(a), // Input (tie to LUT6 O6 pin)
  .I1(b), // Input (tie to LUT6 O6 pin)
  .S(sel) // Input select to MUX
);
endmodule
```

В этом коде атрибут DONT_TOUCH принимает значение «yes», чтобы предотвратить логическую оптимизацию схемы.

Для реализации однобитного D-триггера с синхронным сбросом (R) и разрешением работы (CE) использовался языковой шаблон FDRE:

```
FDRE_inst (
  .Q(out_dff), // 1-bit Data output
  .C(clk), // 1-bit Clock input
  .CE(1'b1), // 1-bit Clock enable input
  .R(1'b0), // 1-bit Synchronous reset input
  .D(d) // 1-bit Data input
);
```

В данном проекте для создания линии задержки delay_line цепи, состоящей из блоков мультиплексоров, используется конструкция языка generate, которая позволяет формировать повторяющиеся участки кода.

Фрагмент кода блока delay_line:

```
(* DONT_TOUCH = "yes" *) wire [2 * C_LENGTH + 1 : 0]net;
```

```
assign net[0] = pulse;
assign net[1] = net[0];
generate
genvar i;
for(i = 1; i <= C_LENGTH; i = i + 1)
begin
  mux inst_mux_1(
    .a(net[i * 2 - 2]),
    .b(net[i * 2 - 1]),
    .sel(challenge[i - 1]),
    .out(net[i * 2])
  );
  mux inst_mux_2(
    .a(net[i * 2 - 1]),
    .b(net[i * 2 - 2]),
    .sel(challenge[i - 1]),
    .out(net[i * 2 + 1])
  );
end
endgenerate
```

Для описания физически неклонированной функции типа арбитр необходимо на вход подавать управляющий сигнал (Challenge) и импульсный сигнал (Pulse), а с выхода снимать отклик системы (Response).

Для формирования сигналов и получение отклика используется низкоуровневый интерфейс ввода-вывода GPIO (general-purpose input/output), в котором для автоматизации процесса формирования управляющего сигнала и входного импульсного, а также считывания отклика применяется встроенной в ПЛИС IP-блок конвейерного 32-х битного soft-процессора процессора MicroBlaze с RISC архитектурой.

Обмен данными между ПК и процессорной системой осуществляется с помощью интерфейса UART (Universal asynchronous receiver/transmitter).

Структурная схема проекта для исследований ФНФ с фиксацией результатов в ПК представлена на рис.3.

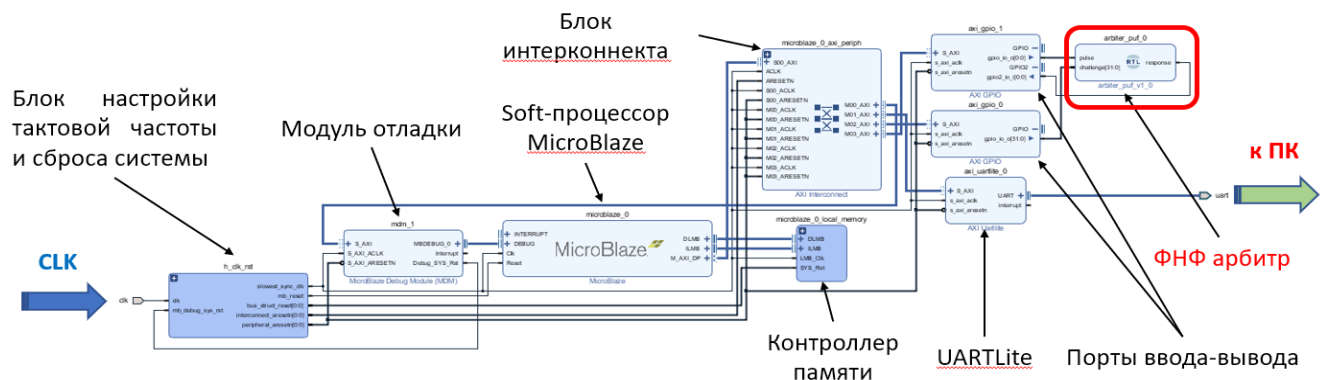


Рис.3. Структурная схема проекта для исследований ФНФ с фиксацией результатов в ПК. Сигналы, генерируемые процессорной системой: Challenge – представляет собой N-битный вектор. Определяет физический путь входного сигнала; Pulse – Входной импульс подаваемый на вход ФНФ; Response – ответ ФНФ получаемый с выхода D-триггера; UART – обмен данными между ПК и процессорной системой.

