ОБ ОДНОМ КЛАССЕ АЛГОРИТМОВ АНАЛИЗА ПОВЕДЕНИЯ КОМПОНЕНТОВ УСТРОЙСТВ С ПРОГРАММИРУЕМЫМИ ПОЛЬЗОВАТЕЛЕМ ВЕНТИЛЬНЫМИ МАТРИЦАМИ

Титов А.С.[1], Гордеев Э.Н.[2]

Цель исследования: *исследование возможностей повышения защищённости аппаратных средств путём обнаружения участков схемы уровня регистровых передач, находящихся под угрозой нарушения конфиденциальности.*

Метод исследования: *математическое моделирование схемы уровня регистровых передач и применение к модели классических вероятностных алгоритмов проверки свойств булевых функций для обнаружения потенциально уязвимых мест внутренней логики микросхемы.*

Результаты исследования: *на основе построения комбинационных и последовательностных схем, выражающих внутреннюю логику устройств через совокупности булевых функций, построена конкретная модель конвейерной микроархитектуры с разделёнными состояниями и передачей данных, позволяющая проводить исследования путём применения выбранного математического аппарата.*

Выделена модель нарушителя конфиденциальности обрабатываемых специальными видами вычислительных устройств данных. Для конкретной модели конвейерной микроархитектуры и нарушителя конфиденциальности, который может быть расположен на любой стадии производственного процесса, рассмотрена задача минимизации размерности считываемых из схемы данных.

Проведён анализ одного класса алгоритмов в рамках исследуемой модели. По результатам анализа предложены модификации некоторых из них.

Построенные в работе алгоритмы позволяют за счёт выделения индексов аргументов булевых функций уточнять расположение тех входов-выходов смоделированных устройств, которые потенциально уязвимы к нарушению конфиденциальности элементов последовательностных и комбинационных схем.

Научная новизна: *заключена в анализе применимости одного класса вероятностных алгоритмов к задаче обнаружения уязвимых участков устройств, использующих схемную логику, и построении на их основе модификаций с целью улучшения точности определения входов уязвимых элементов.*

Ключевые слова: *программируемая логическая интегральная схема, регистровые передачи, конвейерная микроархитектура, модель нарушителя, булевы функции, схемы из функциональных элементов, существенные переменные.*

DOI:10.21681/2311-3456-2023-5-100-112

1. Введение

Программируемая пользователем вентильная матрица (далее – ППВМ) – интегральная микросхема, внутренняя логика (схема) которой проектируется разработчиком в зависимости от функций конструируемого устройства, используемого ППВМ.

Комбинационная схема (комбинационная логика) – схема на основе функциональных элементов (булевых функций), используемая при математическом моделировании.

Последовательностная схема (последовательностная логика) – схема с определенным видом структуры. (См. ниже точное определение).

Уровень регистровых передач – способ описания комбинационной и последовательностной схемы с ис-

1 Титов Анатолий Сергеевич, студент кафедры ИУ-8 «Информационная безопасность» МГТУ им. Н.Э. Баумана, Москва, Россия. E-mail: toliakpurple@gmail.com

2 Гордеев Эдуард Николаевич, доктор физико-математических наук, профессор кафедры ИУ-8 «Информационная безопасность» МГТУ им. Н.Э. Баумана, Москва, Россия. E-mail: werhorn@yandex.ru

пользованием логических операций, применяемых к данным, которые передаются между элементами схемы (см., например, [1]).

Компонент – это описанная на уровне регистровых передач составная часть устройства, которая выполняет заданную его разработчиком функцию.

Аппаратным трояном (аппаратной недекларированной возможностью) называется внедрённая злоумышленником возможность, которая влияет на информационную безопасность устройства. Участок схемы, который может быть использован злоумышленником для размещения аппаратного трояна, принято называть *уязвимым участком* [2].

Области применения ППВМ: анализ сетевых данных[3], цифровая обработка аналоговых сигналов[4], построение роботизированных систем, ускорение вычислений [3–5] и другие. В большинстве случаев актуальна необходимость обеспечения информационной безопасности и вследствие этого исследование угроз и анализа возможных атак злоумышленников на устройства с ППВМ [6].

К целям злоумышленника относятся, например, внедрение аппаратных троянов [7] и другие действия по нарушению конфиденциальности или целостности обрабатываемой информации[5].

Для исследования поведения компонентов устройств с ППВМ и локализации аппаратных троянов авторы работы [8] строят модель передачи данных внутри схемы уровня регистровых передач.

В [9–12] предложены методы анализа информационной безопасности таких устройств с использованием машинного обучения и нейронных сетей.

В работах [13, 14] рассмотрен инструментарий для автоматизированного выявления уязвимых мест схемы.

В статьях [15, 16] описана методика выделения уязвимых участков схемы уровня регистровых передач на основе, в частности, математического моделирования устройства с использованием систем булевых функций.

При анализе свойств и способов конструкции ППВМ с точки зрения информационной безопасности рассматривается несколько разных моделей нарушителей. В настоящей работе нарушителем считается участвующий в конструировании устройства с ППВМ разработчик, у которого есть доступ к описанию внутренней логики на уровне регистровых передач.

Его целью является закладка в конструкцию возможностей раскрытия конфиденциальных данных, которые обрабатываются внутренней логикой устройства. Для этого нарушитель может, например, внедрить в устройство компонент, считывающий, сохраняющий и передающий обрабатываемые этим устройством данные.

Сокрытие такого компонента обеспечивается уменьшением его физических размеров и потребляемых ресурсов (например, электропитания).

Одним из факторов, способствующих снижению использования ресурсов компонента, является уменьшение количества считываемых из устройства сигналов, а, следовательно, обеспечение возможности для хранения и дальнейшей передачи несанкционированной к распространению информации.

Для анализа наличия признаков уменьшения количества считываемых сигналов используется математическая модель устройства. В частности, для моделирования последовательностной логики применяются конечные автоматы Мура и Мили[6]. Конечные автоматы здесь используются для описания *последовательностной логики* с целью дальнейшего преобразования этого описания в схему моделирующую устройство.

Так как в настоящий работе рассматривается анализ уже существующей схемы, то он должен базироваться на свойствах самой ППВМ. А в этом случае конечные автоматы мы не применяем.

Мы будем использовать математическую модель устройства на базе систем булевых функций, а затем к ее анализу нами применён вероятностный алгоритм проверки свойств булевых функций. Подобные исследования осуществлены, в частности, в работах [17, 18].

В статьях [19, 20] рассматривается алгоритм проверки существенной зависимости булевой функции от не более, чем k переменных. Поводом для этого является тот факт, что сигналы, соответствующие фиктивным аргументам математической модели, могут быть исключены из считывания, уменьшив таким образом необходимые ресурсы компонента для хранения и передачи данных и высвободив их для других целей.

Отметим, что приведённые в этих работах вероятностные алгоритмы возвращают результат в форме распознавания («да» или «нет») без указания конкрет-

3 Trimberger S. Three Ages of FPGAs: A Retrospective on the First Thirty Years of FPGA Technology. Proceedings of the IEEE, vol. 103, no. 3, pp. 318-331, 2015. DOI: 10.1109/JPROC.2015.2392104

4 Romoth J., Porrmann M. Survey of FPGA applications in the period 2000 – 2015. 2017. DOI: 10.13140/RG.2.2.16364.56960

5 Li H., Liu Q., Zhang J. A survey of hardware Trojan threat and defense. // Integr. VLSI J., 2016

6 Pedroni V. Finite State Machines in Hardware: Theory and Design (with VHDL and SystemVerilog). The MIT Press. 2013. DOI: 10.7551/mitpress/9657.001.0001

ных существенных аргументов. В ряде случаев знание этих аргументов повышает эффективность методов обнаружения возможных закладок злоумышленника, о которых говорилось выше.

