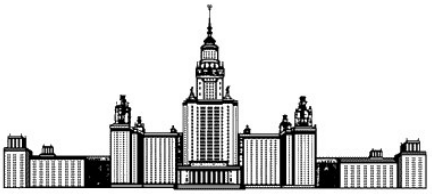

МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ М. В. ЛОМОНОСОВА
ФАКУЛЬТЕТ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ И КИБЕРНЕТИКИ
КАФЕДРА МАТЕМАТИЧЕСКОЙ КИБЕРНЕТИКИ

Отчет по преддипломной практике

«Разработка и реализация алгоритма распознавания некоторых свойств схем из функциональных элементов»

студента 418 группы

Трубицына Юрия Алексеевича

Научный руководитель:
ассистент, к. ф. - м. н.
ШУПЛЕЦОВ МИХАИЛ СЕРГЕЕВИЧ

Руководитель практики от факультета:
ассистент, к. ф. - м. н.
ШУПЛЕЦОВ МИХАИЛ СЕРГЕЕВИЧ

Москва, 2016

Оглавление

1. Введение

Основной тематикой данной работы является проблема несанкционированного копирования IP блоков и интегральных схем. В атаках данного типа злоумышленник, получив доступ к описанию интегральной схемы или IP блока, производит его несанкционированное копирование. В дальнейшем, злоумышленник может внести небольшие изменения в украденное описание, чтобы производить и продавать его в качестве своего собственного.

Рассмотрим один из видов атак подобного рода - обратное проектирование. Обратное проектирование интегральной схемы предполагает выполнение некоторых из следующих действий:

1. определение технологии, по которой была произведена интегральная схема [1];

2. извлечение из готовой интегральной схемы ее описания на уровне логических элементов [2];

3. определение функциональности, реализуемой интегральной схемой [3];

Существует несколько подходов к защите от атак подобного рода:

1. схемная обфускация;

2. маскировка интегральных схем (camouflaging);

3. раздельное производство.

В моей преддипломной практике рассматривается второй способ защиты - маскировка интегральных схем.

Маскировка интегральных схем – это набор методов, изменяющих топологическое описание (layout) интегральной схемы для того, чтобы затруднить работу методов обратного проектирования, которые основаны на алгоритмах распознавания образов, работающих с изображениями слоев, извлеченных из интегральной схемы. Так, в случае, когда интегральная схема производится с использованием методологии стандартных ячеек (standart cells), используются преобразования, которые стараются сделать все ячейки похожими друг на друга, что приводит к ошибкам при извлечении информации о логическом описании схемы. Например, пустое пространство(white space) схемы может быть заполнено специальными фильтрующими ячейками [4], могут использоваться программируемые стандартные ячейки [5], а также фиктивные соединения (проводники) [6].

В данной работе используется следующий метод маскировки интегральной схемы - удаляются отдельные связи(провода) между функциональными элементами. Также возможно присутствие фиктивных частей в схеме.

В данной работе предлагаются алгоритмы и библиотеки, которые помогут провести идентификацию классов (монотонные, линейные, самодвойственные и т. д.) исходной схемы по замаскированной исходной схеме при помощи алгоритмов машинного обучения.

2. Основные определения

Введем ряд основных определений необходимых для формальной постановки задачи идентификации классов исходной схемы по замаскированной исходной схеме.

Определение 2.1. $\mathbb{B}^n = \left\{\tilde{\alpha}^n = (\alpha_1, ..., \alpha_n) \mid \alpha_i \in \{0,1\}\, \forall i = \overline{1,n}\right\}$.

Определение 2.2. *Булевой функцией* от переменных $x_1, ..., x_n$ будем называть отображение $f : \mathbb{B}^n \to \mathbb{B}^1$.

Определение 2.3. Схемой из функциональных элементов (СФЭ) будем называть ориентированный помеченный граф, обладающий следующими свойствами:

1. в нем существует непустое множество вершин, помеченных как входные вершины;
2. в нем существует непустое множество вершин, помеченных как выходные вершины;
3. множества входных и выходных вершин не пересекаются;
4. вершины, не входящие в множество входных вершин, помечены символом некоторой БФ.

Каждая метка вершины из множества входных вершин имеет смысл булевой переменной. Пусть теперь символы, которыми могут быть помечены вершины, могут соответствовать только следующим БФ : NOT, AND, OR, XOR, NAND, NOR, XNOR.