Назначение блоков и сигналов процессорной системы:

- **clk_wiz_1** – блок позволяет настроить тактовую частоту soft-процессора. В данном случае подключаем на вход тактовый сигнал кварцевого генератора платы Artix-35T – 100МГц. По умолчанию на вход clk_wiz_1 подается дифференциальный тактовый генератор. В настройках блока изменяем вход на однополярный и подключаем к внешнему порту. Сигнал locked сигнализирует о захвате ФАПЧ тактового сигнала.
- **rst_clk_wiz_1_100M** - блок сброса системы. Отвечает за корректный сброс процессора и соответствующей периферии. Также в этот блок подается сигнал locked и самый медленный тактовый сигнал в системе. Так как в нашей системе только один тактовый сигнал, то с выхода clk_wiz_1 подаем и на блок сброса системы, поскольку относительно самого медленного сигнала происходит сброс. В данном блоке есть возможность подключения внешнего сигнала сброса для исключения ошибки при сборке системы был добавлен блок констант и подключен к входам ext_reset_in и aux_reset_in. Внешний сброс осуществляется логическим нулем. mb_debug_sys_rst – дополнительный сигнал сброса от отладчика системы MicroBlaze. Mb_reset – сброс процессора. Оставшиеся выходные сигналы отвечают за сброс периферии.\
- **Microblaze_0_local_memory** – контроллер памяти. В данном модуле находится память данных и инструкции для процессора.
- **Microblaze_0** – soft-процессор. В настройка процессора можно выбрать определенные предустановки. В данном случае настройки устанавливаем настройки по умолчанию.
- **Microblaze_0_axi_periph** – блок интерконнектора. Поскольку используется стандартная «point to point connection» шина AXI, в которой нельзя на одно ведущее устройство (master) подключить несколько ведомых устройств (slave), данный блок создает отдельное адресное пространство, что позволяет подключить одно или несколько ведущих устройств к одному и нескольким ведомым устройствам.
- **Axi_uartlite_0** – отвечает за передачу данных от процессорной системы к компьютеру. Для создания использовался готовый IP-блок UARTLite. Шина AXI подключена к интерконнектору и реализован внешний порт UART (tx, tx).

Алгоритм работы процессорной системы представлен на рис.4.

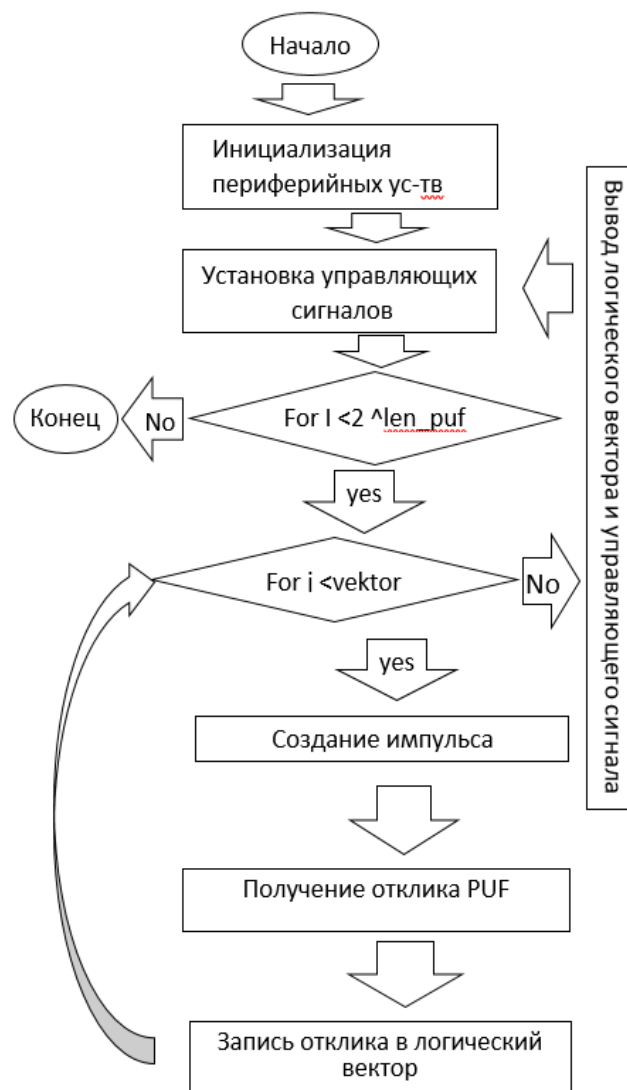


Рис.4. Блок-схема алгоритма работы процессорной системы.

Фрагмент кода процессорной системы, отвечающий за формирование входных сигналов и регистрацию отклика системы:

```
for (challenge = 0; challenge < 1000000; challenge++){
    set_challenge(challenge); // устанавливаем управляющий
    //сигнал PUF ТИПА «АРБИТР» на каждой итерации

    resp32 = 0;
    // подготовка логического вектора к записи
    for(i = 0; i < 32; i++){
        create_pulse(); // создаем 32 импульса
        response = get_response();
        //считываем ответ системы 32 раза
        resp32 = resp32 | (response << i);
        // записываем ответ системы в логический вектор
    }
    // условие для вывода данных:
    if ((resp32 != 0x00000000) && (resp32 !=
    0xFFFFFFFF)){
        xil_printf("challenge = %d; response =
        %x;\n\r", challenge, resp32);
    }
}
```

Для формирования сигналов и получение отклика PUF

типа «арбитр» применяется два низкоуровневых интерфейса GPIO:

- **Axi_gpio_0** – служит для создания управляющего сигнала (challenge).
-
- **Axi_gpio_1** – для создания импульса (pulse) на вход и отклика (response).
- Контроль входных и выходных данных будет осуществляться с помощью встроенного в среду проектирования логического анализатора **ILA (Integrated Logic Analyzer)**.

Проектные ограничения, реализуемые в данном проекте, можно разделить на четыре группы:

1) Физическое ограничение размещения ФНФ типа «арбитр» в Pblock-e. Так как длина цепи и способ размещения могут изменяться, то для ускорения имплементации проекта применяется скрипт на языке Tcl:

```
set cell_path_0
{system_i/arbiter_puf_0/inst/inst_delay_line/genblk1}
set cell_path_1 {MUXF7_inst}
set start_loc_x 0 //стартовая позиция по x
set start_loc_y 101 //стартовая позиция по y
set current_loc_x 0 //текущее положение по x
set current_loc_y 0 //текущее положение по y
for {set i 1} {$i <= 32} {incr i} {
  set current_loc_x [expr $start_loc_x + $i]
  //изменение положения по x
  set current_loc_y [expr $start_loc_y + 0]
  //изменение положения по y
  startgroup
  place_cell "$cell_path_0[$i].inst_mux_1$cell_path_1"
  SLICE_X${current_loc_x}Y${current_loc_y}/F7BMUX
  endgroup

  startgroup
  place_cell "$cell_path_0[$i].inst_mux_2$cell_path_1"
  SLICE_X${current_loc_x}Y${current_loc_y}/F7AMUX
  endgroup
  // размещение пары мультиплексов
}
startgroup // размещение dff
place_cell
system_i/arbiter_puf_0/inst/inst_dff_0/dff_primitive.FDRE_inst
SLICE_X32Y101/B5FF
endgroup
```

2) Подключение портов ввода-вывода UART и тактовой частоты к процессорной системе

3). Физическое ограничение на размещение в ПЛИС процессорной системы и ФНФ типа «арбитр». Для этого применяется функция PBlock, формирующая набор ячеек и одну или несколько прямоугольных областей, которые определяют ресурсы устройства, содержащиеся в блоке. [16,17].

IV. МЕТОДИКА ЭКСПЕРИМЕНТА И РЕЗУЛЬТАТЫ

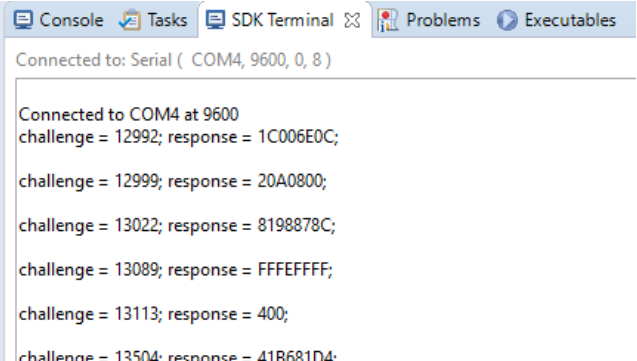
Для определения положения метастабильного состояния отклик системы собирается в 32-х битный логический вектор. При этом вектор, который будет состоять из 32-х логических 0 или 1 не будет нести информацию о ФНФ. Если временные параметры работы триггера time setup и time hold не нарушаются и сигнал на D-вход приходит раньше, то на выходе

сформируется логический вектор из 32-х логических единиц. Соответственно, если временные параметры не нарушаются и сигнал на D-вход приходит позже, то выходной вектор будет состоять из нулей. При нарушении параметров будет наблюдаться метастабильное состояние и выходной вектор будет состоять из единиц и нулей.

Эксперименты показали, что в общем случае отклик системы при заданном управляющем сигнале является вероятностной функцией и свидетельствует о метастабильном состоянии триггера на выходе. Чтобы исключить влияние метастабильного состояния триггера на вход PUF типа «арбитр» подается серия из 32-х импульсов при фиксированном значении управляющего сигнала. Отклик PUF типа «арбитр» записывается в 32-х битный логический вектор. Терминал для вывода информации подключен в режиме отладки в программе SDK. Целью эксперимента является выявление возможности генерирования уникальных идентификаторов в PUF типа «арбитр» с использованием цепей разной длины. Анализ направлен на определение, насколько различная длина цепи влияет на способность PUF генерировать уникальные идентификаторы.

В данном эксперименте производилось тестирование каждого возможного значения управляющего сигнала Challenge на отладочной плате Arty-A7 35T.

Пример фрагмента вывода ФНФ типа «арбитр» в терминал приведен на рис.5.