Статья состоит из введения и трех разделов. В разделе 2 описана модель комбинационной и последовательностной логики, построена модель конвейерной микроархитектуры и поставлена задача минимизации размерности считываемых из схемы данных.

Раздел 3 посвящён решению этой задачи. Для этого используемые ранее алгоритмы проверки существенной зависимости булевой функции от не более, чем k переменных, модифицированы так, чтобы возвращаемым значением было множество индексов существенных аргументов.

В разделе 4 рассмотрена сравнительная характеристика исходных и модифицированных алгоритмов.

Полученные здесь алгоритмы могут быть использованы для определения исключаемых или нарушаемых злоумышленником сигналов.

С точки зрения защиты информации, применение такого анализа предоставит данные для определения уязвимых участков схемы. К ним могут быть отнесены те участки, например, где количество считываемых сигналов отличается от того, которое может быть получено на базе проведенного анализа.

2. Математическая модель комбинационной и последовательностной логики

Определение 1. *Моделью комбинационной логики* названа тройка $\langle f, n, m\rangle$, где: $f\colon \{0,1\}^n \to \{0,1\}^m$ – система булевых функций.

Пусть $f_i\colon\{0,1\}^n \to \{0,1\}$, $i = \overline{1,m}$. Тогда элемент комбинационной логики задаётся вектор-функцией $f(\vec{X}) = \left(f_1(\vec{X}), \ldots, f_m(\vec{X})\right)$.

При описании *последовательностной логики* обычно используется понятие тактирования – пошагового задания входных и выходных данных. Вектор данных $\vec{X}$ такта (шага) $\mathbf{t}$ обозначается как $\vec{X}^t$.

Определение 2. *Моделью последовательностной логики* названа четвёрка $\langle g, I, n, m\rangle$, где g – отображение $\{0,1\}^{n+k} \to \{0,1\}^{m+k}$, а I – множество (мощности k) индексов аргументов, задающих внутреннее состояние последовательностной схемы.

На вход отображения g подаётся вектор $\vec{\tilde{X}}^t = \left(X_1^t, \ldots, X_n^t, S_1^t, \ldots, S_k^t\right)$, содержащий значения входов $X_1^t, \ldots, X_n^t$ и состояния $S_1^t, \ldots, S_k^t$ схемы. Выходом отображения является (1) конкатенация вектора выходных значений $\left(Y_1^{t+1}, \ldots, Y_m^{t+1}\right)$ и вектора состояния на следующем такте $\left(S_1^{t+1}, \ldots, S_k^{t+1}\right)$:

$$g\left(X_1^t, \ldots, X_n^t, S_1^t, \ldots, S_k^t\right) = \\ = \left(Y_1^{t+1}, \ldots, Y_m^{t+1}, S_1^{t+1}, \ldots, S_k^{t+1}\right) \quad (1)$$

Отображение g модели $\langle g, I, n, m\rangle$ представимо в виде (2) совокупности булевых функций преобразования данных $g_i\colon\{0,1\}^{n+k} \to \{0,1\}$ и функций перехода состояний $s_j\colon\{0,1\}^{n+k} \to \{0,1\}$:

$$\begin{cases} g_1\left(X_1^t, \ldots, X_n^t, S_1^t, \ldots, S_k^t\right) = Y_1^{t+1} \\ \ldots \\ g_m\left(X_1^t, \ldots, X_n^t, S_1^t, \ldots, S_k^t\right) = Y_m^{t+1} \\ s_1\left(X_1^t, \ldots, X_n^t, S_1^t, \ldots, S_k^t\right) = S_1^{t+1} \\ \ldots \\ s_k\left(X_1^t, \ldots, X_n^t, S_1^t, \ldots, S_k^t\right) = S_k^{t+1} \end{cases} \quad (2)$$

Приведенное ниже обозначение $\vec{X}^{(\pi)}$ указывает на применение подстановки π к индексам значений $\vec{X}$: $\vec{X}^{(\pi)} = \left(X_{\pi(1)}, \ldots, X_{\pi(n)}\right)$.

Определение 3. Отображение $\hat{g}\colon\{0,1\}^{n+k} \to \{0,1\}^{m+k}$ называется эквивалентным описанием для модели $\langle g, I, n, m\rangle$ тогда и только тогда, когда существует подстановка π_X на множестве индексов аргументов функции; и подстановка π_Y на множестве индексов вектора выходных данных, для которой верно (3):

$$\forall \vec{X} \in \{0,1\}^n, \vec{S} \in \{0,1\}^k: \\ \left[\hat{g}\left(\vec{X}^{(\pi_X)}, \vec{S}^{(\pi_X)}\right)\right]^{(\pi_Y)} = g\left(\vec{X}, \vec{S}\right) \quad (3)$$

Определение 4. Модель $\langle g, I, n, m\rangle$ эквивалентна модели $\langle \hat{g}, \hat{I}, \hat{n}, \hat{m}\rangle$ при выполнении следующих условий:

1) $\hat{n} = n$;

2) $\hat{m} = m$;

3) существует такая подстановка σ на множестве состояний $\hat{S} \subseteq \{0,1\}^k$, что $\hat{g}$ является *эквивалентным* (с подстановкой π_X и π_Y) *описанием* элемента с состояниями $\sigma\left(\vec{S}\right)$, и $\hat{I} = I^{(\pi_X)}$.

Пусть задано множество $M \subseteq \{1,n\}$ и $\overline{M} = \{1,n\} \setminus M$.

Определение 5. Модель $\langle \hat{g}, \hat{K}, \hat{n}, \hat{m}\rangle$ называется составной частью модели $\langle g, K, n, m\rangle$ тогда и только тогда, когда существует $J_P = \{j_1, \ldots, j_{\hat{n}}, s_1, \ldots, s_{|\hat{K}|}\} \subseteq \{1, \ldots, n + |K|\}$ и суще-

ствует $I_P = \{i_1, \ldots, i_{\hat{m}}\} \subseteq \{1, \ldots, m + |K|\}$ такое, что для любого входного $\vec{X} \in \{0,1\}^{\hat{n}+|\hat{K}|}$ и для любого $\vec{\bar{X}} \in \{0,1\}^{n+|K|-\hat{n}-|\hat{K}|}$ выполнено: $\hat{\vec{g}}\left(\vec{X}\right) = \left(Y_{i_1}, \ldots, Y_{i_{\hat{m}}}\right)$, где $\vec{Y} = \left(Y_{i_1}, \ldots, Y_{i_m}\right) = g(\vec{X}_{(J_P)}, \vec{\bar{X}}_{(\bar{J}_P)})$.

Для краткости здесь введено обозначение: $\vec{X}_{(J_P)} = \left(X_{J_{P_1}}, \ldots, X_{J_{P_n}}\right)$.

Теперь рассмотрим частный случай *последовательностной логики – конвейерную микроархитектуру* (*конвейеризацию*), на основе которой ниже будет формализована задача минимизации считываемых злоумышленником данных.