```
Connected to: Serial ( COM4, 9600, 0, 8 )
Connected to COM4 at 9600
challenge = 12992; response = 1C006E0C;
challenge = 12999; response = 20A0800;
challenge = 13022; response = 8198878C;
challenge = 13089; response = FFFFFFFF;
challenge = 13113; response = 400;
challenge = 13504; response = 41B681D4;
```

Рис. 5. Пример вывода ФНФ типа «арбитр» в терминал.

Для наглядности и последующего анализа полученные результаты выходного вектора ФНФ, представляются в виде карт откликов системы, пример которой приведен на рис.6.

Управляющий сигнал разделен на две части. Горизонтальная ось представляет собой первые четыре бита, а вертикальная вторые четыре бита управляющего сигнала Challenge. Справа от карты представлена нормированная шкала логического вектора отклика системы Response. Белый цвет на карте означает, что логический вектор Response представляет собой тридцать две логические единицы. Черный цвет представляет собой логический вектор, заполненный логическими нулями. Отличный от белого и черного –

метастабильное состояние D-триггера на выходе. Идентификатором ФНФ является положение соответствующего управляющего сигнала, который вызывает метастабильное состояние.

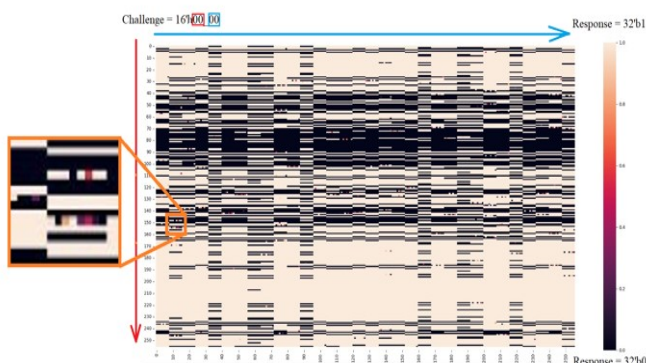


Рис.6. Карта откликов, полученная на основе пар «запрос-ответ» для длины цепи из 16 пар мультиплексоров.

В таблице 1 представлены результаты, полученные в результате обработки откликов системы, разбитые по категориям.

Таблица 1. Данные отклика ФНФ типа «арбитр», разделенные по категориям.

Код FPGA	Стабильная '1'	Стабильный '0'	Мета-стабильное состояние	Нестабильный отклик системы
D39	35473	28798	408	857
D26	35487	28757	420	872
EC3	34674	29631	402	829
D28	35897	28470	388	781
107	34621	29650	387	878
D3E	35243	28944	447	902
D48	35075	29249	449	763
D37	34596	29748	365	827
D44	34266	30053	407	810
D4B	35516	28762	441	817

Анализ полученных данных позволяют сделать следующие выводы:

- 1) Если логический вектор состоит из 32-х логических единиц, то задержка меньше до информационного входа D-триггера, чем до тактирующего.
- 2) Если логический вектор состоит из 32-х логических нулей, то задержка меньше до тактирующего входа D-триггера, чем до информационного.
- 3) Логический вектор содержит комбинацию нулей и единиц. В этом случае имеет место метастабильное состояние D-триггера, когда нарушается время предустановки (время, в течение которого сигнал на входе D должен оставаться стабильным перед приходом фронта тактового сигнала) и время удержания (время, в течение которого сигнал на входе D должен оставаться стабильным после прихода фронта тактового сигнала).

Для обработки данных использовался объектно-

ориентированный язык программирования Python. Блок-схема программы обработки информации представлена на рис.7.

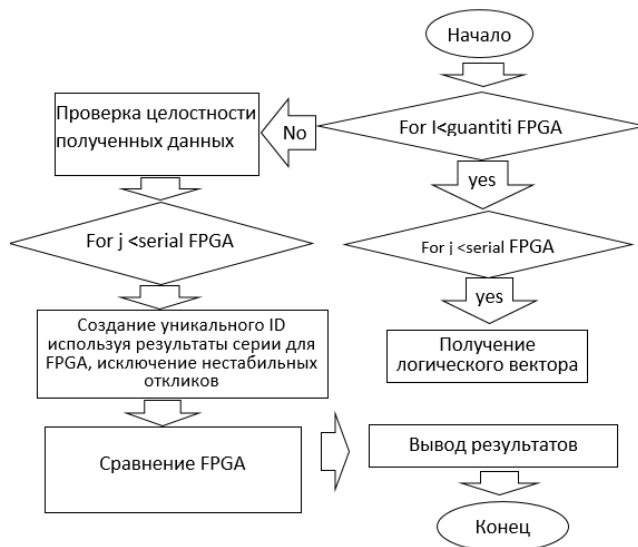


Рис. 7. Блок-схема программы обработки информации.

Листинг кода для графического отображения карты откликов системы:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import pdb
import fileinput
import re

textToSearch = "response"
str_in_string = []
# забираем оклик PUF ТИПА «АРБИТР» в виде str
for line in fileinput.input("response_16len_doublepuf_0.txt"):
    if textToSearch in line:
        find_num = re.findall(r'=(\w+)', line)
        str_in_string.append(find_num[0])
print(len(str_in_string))
#
# превод str в int и нормировка на 0xFFFFFFFF
len_chain_8 = int(256)
# переменная для заполнения массива с длиной 8
len_chain_16 = int(65536)
# переменная для заполнения массива с длиной 16
max_value_logic_vector = int(4294967295)
# переменная для нормировки
num_in_string = []
for i in range(len_chain_16):
    num_in_string.append(int(str_in_string[i], 16))
# перевод в целочисленный вид с указанием начальной
# системы исчисления
num_in_string[i] = num_in_string[i] /
max_value_logic_vector
print(len(num_in_string))
#
# создаем пустой двухмерный массив 16*16
data = np.array(num_in_string).reshape(256,256).tolist()
print(data)
# plotting the heatmap
hm = sns.heatmap(data = data, vmin = 0, vmax = 1)
# displaying the plotted heatmap
```

plt.show()).

Так как отклик системы получен в виде пары «запрос-ответ», то необходимо проверить уникальность полученных пар. На рис.8 представлена гистограмма, иллюстрирующая несовпадения положений векторов, реализующих метастабильные состояния исследуемых ПЛИС с вектором метастабильных состояний ПЛИС с номером D39.

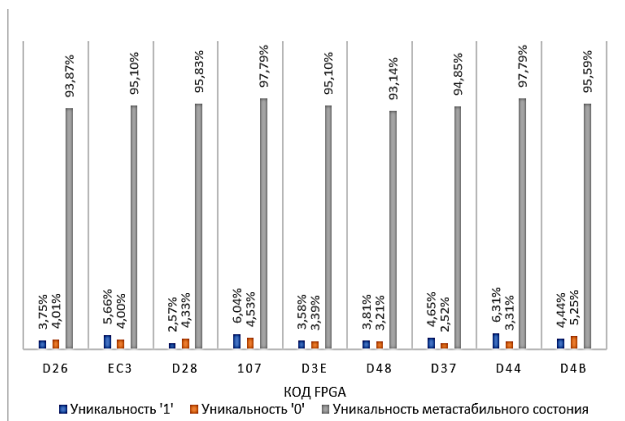


Рис.8. Гистограмма, иллюстрирующая уникальность ФНФ, реализованной в ПЛИС с номером D39 по отношению к другим ПЛИС.

Целью следующих экспериментов является проверка возможности получения информации о PUF типа «арбитр» при имплементации в разные топологические области ПЛИС. Это исследование показывает, насколько эффективно можно получить секретные данные или узнать характеристики PUF путем перемещения физической цепи, включающей PUF, из одной топологической области в другую. Данный анализ позволяет оценить стойкость PUF к физическим атакам и определить, насколько безопасна такая система при перемещении ее компонентов. Результаты, полученные для двух возможных вариантов перемещения: 1) имплементация цепи с использованием одного тактового региона и 2) имплементация в тактовые регионы ПЛИС, показывают, что при имплементации ФНФ типа «арбитр» в соседние секции или блоки логических элементов ПЛИС, невозможно получить информацию о ФНФ.