Конвейеризация (см., например, [21]) – это временной параллелизм, предполагающий разбиение составной операции таким образом, чтобы в процессе выполнения одной операции начинали выполняться следующие, согласно приведённой ниже схеме:

Вектор $\vec{Z}$ назовём *вектором передаваемых данных*. Его смысл проиллюстрирован на следующем рисунке:

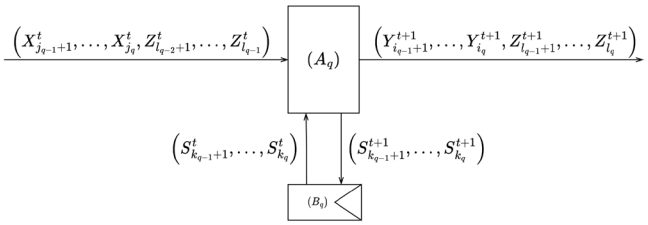

Рис. 2. Модель последовательностной логики с выделенным для передачи данных вектором $\vec{Z}$

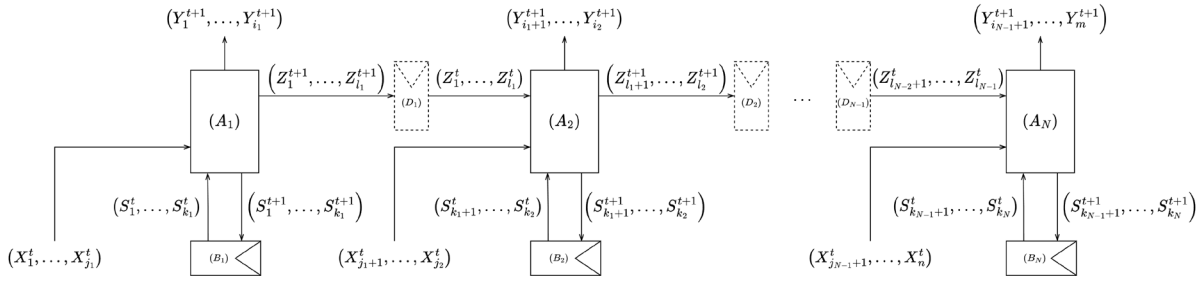

Рис. 1. Модель конвейерной микроархитектуры

Формально это определяется следующим образом.

Определение 6. *Моделью конвейерной микроархитектуры* (рис.1) это такая *последовательностная модель* $\langle g, I, n, m \rangle$, что в ней:

— вектор состояния $\vec{S} \in \{0,1\}^{|I|}$ имеет вид:

$$\left(S_1, \ldots, S_{\sum_{q=1}^{N}|I_q|}, Z_1, \ldots, Z_{\sum_{q=1}^{N-1}\alpha_q}\right);$$

— функция $g\left(\vec{X^t}, \vec{S^t}\right) = \left(\vec{Y^{t+1}}, \vec{S^{t+1}}\right)$ задана так, как показано ниже (4):

$$g\left(\vec{X^t}, \vec{S^t}\right) = \begin{bmatrix} g_1\begin{pmatrix} X^t_1, \ldots, X^t_{j_1}, \\ S^t_1, \ldots, S^t_{k_1} \end{pmatrix}, \\ \cdots \\ g_q\begin{pmatrix} X^t_{j_{q-1}+1}, \ldots, X^t_{j_q}, \\ Z^t_{l_{q-2}+1}, \ldots, Z^t_{l_{q-1}}, \\ S^t_{k_{q-1}+1}, \ldots, S^t_{k_q} \end{pmatrix}, \\ \cdots \\ g_N\begin{pmatrix} X^t_{j_{N-1}+1}, \ldots, X^t_{j_N}, \\ Z^t_{l_{N-2}+1}, \ldots, Z^t_{l_{N-1}}, \\ S^t_{k_{N-1}+1}, \ldots, S^t_{k_N} \end{pmatrix} \end{bmatrix} \quad (4)$$

Определение 7. *Вспомогательным элементом* (рис.2) конвейерной микроархитектуры является модель $g_q, I_q, n_q + \alpha_{q-1}, m_q + \alpha_q$. Где α_{q-1} – размерность вектора входной передачи данных, а $\pm_q$ – выходной. Остальные параметры обладают следующими свойствами: $l_q - l_{q-1} = \alpha_q$, $k_q - k_{q-1} = |I_q|$, $j_q - j_{q-1} = n_q$ и $i_q - i_{q-1} = m_q$. А отображение g_q имеет вид:

$$g_q\begin{pmatrix} X^t_{j_{q-1}+1}, \ldots, X^t_{j_q}, \\ Z^t_{l_{q-2}+1}, \ldots, Z^t_{l_{q-1}}, \\ S^t_{k_{q-1}+1}, \ldots, S^t_{k_q} \end{pmatrix} = \begin{pmatrix} Y^{t+1}_{i_{q-1}+1}, \ldots, Y^{t+1}_{i_q}, \\ Z^{t+1}_{l_{q-1}+1}, \ldots, Z^{t+1}_{l_q}, \\ S^{t+1}_{k_{q-1}+1}, \ldots, S^{t+1}_{k_q} \end{pmatrix} \quad (5)$$

При этом, для задания элемента A_1 полагаем $\alpha_0 = 0$ (так как у A_1 отсутствует входной вектор передаваемых данных), а для элемента A_N полагаем $\alpha_N = 0$ (так как для него отсутствует выходной вектор передаваемых данных).

Очевидно, что модель конвейерной микроархитектуры и её вспомогательные элементы обладают следующими свойствами (6):

$$\begin{cases} |I| = \sum_{q=1}^{N} |I_q| + \sum_{q=1}^{N-1} \alpha_q \\ n = \sum_{q=1}^{N} n_q \\ m = \sum_{q=1}^{N} m_q \end{cases} \quad (6)$$

Теперь мы подошли к возможности формализовать задачу, о которой шла речь во введении.

Задача минимизации собираемых злоумышленником данных. **(Задача 1)**.

Пусть задана модель конвейерной микроархитектуры g, I, n, m, составными частями которой являются g_q, I_q, n_q, m_q. На каждом такте вредоносный компонент E считывает векторы X^t, S^t, Y^t (рис.3).

Пусть $E^t = \left[X^t, S^t, Y^t \right]$, тогда считанные за T тактов данные обозначим матрицей E (7). Размерность $\vec{E}$ равна $e = n + m + |I|$, а размерность матрицы E – $\mathrm{T} \times \mathrm{e} = T \times \left(n + m + |I| \right)$.

$$E = \begin{bmatrix} \vec{E_1} \\ \dots \\ \vec{E_T} \end{bmatrix} = \begin{bmatrix} \vec{X^0} & \vec{0} & 0 \\ \vec{X^1} & \vec{S^1} & \vec{Y^1} \\ \dots & \dots & \dots \\ \vec{X^{T-1}} & \vec{S^{T-1}} & \vec{Y^{T-1}} \end{bmatrix} \quad (7)$$

Как было отмечено ранее, для уменьшения потребляемых внедрённым компонентом ресурсов необходимо минимизировать считываемые данные (матрицу E). Следовательно, необходимо снизить размерность вектора $\vec{X}$ и $\vec{Z}$, а для этого нужно выделить существенные переменные булевых функций, образующих g_q.

Чтобы проиллюстрировать поведение составных элементов конвейерной микроархитектуры (рис.3), на рис.4 показана диаграмма переходов векторов входных, выходных, передаваемых данных и состояний между составными частями $g_q, I_q, n_q + \alpha_{q-1}, m_q + \alpha_q$ модели g, I, n, m.

Работа, согласно этой схеме, выглядит следующим образом. Допустим, некоторые данные поступили в A_1 на шаге $\mathrm{t} = 1$. Обработанные данные будут записаны в A_2 на шаге $t = 2$, при этом в A_1 поступят новые данные. В результате, на шаге $t = N$ вектор $\left(Y^N_{i_{N-1}}, \dots, Y^N_m \right)$ будет содержать прошедшие N шагов обработки данные $\left(X^1_1, \dots, X^1_{j_1} \right)$. При этом на шаге $t = N$ каждый элемент A_q содержит данные, прошедшие q шагов обработки.

Таким образом осуществляется *временной параллелизм* с разбиением одной составной операции на несколько.

3. Применение вероятностных алгоритмов проверки свойств булевых функций

Для решения *Задачи 1* мы применим классические вероятностные алгоритмы проверки свойств булевых функций. А именно алгоритмы проверки существенной зависимости булевой функции от не более, чем k переменных.

Пусть: $[n] = \{1, \dots, n\}$ и $\bar{S} = [n] \setminus S$.

Определение 8. *Свойством* $\mathcal{P}$ *булевой функции* от n переменных называется множество $\mathcal{P} = \left\{ g : \{0,1\}^n \to \{0,1\} \right\}$.

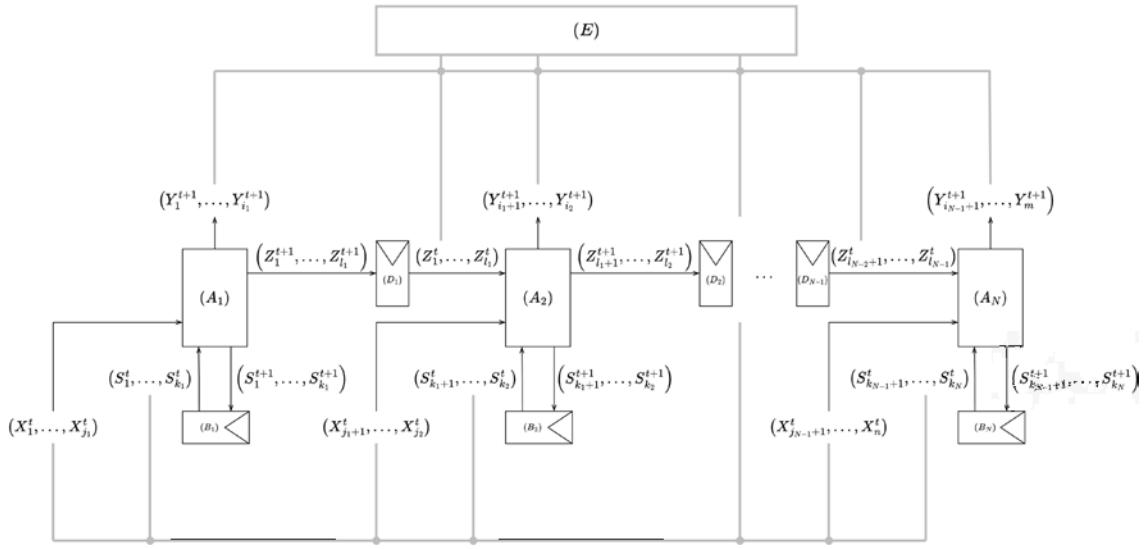

Рис. 3. Модель конвейерной микроархитектуры с внедрённым злоумышленником архитектурным компонентом

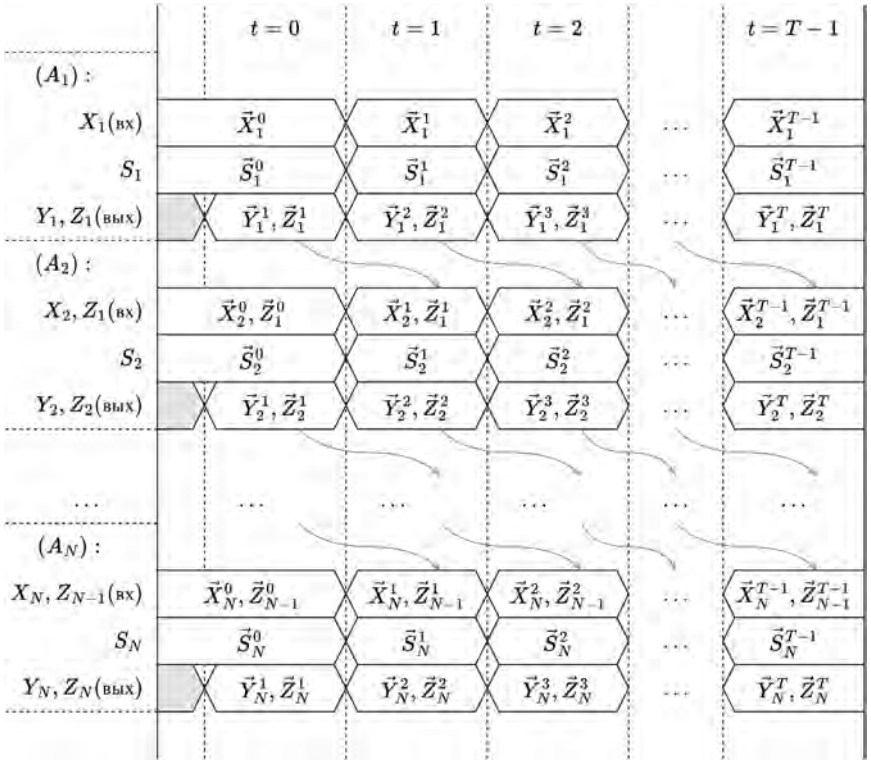

Рис. 4. Диаграмма потока данных модели конвейерной микроархитектуры

Определение 9. *Расхождением булевой функции* f *от булевой функции* g называется такое $\rho(f,g)$, что (8):

$$\rho(f,g) = P_{\forall x\in\{0,1\}^n}\left(f(x)\neq g(x)\right) = \frac{\left|\left\{f(x)\neq g(x) \;:\; x\in\{0,1\}^n\right\}\right|}{2^n} \quad (8)$$

Определение 10. *Расхождением булевой функции* f *от свойства* $\mathcal{P}$ называется такое $\rho(f,\mathcal{P})$, что (9):

$$\rho(f,\mathcal{P}) = \min_{g\in\mathcal{P}} \rho(f,g) \quad (9)$$

Вероятностный алгоритм проверки свойства $\mathcal{P}$ *булевой функции* f – это алгоритм, возвращающий ответ в форме распознавания, с параметрами:

- s – мощность множества S входных данных функции f со случайным распределением D;
- q – количество входных векторов, значение функции на которых необходимо проверить;
- μ (точность) – максимальное расстояние булевой функции от свойства.

Такой алгоритм проверки свойства $\mathcal{P}$ *«принимает»* булеву функцию f с вероятностью не менее $\frac{2}{3}$ тогда и только тогда, когда $f\in\mathcal{P}$. И *«отвергает»* булеву функцию f с вероятностью не менее $\frac{2}{3}$ тогда и только тогда, когда $\rho(f,\mathcal{P})\geq\varepsilon$. То есть достоверность результата выполнения алгоритма составляет не менее $\frac{2}{3}$.

Пусть f – булева функция, которая вычисляется в процессе работы алгоритма.

В качестве *меры сложности* используется функция, зависящая от длины входа алгоритма и результатом которой является количество вычислений заданной выше функции f.

Используется подход, который, в частности, применялся в работах [22, 23].

Определение 11. *Избыточное множество индексов* существенных переменных I – такое множество $I\subseteq[n]$, для которого верно, что $J\subseteq I$, где J – множество индексов существенных аргументов.

То есть избыточное множество индексов содержит индексы существенных переменных, однако может содержать, в том числе, индексы фиктивных.

Определение 12. Множество индексов существенных переменных I *с недостатком* – такое множество $I\subseteq[n]$, для которого верно, что $I\subseteq J$, где J – множество индексов существенных аргументов.

То есть, I не содержит индексы фиктивных переменных, однако, может содержать не все индексы существенных переменных.

Теперь в рамках сформулированной в предыдущем разделе *Задачи 1* необходимо для каждой составной части g_q, I_q, n_q, m_q выделить количество существенных аргументов булевых функций g_{q_i}, образующих g_q, и определить индексы этих аргументов.