В рамках данной работы также выполнены опыты по определению влияния на реализацию ФНФ окружающей температуры. Показано, что при изменении температуры корпуса ПЛИС количество положений метастабильного состояния возрастает с увеличением температуры. Результаты иллюстрируются рис.9, на котором сравнение пар запрос-ответ, формирующих метастабильные состояния, обобщены для десяти исследуемых ПЛИС относительно температуры +20°C.

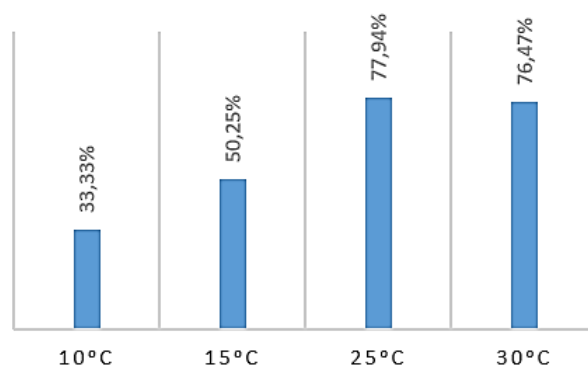


Рис.9. Влияние температуры на ФНФ

Таким образом, при изменении температуры окружающей среды отклик системы нестабилен. Для реализации стабильного отклика ФНФ необходимо поддерживать температурный режим схемы.

VI. ЗАКЛЮЧЕНИЕ

Предложена методика экспериментальных исследований реализации физически неклонированных функций типа «арбитр» в схемах программируемой логики, предназначенных для применения в информационных системах и разработан комплект программных модулей для выполнения экспериментальных исследований.

Методика исследований и соответствующие программы испытаны при исследованиях реализации ФНФ десяти экземпляров семейства Artix-7 (Xilinx). Показано, что:

- 1) Для реализации ФНФ на основе мультиплексоров достаточно сформировать последовательную цепочку из не менее, чем 16 мультиплексоров в блоках SLICE ПЛИС. При этом уникальность ФНФ составляет более 90% для всех исследованных микросхем.
- 2) Метастабильные состояния как индикаторы для идентификации ИС при реализации ФНФ в соседних блоках SLICE воспроизводятся в более 99% испытаний. При использовании других тактовых регионов совпадение составляет 98%.
- 3) Отклик метастабильных состояний в исследуемых ПЛИС зависит от температуры окружающей среды, что обуславливает необходимость фиксации рабочей температуры при реализации ФНФ.

Разработанная методика реализации ФНФ типа «арбитр» предназначена для генерации уникальных идентификационных ключей ПЛИС и, соответственно, может быть применена как одно из технических решений повышения доверенности при проектировании устройств на основе схем программируемой логики.

БЛАГОДАРНОСТИ

Работа выполнена при поддержке Министерства науки и высшего образования РФ (Государственное задание для университетов № ФГФ3-2023-0005).

The work of supported by Ministry of Science and High

Education of RF (State task for the Universities No. FSFZ-2023-0005).