Для определения количества существенных переменных используется следующий подход.

По заданной функции g_{q_i} выбирается минимальное по размеру подмножество индексов аргументов J так, чтобы существовала существенно зависимая от $|J|$ переменных функция h, для которой выполнено: $\rho\left(g_{q_i}, h\right) < \varepsilon$.

Для определения индексов аргументов в нашем случае нельзя использовать общепринятую формулировку вероятностного алгоритма проверки существенной зависимости функции не более чем от k переменных.

Это связано с тем, что алгоритмы, например, построенные в [22, 23] возвращают *вероятностный ответ* в форме распознавания без указания конкретного множества индексов существенных аргументов.

Поэтому предложены модификации алгоритмов из работы [22] (обозначен как А1) и [23] (обозначен как А2) с целью обеспечения этого свойства.

Опишем ниже последовательность действий алгоритма А1 с сопровождающей его процедурой П1.

Входными данными процедуры П1 является функция f и множество $S \subseteq [n]$.

Процедура П1 состоит из следующих шагов.

а) Случайно задаются два вектора $\vec{x}, \vec{y} \in \{0,1\}^n$. Распределение равномерное.

б) Формируется вектор $\vec{x} = \left(x_{\bar{S}}, y_S\right)$; если $f\left(\vec{x}\right) = f\left(\vec{x}\right)$, то результат выполнения процедуры положительный. Иначе – процедура выполнена отрицательно.

Теперь приведём шаги алгоритма А1.

а) Произвольно разбивается множество индексов аргументов $[n]$ на $\mathrm{O}\left(\mathrm{k}^2\right)$ подмножеств S_i.

б) Для всех $\mathrm{i}: O\left(k^2/\varepsilon\right)$ раз выполняется процедура П1 для f и S_i.

в) Если для не более k подмножеств S_i в предыдущем пункте получен отрицательный результат, то заданная f существенно зависит не более чем от k переменных.

Далее описан алгоритм А2 с используемой процедурой П2.

Входными данными процедуры П2 являются:

— булева функция f от n переменных;

— анализируемый вектор $\vec{x}$;

— множество S анализируемых индексов переменных;

— множества $S_1, \ldots, S_s$, которые являются разбиением $[n]$.

Процедура П2 состоит из следующих шагов.

а) Если $|S| = 1$, то процедура возвращает тот S_j, которому принадлежит индекс из S.

б) Выполняется разбиение S на два множества $\widehat{S_1}$ и $\widehat{S_2}$. Их мощности, соответственно, равны $\left\lfloor \frac{|S|}{2} \right\rfloor$ и $\left\lceil \frac{|S|}{2} \right\rceil$.

в) Составляется вектор $\vec{y} = \vec{x}$ с инверсией битов множества $\widehat{S_2}$.

г) Если $f\left(\vec{x}\right) \neq f\left(\vec{y}\right)$, то выполняется процедура П2 от f, $\vec{x}$, $\widehat{S_2}$ и разбиения $S_1, \ldots, S_s$.

д) Иначе, выполняется процедура П2 от f, $\vec{z}$, $\widehat{S_1}$ и разбиения $S_1, \ldots, S_s$. За вектор $\vec{z}$ принят $\vec{x}$ с инверсией битов множества S.

Шаги алгоритма А2 приведены ниже.

а) Инициализируются переменные $S \leftarrow [n]$ и $\psi \leftarrow 0$.

б) Индексы $[n]$ случайно разбиваются на s подмножеств: $S_1, \ldots, S_s$.

1) Выполняется цикл из r итераций, в котором создаётся случайная пара $(\vec{x}, \vec{y}) \in \{0,1\}^n \times \{0,1\}^n$ и составляется вектор $\vec{x} = \left(x_{\bar{S}}, y_S\right)$.

2) Проверяется $f\left(\vec{x}\right) = f\left(\vec{x}\right)$ – если верно, то выполняется переход к следующей итерации цикла; если ложно – то к следующему пункту.

3) Выполняется процедура П2 от f, $\vec{x}$, $\left\{i : x_i \neq \widehat{x_i}\right\}$ (множества анализируемых индексов) и разбиения $S_1, \ldots, S_s$. Результат процедуры – I_j.

4) Обновляется переменная $S \leftarrow S \setminus I_j$ и $\psi \leftarrow \psi + 1$.

5) Если счётчик E превысил значение k, то алгоритм «отвергает» f.

в) В случае успешного прохождения цикла алгоритм «принимает» f как функцию, существенно зависящую не более чем от k переменных. Иначе – «отвергает».

Рассмотрим модифицированный алгоритм А1*, который включает изменённую процедуру П1*.

Входными данными процедуры П1* является функция f и множество $S \subseteq [n]$. Процедура П1* состоит из следующих шагов.

а) Случайно задаются два вектора $\vec{x}, \vec{y} \in \{0,1\}^n$. Распределение равномерное.

б) Формируется вектор $\vec{\tilde{x}} = \left(x_{\bar{S}}, y_S\right)$; если $f\left(\vec{x}\right) = f\left(\vec{\tilde{x}}\right)$, то процедура выполнена успешно.

в) (Модификация) в случае отрицательного результата, полученного в пункте (б), процедура возвращает отказ и множество индексов $\left\{i : \ x_i \neq \widehat{x_i}\right\}$.

Шаги алгоритма А1*:

а) Произвольно разбивается множество индексов аргументов $[n]$ на $O(k^2)$ подмножеств S_i.

б) Для всех $i: O(k^2/\varepsilon)$ раз выполняется процедура П1* для f и S_i.

в) Если для не более k подмножеств S_i в предыдущем пункте получен отрицательный результат, то заданная f существенно зависит не более чем от k переменных.

г) (Модификация) если функция f «принята», то возвращается множество $I = \bigcup_i I_i$ – объединение результатов процедуры П1*, которые были не пройдены для S_i.

Таким образом, алгоритм А1* возвращает множество индексов аргументов функции f в случае её «принятия».

Ниже рассмотрена модификация алгоритма А2 – А2.1*. Она включает процедуру П2.1*, набор входных данных и действий которой аналогичен процедуре П2.

Шаги модифицированного алгоритма А2.1* приведены ниже.

а) Инициализируются переменные $S \leftarrow [n]$ и $\psi \leftarrow 0$.

б) Индексы $[n]$ случайно разбиваются на s подмножеств: $S_1, \ldots, S_s$.

1) Выполняется цикл из r итераций, в котором создаётся случайная пара $(\vec{x}, \vec{y}) \in \{0,1\}^n \times \{0,1\}^n$ и составляется вектор $\vec{\hat{x}} = (x_{\bar{S}}, y_S)$.

2) Проверяется $f(\vec{x}) = f(\vec{\hat{x}})$ – если верно, то выполняется переход к следующей итерации цикла; если ложно – то к следующему пункту.

3) Выполняется процедура П2.1* от f, $\vec{x}$, $\{i: x_i \neq \hat{x}_i\}$ (множества анализируемых индексов) и разбиения $S_1, \ldots, S_s$. Результат процедуры – I_j.

4) Обновляется переменная $S \leftarrow S \setminus I_j$ и $\psi \leftarrow \psi + 1$.

5) Если счётчик ψ превысил значение k, то алгоритм «отвергает» f.

в) В случае успешного прохождения цикла алгоритм «принимает» f как функцию, существенно зависящую не более чем от k переменных.

г) (Модификация 1) для «принятой» функции алгоритм возвращает множество $[n] \setminus S$.

Вторая модификация алгоритма А2 (обозначим её А2.2*) состоит в следующем.