БИБЛИОГРАФИЯ

- [1] C. Herder, M. -D. Yu, F. Koushanfar and S. Devadas, "Physical Unclonable Functions and Applications: A Tutorial," in Proceedings of the IEEE, vol. 102, no. 8, pp. 1126-1141, Aug. 2014, doi: 10.1109/JPROC.2014.2320516.
- [2] Suh, G. & Devadas, Sahana. (2007). Physical Unclonable Functions for Device Authentication and Secret Key Generation. Proc 44th ACM/IEEE Design Automation Conference. 9-14. 10.1109/DAC.2007.375043. DOI: 10.1109/DAC.2007.375043.
- [3] Gao, Y., Al-Sarawi, S.F. & Abbott, D. Physical unclonable functions. Nat Electron 3, 81–91 (2020). <https://doi.org/10.1038/s41928-020-0372-5>.
- [4] Guajardo*, J. (2011). Physical Unclonable Functions (PUFs). In: van Tilborg, H.C.A., Jajodia, S. (eds) Encyclopedia of Cryptography and Security. Springer, Boston, MA. https://doi.org/10.1007/978-1-4419-5906-5_912.
- [5] Ning, H., Farha, F., Ullah, A., & Mao, L. (2020). Physical unclonable function: architectures, applications and challenges for dependable security. IET Circuits, Devices & Systems, 14(4), 407-424. <https://doi.org/10.1049/iet-cds.2019.0175>.
- [6] Physical Unclonable Functions (PUF) for IoT Devices Authors: Abdulaziz Al-Meer, Saif Al-Kuwari ACM Computing Surveys Volume 55 Issue 14 Article No.: 314 pp 1–31 <https://doi.org/10.1145/3591464>
- [7] Bin Tarik, Farhan, Famili, Azadeh, Lao, Yingjie and Ryckman, Judson D.. "Robust optical physical unclonable function using disordered photonic integrated circuits" Nanophotonics, vol. 9, no. 9, 2020, pp. 2817-2828. <https://doi.org/10.1515/nanoph-2020-0049>.
- [8] Cyber Security: Physically Unclonable Functions – YouTube <https://www.youtube.com/watch?v=Gha3Fzh9Y38>
- [9] Jingchang Bian, Zhengfeng Huang, Peng Ye, Zhao Yang and Huaguo Liang A Reliability-Aware Splitting Duty-Cycle Physical Unclonable Function Based on Trade-off Process, Voltage, and Temperature Variations // Journal: ACM Transactions on Design Automation of Electronic Systems, 2023. DOI: 10.1145/3594667. This item cites DOI: 10.1007/978-3-031-19185-5
- [10] Roel MAES Physically Unclonable Functions: Constructions, Properties and Applications. Dissertation presented in partial fulfillment of the requirements for the degree of Doctor. in Engineering // © Katholieke Universiteit Leuven – Faculty of Engineering, Kasteelpark Arenberg 10 box 2446, B-3001 Heverlee (Belgium). D/2012/7515/95. ISBN 978-94-6018-561-8.
- [11] Anandakumar, N. Nalla & Hashmi, Dr. Mohammad & Tehranipoor, Mark. (2021). FPGA-based Physical Unclonable Functions: A comprehensive overview of theory and architectures. Integration. 81. DOI: 10.1016/j.vlsi.2021.06.001.
- [12] Majzoobi, Mehrdad & Koushanfar, Farinaz & Devadas, Sahana. (2011). FPGA PUF using programmable delay lines. 1 - 6. DOI: 10.1109/WIFS.2010.5711471.
- [13] Machida, Takanori & Yamamoto, Dai & Iwamoto, Mitsugu & Sakiyama, Kazuo. (2015). A New Arbiter PUF for Enhancing Unpredictability on FPGA. The Scientific World Journal. 2015. DOI: 10.1155/2015/864812.
- [14] Soybali, Mehmet & Ors, Berna & Saldamli, Gokay. (2011). Implementation of a PUF circuit on a FPGA. 1 - 5. DOI: 10.1109/NTMS.2011.5720638.
- [15] UG835 Vivado Design Suite Tcl Command Reference Guide – 2023. – 1921 с. – Режим доступа: <https://docs.xilinx.com> (дата обращения 20.01.2023).
- [16] UG912 Vivado Design Suite Properties Reference Guide - 2018. – 356 с. – Режим доступа: <https://docs.xilinx.com> (дата обращения 20.09.2023).
- [17] UG903 Vivado Design Suite User Guide Using Constraints – 2021. – 201 с. – Режим доступа: <https://www.xilinx.com> (дата обращения 20.09.2023).

Method for studying the implementation of Physical Unclonable Function in information systems

Vyacheslav Lebedev, Evgeny Pevtsov, Tatiana Demenkova, Mikhail Maletov, Vladimir Filimonov

Abstract - A method has been developed for studying the implementation of physically unclonable functions (FNF) of the “arbiter” type, designed to identify instances of programmable logic integrated circuits included in the hardware of information systems to ensure their trusted design. FNFs of the “arbiter” type are formed as a set of metastable states of the flip-flop at the end of two identical sequential multiplexer circuits, implemented on the basis of SLICE blocks included in the microarchitecture of programmable logic circuits. In this case, the path of the pulse signal from the inputs of the circuits to the trigger is specified by the request vector supplied to the inputs of the multiplexers of both circuits. An indicator of the implementation of the FNF for a given circuit is the values of the response vectors fixed during the implementation of the metastable output of the trigger at the end of the chains of multiplexers with the same specific values of the request vectors. The FNF design methodology and the original program codes for their implementation and research were tested during experiments with programmable logic circuits intended, in particular, for use in information systems. Programs for processing request-response data in the form of a two-dimensional response map have been developed. The dependences of the repeatability of the FNF on the bit depth of the request vector, the location of blocks inside the integrated circuit of programmable logic and the ambient temperature have been studied.