В А2.2* в пункт (а) процедуры П2.1* добавлен возврат индекса из множества S. Пункт (в.4) алгоритма А2.1* дополнен сохранением результата процедуры во множество I (которое инициализируется пустым множеством). В пункте (г) добавлен возврат полученного множества I.

С применённой модификацией алгоритм А2.1* вернёт избыточное множество индексов существенных переменных функции f, а алгоритм А2.2* – множество индексов с недостатком.

В табл.1 приведены характеристики рассмотренных в данном разделе алгоритмов.

Колонка «*Выделение множества*» указывает, реализована ли алгоритмом выдача множества индексов существенных переменных. В колонке «Множество индексов» отмечена характеристика множества индексов. «Множество индексов» может быть указано только для тех алгоритмов, которые реализуют «Выделение множества».

Таблица 1

Характеристики рассмотренных алгоритмов

Алгоритм	Выделение множества	Множество индексов
А1	Нет	
А2	Нет	
А1*	Да	С избытком
А2.1*	Да	С избытком
А2.2*	Да	С недостатком

4. Анализ результатов работы модифицированных алгоритмов

Для проведения численного эксперимента модифицированные алгоритмы была реализованы программно. Установлены следующие параметры:

- количество аргументов булевой функции – $n = 14$;
- множество аргументов $[n]$ разбито на $s = n/2$ подмножеств;
- количество проверяемых существенных переменных – $k = 10$;
- булевых функций – 128;
- если в алгоритме есть цикл, то он выполняется 1000 итераций.

Эксперимент состоит из следующих шагов.

а) Произвольно создаётся 128 булевых функций от 14 переменных со случайным выбором фиктивных и существенных аргументов;

б) Для каждой функции каждый алгоритм испытывается 1000 раз;

в) Количество обращений к булевой функции и полученные результаты группируются по количеству существенных аргументов созданных в (а) функций и усредняются.

Пусть k – проверяемое количество существенных переменных.

Сложность алгоритма А1* совпадает со сложностью А1 и равна: $O(k^4 \cdot \log(k+1)/\varepsilon)$.

Таблица 2

Сравнительная характеристика сложностей модифицированных алгоритмов

Алгоритм	Количество существенных аргументов	Обращений к функции	Аналитическая Сложность
A1*	0	14000	$O\left(k^4 \cdot \log(k+1)/\varepsilon\right)$
	2	10299	
	4	7038	
	6	3839	
	8	2574	
	10	1043	
	12	404	
	14	38	
A2.1*	0	2000	$O\left(k \cdot \log k + k/\varepsilon\right)$
	2	2019	
	4	2031	
	6	2041	
	8	1910	
	10	1057	
	12	429	
	14	71	
A2.2*	0	2000	$O\left(k \cdot \log n + k/\varepsilon\right)$
	2	2030	
	4	2052	
	6	2071	
	8	1945	
	10	1096	
	12	470	
	14	112	

Сложность алгоритма A2.1*: $O\left(k \log k + k/\varepsilon\right)$.

Сложность алгоритма A2.2* равна

$$O\left(2r + r \cdot \log|S|\right) = O\left(\frac{k}{\varepsilon} + k \cdot \log n\right). \quad (10)$$

В результате проведённого анализа получена сравнительная характеристика сложностей модифицированных алгоритмов (табл.2) и результатов их работы (табл.3). Графическое представление зависимости перечисленных характеристик от количества существенных аргументов произвольной булевой функции показано на рис.5.

Выполненный анализ показывает, что алгоритм A1* уступает A2.1* и A2.2* по количеству обращений к булевой функции.

По количеству верно и ложно определённых индексов существенных переменных он расположен между алгоритмами A2.1* и A2.2*.

Алгоритм A2.1* имеет наибольший показатель ошибочно выделенных индексов. При этом он обладает самым низким, по сравнению с A1* и A2.2*, количеством пропущенных существенных аргументов.

Результат выполнения алгоритма A2.2* характеризуется отсутствием ошибочно выделенных аргументов. С другой стороны, показатель пропущенных существенных индексов аргументов превышает показатели других рассмотренных в рамках сравнительной характеристики алгоритмов.

На практике можно использовать аналитическую программу на основе линейной комбинации приведенных алгоритмов. Можно также построить адаптивную процедуру, которая подстраивается под свойства анализируемого устройства.

5. Заключение

В настоящей работе исследованы возможности повышения степени защиты аппаратных средств путём определения участков схемы, находящихся под угрозой нарушения конфиденциальности.

Таблица 3

Сравнительная характеристика работы модифицированных алгоритмов

Алгоритм	Количество существенных аргументов	Выделено верно	Выделено ошибочно	Не выделено
A1*	0	0.00	0.00	0.00
	2	1.91	0.86	0.09
	4	3.63	1.49	0.37
	6	5.38	2.09	0.62
	8	6.50	1.72	1.50
	10	7.68	1.50	2.32
	12	8.53	0.82	3.47
	14	9.32	0.00	4.68
A2.1*	0	0.00	0.00	0.00
	2	2.00	1.71	0.00
	4	3.80	3.00	0.20
	6	6.00	4.20	0.00
	8	8.00	3.47	0.00
	10	10.00	3.00	0.00
	12	12.00	1.64	0.00
	14	14.00	0.00	0.00
A2.2*	0	0.00	0.00	0.00
	2	1.86	0.00	0.14
	4	3.40	0.00	0.60
	6	5.10	0.00	0.90
	8	5.73	0.00	2.27
	10	6.50	0.00	3.50
	12	6.82	0.00	5.18
	14	7.00	0.00	7.00

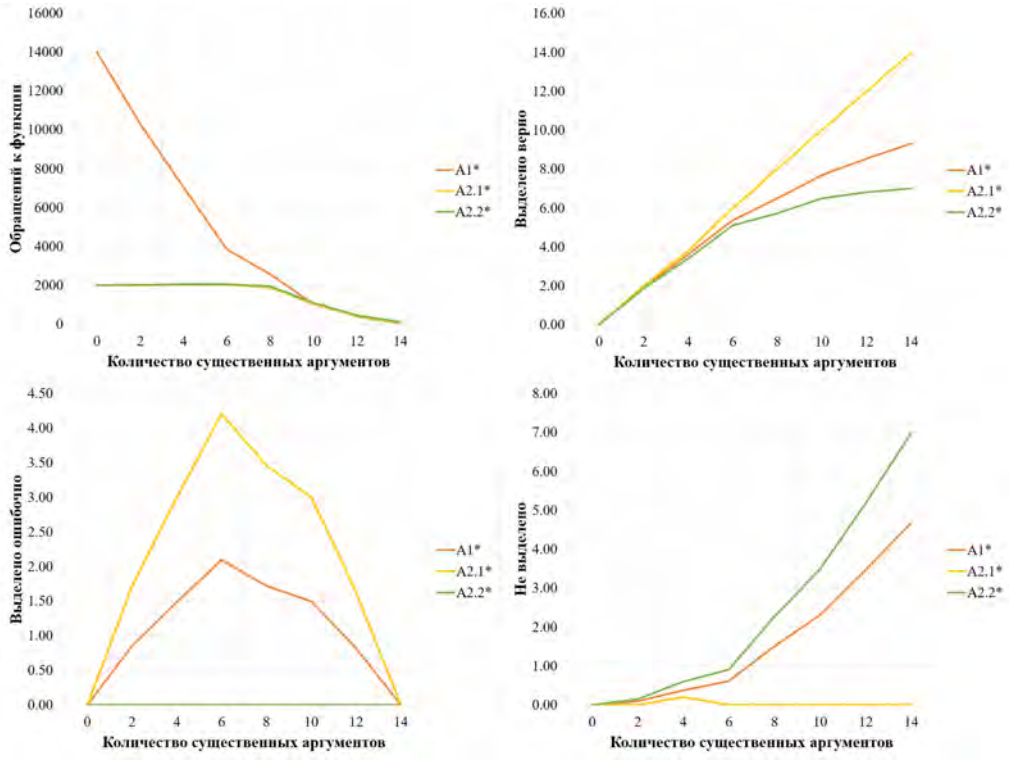

Рис. 5. Графическое представление зависимости характеристик модифицированных алгоритмов от количества существенных переменных

Построена конкретная модель конвейерной микроархитектуры на основе математической модели комбинационной и последовательностной логики.