Key words – elements of information systems, programmable logic circuits, physically non-clonable functions, data processing programs.

REFERENCES

- [1] C. Herder, M. -D. Yu, F. Koushanfar and S. Devadas, "Physical Unclonable Functions and Applications: A Tutorial," in Proceedings of the IEEE, vol. 102, no. 8, pp. 1126-1141, Aug. 2014, doi: 10.1109/JPROC.2014.2320516.
- [2] Suh, G. & Devadas, Sahana. (2007). Physical Unclonable Functions for Device Authentication and Secret Key Generation. Proc 44th ACM/IEEE Design Automation Conference. 9-14. 10.1109/DAC.2007.375043. DOI: 10.1109/DAC.2007.375043.
- [3] Gao, Y., Al-Sarawi, S.F. & Abbott, D. Physical unclonable functions. Nat Electron 3, 81–91 (2020). <https://doi.org/10.1038/s41928-020-0372-5>.

- [4] Guajardo*, J. (2011). Physical Unclonable Functions (PUFs). In: van Tilborg, H.C.A., Jajodia, S. (eds) Encyclopedia of Cryptography and Security. Springer, Boston, MA. https://doi.org/10.1007/978-1-4419-5906-5_912.
- [5] Ning, H., Farha, F., Ullah, A., & Mao, L. (2020). Physical unclonable function: architectures, applications and challenges for dependable security. IET Circuits, Devices & Systems, 14(4), 407-424. <https://doi.org/10.1049/iet-cds.2019.0175>.
- [6] Physical Unclonable Functions (PUF) for IoT Devices Authors: Abdulaziz Al-Meer, Saif Al-Kuwari ACM Computing Surveys Volume 55 Issue 14 Article No.: 314pp 1–31 <https://doi.org/10.1145/3591464>
- [7] Bin Tarik, Farhan, Famili, Azadeh, Lao, Yingjie and Ryckman, Judson D.. "Robust optical physical unclonable function using disordered photonic integrated circuits" Nanophotonics, vol. 9, no. 9, 2020, pp. 2817-2828. <https://doi.org/10.1515/nanoph-2020-0049>.
- [8] Cyber Security: Physically Unclonable Functions – YouTube <https://www.youtube.com/watch?v=Gha3Fzh9Y38>
- [9] Jingchang Bian, Zhengfeng Huang, Peng Ye, Zhao Yang and Huaguo Liang A Reliability-Aware Splitting Duty-Cycle Physical Unclonable Function Based on Trade-off Process, Voltage, and Temperature Variations // Journal: ACM Transactions on Design Automation of Electronic Systems, 2023. DOI: 10.1145/3594667. This item cites DOI: 10.1007/978-3-031-19185-5
- [10] Roel MAES Physically Unclonable Functions: Constructions, Properties and Applications. Dissertation presented in partial fulfillment of the requirements for the degree of Doctor. in Engineering // © Katholieke Universiteit Leuven – Faculty of Engineering. Kasteelpark Arenberg 10 box 2446, B-3001 Heverlee (Belgium). D/2012/7515/95. ISBN 978-94-6018-561-8.
- [11] Anandakumar, N. Nalla & Hashmi, Dr. Mohammad & Tehranipoor, Mark. (2021). FPGA-based Physical Unclonable Functions: A comprehensive overview of theory and architectures. Integration. 81. DOI: 10.1016/j.vlsi.2021.06.001.
- [12] Majzoobi, Mehrdad & Koushanfar, Farinaz & Devadas, Sahana. (2011). FPGA PUF using programmable delay lines. 1 - 6. DOI: 10.1109/WIFS.2010.5711471.
- [13] Machida, Takanori & Yamamoto, Dai & Iwamoto, Mitsugu & Sakiyama, Kazuo. (2015). A New Arbiter PUF for Enhancing Unpredictability on FPGA. The Scientific World Journal. 2015. DOI: 10.1155/2015/864812.
- [14] Soybali, Mehmet & Ors, Berna & Saldamli, Gokay. (2011). Implementation of a PUF circuit on a FPGA. 1 - 5. DOI: 10.1109/NTMS.2011.5720638.
- [15] UG835 Vivado Design Suite Tcl Command Reference Guide – 2023. – 1921 c. – URL: <https://docs.xilinx.com>.
- [16] UG912 Vivado Design Suite Properties Reference Guide - 2018. – 356 c. – URL : <https://docs.xilinx.com>.
- [17] UG903 Vivado Design Suite User Guide Using Constraints – 2021. – 201 c. – URL: <https://www.xilinx.com>.