Рассмотрена задача минимизации размерности для данных, которые аккумулируются в схеме. Для решения этой задачи проанализированы некоторые применяемые ранее вероятностные алгоритмы проверки свойств булевых функций, в частности проверки существенной зависимости функции от не более чем k переменных.

Предложены модификации этих алгоритмов и процедур для решения задачи 1 (см. раздел 3). Проведён анализ сложности и количества определяемых индексов при помощи изменённых алгоритмов.

Полученный в данной работе результат применим для обнаружения входов-выходов элементов комбинационной и последовательностной схемы, считывание данных с которых представляют угрозу нарушения конфиденциальности обрабатываемой информации в устройстве, использующем ППВМ.

Литература

1. Антонов А. А., Барабанов А. В., др. Цифровой синтез: практический курс / под общ. ред. А.Ю. Романова, Ю.В. Панчула. – М.: ДМК Пресс, 2020. – 556 с.
2. Xue M., Gu C., Liu W., et al. Ten years of hardware Trojans: a survey from the attacker's perspective. // IET Computers & Digital Techniques. 14. pp. 231-246. 2020. DOI: 10.1049/iet-cdt.2020.0041
3. Wan Z., Yu B., et al. A Survey of FPGA-Based Robotic Computing. // IEEE Circuits and Systems Magazine, 21, pp. 48-74. 2020. DOI: 10.1109/MCAS.2021.3071609.
4. Quraishi M., Tavakoli E., Ren F. A Survey of System Architectures and Techniques for FPGA Virtualization. // IEEE Transactions on Parallel & Distributed Systems, vol. 32, no. 09, pp. 2216-2230. 2021. DOI: 10.1109/TPDS.2021.3063670
5. Туринцев К. А., Поплавский Д. А., Калинкина А. А. Аппаратное средство для ускорения решения задач дизассемблирования // МОЛОДЫЕ УЧЁНЫЕ РОССИИ: сборник статей XVII Всероссийской научно-практической конференции. – Пенза: Наука и Просвещение, 2023. С. 32-37.
6. Zhang J., Qu G. Recent Attacks and Defenses on FPGA-based Systems. ACM Transactions on Reconfigurable Technology Syst. 12, 3, Article 14 (2019), p. 24. 2019. DOI: 10.1145/3340557
7. Деменкова Т. А., Певцов Е. Ф. Аппаратно-программные ресурсы защиты интегральных схем и интеллектуальных систем // Научно-технический вестник Поволжья. 2018. № 12. С. 213-218.
8. Huang H., Shen H., Li S., et al. A Hardware Trojan Trigger Localization Method in RTL based on Control Flow Features. 2022 IEEE 31st Asian Test Symposium (ATS), Taichung City, Taiwan, pp. 138-143. 2022. DOI: 10.1109/ATS56056.2022.00036
9. Palumbo A., Cassano L., Luzzi B., Hernandez J., et al. Is your FPGA bitstream Hardware Trojan-free? Machine learning can provide an answer. // Journal of Systems Architecture. Volume 128. 2022. DOI: 10.1016/j.sysarc.2022.102543
10. Chithra C., Kokila J., Ramasubramanian N. Detection of Hardware Trojans using Machine Learning in SoC FPGAs // 2020 IEEE International Conference on Electronics, Computing and Communication Technologies, Bangalore, India. pp. 1-7. 2020. DOI: 10.1109/CONECCT50063.2020.9198475
11. Zhang L., Dong Y., Wang J., et al. A hardware Trojan detection method based on the electromagnetic leakage. China Communications, vol. 16, no. 12, pp. 100-110. 2019. DOI: 10.23919/JCC.2019.12.007
12. Yu S., Gu C., Liu W., et al. A Novel Feature Extraction Strategy for Hardware Trojan Detection 2020 IEEE International Symposium on Circuits and Systems, Seville, Spain, 2020, pp. 1-5. DOI: 10.1109/ISCAS45731.2020.9180479
13. Cruz J., Posada C., Masna N., et al. A Framework for Automated Exploration of Trojan Attack Space in FPGA Netlists. IEEE Transactions on Computers. pp. 1-12. 2023. DOI: 10.1109/TC.2023.3266592
14. Ahmed Q., Platzner M. On the Detection and Circumvention of Bitstream-level Trojans in FPGAs // 2022 IEEE Computer Society Annual Symposium on VLSI , Nicosia, Cyprus. pp. 434-439. 2022. DOI: 10.1109/ISVLSI54635.2022.00097
15. Yang J., Zhang Y., Hua Y., et al. Hardware Trojans Detection Through RTL Features Extraction and Machine Learning. 2021 Asian Hardware Oriented Security and Trust Symposium (AsianHOST), Shanghai, China, pp. 1-4, 2021. DOI: 10.1109/AsianHOST53231.2021.9699658
16. Zhang Q., Liu L., Yuan Y., et al. A Gate-Level Information Leakage Detection Framework of Sequential Circuit Using Z3. // Electronics, 11, 4216. 2022.
17. Matrosova A., Provkin V. Applying Incompletely Specified Boolean Functions for Patch Circuit Generation // 2021 IEEE East-West Design & Test Symposium. Batumi, Georgia. pp. 1-4 2021. DOI: 10.1109/EWDTS52692.2021.9581029
18. Sabri M., Shabani A., Alizadeh B. SAT-Based Integrated Hardware Trojan Detection and Localization Approach Through Path-Delay Analysis // IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 68, no. 8. pp. 2850-2854. 2021. DOI: 10.1109/TCSII.2021.3074549
19. Xie Z., Daowen Q., Guangya C., et al. Testing Boolean Functions Properties. FundamFundamenta Informaticae 182(4). pp. 321-344. 2021. DOI: 10.3233/FI-2021-2076
20. De A., Mossel E., Neeman J. Junta Correlation is Testable // 2019 IEEE 60th Annual Symposium on Foundations of Computer Science, Baltimore, MD, USA. pp. 1549-1563. 2019. DOI: 10.1109/FOCS.2019.00090
21. Харрис Д. М., Харрис С. Л. Цифровая схемотехника и архитектура компьютера. / пер. с англ. Imagination Technologies. – М.: ДМК Пресс, 2018. – 792 с.: цв. ил.
22. Liu Z., Chen X., Servedio R., et al. Distribution-free Junta Testing. ACM Trans. Algorithms 15, 1, Article 1, 23 p. 2018. DOI: 10.1145/3264434
23. Bshouty N. Almost optimal distribution-free junta testing // 34th Computational Complexity Conference, CCC 2019, NJ, USA. pp. 2:1–2:13, 2019. DOI: 10.4230/LIPIcs.CCC.2019.2

ON THE ONE ALGORITHMS CLASS APPLICABILITY FOR THE COMPONENTS BEHAVIOR ANALYSIS OF DEVICES WITH FIELD-PROGRAMMABLE GATE ARRAYS

Titov A.S.[7], Gordeev E.N.[8]

The research purpose: *research of opportunities to increase the hardware security using detection of the register-transfer level schema parts that are at confidentiality violation risk.*

Methods: *the use of register-transfer level scheme mathematical modeling and the application of classical probabilistic algorithms for Boolean functions property testing to the model in order to locate potentially vulnerable areas in the microcircuit internal logic.*

Results: *based on the combinational and sequential circuits mathematical model, which defines the internal logic via Boolean function sets, a concrete pipeline microarchitecture model with split states and data transfer has been developed, which allows the one to research using chosen mathematical apparatus further.*

Singled out the processed by special computing devices data confidentiality intruder model. For the specific pipeline microarchitecture model and the confidentiality violator, that can be placed at any production process stage, the accumulated from the schema data dimension minimization problem has been considered.

In the context of the researching model, the complexity analysis of the one algorithms class has been performed. Based on the results of the analysis, modifications of some of the algorithms are proposed.

The constructed algorithms make it possible to find the location of input and output pins that are potentially vulnerable to sequential and combinational circuits elements confidentiality violations, by determining the indices of arguments of Boolean functions.

The scientific novelty: *consists in the one probabilistic algorithms class applicability analysis to the detection vulnerable schema logic device areas problem and in based on the algorithm's modification implementation in the purpose of vulnerable elements input pins detection accuracy increasing.*

Keywords: *complex programmable logic device, register-transfer level, pipeline microarchitecture, intruder model, Boolean functions, Boolean circuits, actual arguments.*

References

1. Antonov A. A., Barabanov A. V., dr. Cifrovoj sintez: prakticheskij kurs / pod obshh. red. A.Ju. Romanova, Ju.V. Panchula. – M.: DMK Press, 2020. – 556 s.
2. Xue M., Gu C., Liu W., et al. Ten years of hardware Trojans: a survey from the attacker's perspective. // IET Computers & Digital Techniques. 14. pp. 231-246. 2020. DOI: 10.1049/iet-cdt.2020.0041
3. Wan Z., Yu B., et al. A Survey of FPGA-Based Robotic Computing. // IEEE Circuits and Systems Magazine, 21, pp. 48-74. 2020. DOI: 10.1109/MCAS.2021.3071609.
4. Quraishi M., Tavakoli E., Ren F. A Survey of System Architectures and Techniques for FPGA Virtualization. // IEEE Transactions on Parallel & Distributed Systems, vol. 32, no. 09, pp. 2216-2230. 2021. DOI: 10.1109/TPDS.2021.3063670
5. Turincev K. A., Poplavskij D. A., Kalinkina A. A. Apparatnoe sredstvo dlja uskorenija reshenija zadach dizassemblirovanija // MOLODYE UChJoNYE ROSSII: sbornik statej XVII Vserossijskoj nauchno-prakticheskoj konferencii. – Penza: Nauka i Prosveshhenie, 2023. S. 32-37.
6. Zhang J., Qu G. Recent Attacks and Defenses on FPGA-based Systems. ACM Transactions on Reconfigurable Technology Syst. 12, 3, Article 14 (2019), p. 24. 2019. DOI: 10.1145/3340557
7. Demenkova T. A., Pevcov E. F. Apparatno-programmnye resursy zashhity integral'nyh shem i intellektual'nyh sistem // Nauchno-tehnicheskij vestnik Povolzh'ja. 2018. № 12. S. 213-218.
8. Huang H., Shen H., Li S., et al. A Hardware Trojan Trigger Localization Method in RTL based on Control Flow Features. 2022 IEEE 31st Asian Test Symposium (ATS), Taichung City, Taiwan, pp. 138-143. 2022. DOI: 10.1109/ATS56056.2022.00036
9. Palumbo A., Cassano L., Luzzi B., Hernandez J., et al. Is your FPGA bitstream Hardware Trojan-free? Machine learning can provide an answer. // Journal of Systems Architecture. Volume 128. 2022. DOI: 10.1016/j.sysarc.2022.102543

7 Anatoliy S. Titov, student of the «Information Security» department, Bauman Moscow State Technical University, Moscow, Russia. E-mail: toliakpurple@gmail.com

8 Eduard N. Gordeev, Dr.Sc. (Math.), Professor of the «Information Security» department, Bauman Moscow State Technical University, Moscow, Russia. E-mail: werhorn@yandex.ru

10. Chithra C., Kokila J., Ramasubramanian N. Detection of Hardware Trojans using Machine Learning in SoC FPGAs // 2020 IEEE International Conference on Electronics, Computing and Communication Technologies, Bangalore, India. pp. 1-7. 2020. DOI: 10.1109/CONECCT50063.2020.9198475
11. Zhang L., Dong Y., Wang J., et al. A hardware Trojan detection method based on the electromagnetic leakage. China Communications, vol. 16, no. 12, pp. 100-110. 2019. DOI: 10.23919/JCC.2019.12.007
12. Yu S., Gu C., Liu W., et al. A Novel Feature Extraction Strategy for Hardware Trojan Detection 2020 IEEE International Symposium on Circuits and Systems, Seville, Spain, 2020, pp. 1-5. DOI: 10.1109/ISCAS45731.2020.9180479
13. Cruz J., Posada C., Masna N., et al. A Framework for Automated Exploration of Trojan Attack Space in FPGA Netlists. IEEE Transactions on Computers. pp. 1-12. 2023. DOI: 10.1109/TC.2023.3266592
14. Ahmed Q., Platzner M. On the Detection and Circumvention of Bitstream-level Trojans in FPGAs // 2022 IEEE Computer Society Annual Symposium on VLSI , Nicosia, Cyprus. pp. 434-439. 2022. DOI: 10.1109/ISVLSI54635.2022.00097
15. Yang J., Zhang Y., Hua Y., et al. Hardware Trojans Detection Through RTL Features Extraction and Machine Learning. 2021 Asian Hardware Oriented Security and Trust Symposium (AsianHOST), Shanghai, China, pp. 1-4, 2021. DOI: 10.1109/AsianHOST53231.2021.9699658
16. Zhang Q., Liu L., Yuan Y., et al. A Gate-Level Information Leakage Detection Framework of Sequential Circuit Using Z3. // Electronics, 11, 4216. 2022.
17. Matrosova A., Provkin V. Applying Incompletely Specified Boolean Functions for Patch Circuit Generation // 2021 IEEE East-West Design & Test Symposium. Batumi, Georgia. pp. 1-4 2021. DOI: 10.1109/EWDTS52692.2021.9581029
18. Sabri M., Shabani A., Alizadeh B. SAT-Based Integrated Hardware Trojan Detection and Localization Approach Through Path-Delay Analysis // IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 68, no. 8. pp. 2850-2854. 2021. DOI: 10.1109/TCSII.2021.3074549
19. Xie Z., Daowen Q., Guangya C., et al. Testing Boolean Functions Properties. FundamFundamenta Informaticae 182(4). pp. 321-344. 2021. DOI: 10.3233/FI-2021-2076
20. De A., Mossel E., Neeman J. Junta Correlation is Testable // 2019 IEEE 60th Annual Symposium on Foundations of Computer Science, Baltimore, MD, USA. pp. 1549-1563. 2019. DOI: 10.1109/FOCS.2019.00090
21. Harris D. M., Harris S. L. Cifrovaja shemotehnika i arhitektura komp'jutera. / per. s angl. Imagination Technologies. – M.: DMK Press, 2018. – 792 s.: cv. il.
22. Liu Z., Chen X., Servedio R., et al. Distribution-free Junta Testing. ACM Trans. Algorithms 15, 1, Article 1, 23 p. 2018. DOI: 10.1145/3264434
23. Bshouty N. Almost optimal distribution-free junta testing. // 34th Computational Complexity Conference, CCC 2019, NJ, USA. pp. 2:1–2:13, 2019. DOI: 10.4230/LIPIcs.CCC.2019.2