Определение 2.4. Булеву функцию, реализуемую в вершине СФЭ определим по индукции:

1. если вершина входит во множество входных вершин и помечена как x_i, то БФ, реализуемая в этой вершине есть: $f = x_i$;
2. если вершина не является входом и помечена символом некоторой БФ $F(p_1, ..., p_k)$, то в данной вершине реализуется БФ: $f = F(f_{i_1}, ..., f_{i_k})$, где $f_{i_1}, ..., f_{i_k}$ есть функции, реализуемые в вершинах, из которых исходят входящие в данную вершину ребра;

Определение 2.5. Частично заданной СФЭ (замаскированной СФЭ) Σ будем называть такую схему Σ', которая получается путем удаления одного или нескольких ребер из исходной схемы Σ.

Определение 2.6. Одновыходной СФЭ будем называть такую СФЭ, у которой множество выходных вершин содержит всего одну вершину графа.

Далее будем рассматривать только одновыходные схемы, т. е. схемы с единственной выходной вершиной.

Определение 2.7. Классом БФ будем называть подмножество множества всех БФ, удовлетворяющих некоторому условию.

Определим несколько классов.

Определение 2.8. $T_1 = \cup_{n=1}^{\infty}\{f\,(x_1, ..., x_n) \in P_2^n | f(1, ..., 1) = 1\}$.

Определение 2.9. $T_0 = \cup_{n=1}^{\infty}\{f\,(x_1, ..., x_n) \in P_2^n | f(0, ..., 0) = 0\}$.

Определение 2.10. Пусть $\overline{\alpha} = (\alpha_1, ..., \alpha_n)$ и $\overline{\beta} = (\beta_1, ..., \beta_n)$, тогда: $\overline{\alpha} \leq (\geq)\overline{\beta} \Leftrightarrow \forall i \in \{1, ..., n\} : \alpha_i \leq (\geq)\beta_i$.

Определение 2.11. $M = \cup_{n=1}^{\infty}\{f\,(x_1, ..., x_n) \in P_2^n \mid \forall\overline{\alpha} \leq \overline{\beta} : f\,(\overline{\alpha}) \leq f\left(\overline{\beta}\right)\}$.

Определим понятие вхождения СФЭ в класс.

Определение 2.12. Одновыходная СФЭ Σ входит в класс $G \in P_2$ тогда и только тогда, когда в этот класс входит функция, реализуемая на выходе СФЭ.

Определение 2.13. Восстановлением по частично заданной одновыходной СФЭ Σ над множеством классов Γ будем называть определение вхождения частично заданной одновыходной СФЭ Σ в каждый из классов множества Γ. При этом оценкой качества восстановления над множеством Γ будем называть долю классов, таких что исходная одновыходная СФЭ Σ входит (не входит), а частично заданная одновыходная СФЭ Σ не входит (входит), т. е. количество различий между вхождениями СФЭ.

Очевидно, чем ниже оценка, тем лучше восстановление.

3. Постановка задачи

1. Реализовать и протестировать способ представления СФЭ на ЭВМ;
2. Выделить набор признаков СФЭ, которые будут использоваться для машинного обучения;
3. Реализовать и протестировать алгоритмы вычисления признаков схемы;
4. Реализовать алгоритм построения случайных СФЭ.

4. Основная часть

4.1. Реализация и тестирование способа представления СФЭ на ЭВМ

Имеется класс Vertex и унаследованный от него класс Gate. Vertex предназначен для хранения входных вершин. Gate хранит данные о вершинах, не являющихся входными. Vertex - имеет поля имени, выходной степени и глубины. Унаследованный от него Gate ещё имеет поле для хранения входных ребер и типа(OR,XOR и т. д.).

Реализован класс Parser, который позволяет получать значащие лексемы из файла-описания СФЭ.

СФЭ реализована на ЭВМ как отдельный класс SFE с контейнером указателей на входные вершины(inputs), контейнером указателей на выходные вершины(ouputs) и контейнером указателей на вершины, которые не являются ни входными, ни выходными(gates). Схема задается в файле на языке Verilog. Также реализован ряд методов для работы с данным классом.

Данный набор классов реализован и полность протестирован на разлиных схемах.

4.2. Выделение набора признаков СФЭ, которые будут использоваться для машинного обучения

Были выделены следующие признаки:

1. процент каждого возможного функционального элемента из множества NOT, AND, OR, XOR, NAND, NOR, XNOR;
2. максимальная входная/выходная степень;
3. минимальная входная/выходная степень;
4. средняя входная/выходная степень;
5. средняя глубина, нормированная на максимальную глубину;
6. среднее количество значащих переменных, нормированное на общее количество переменных;

Как рассчитывать первые три пункта - очевидно. Приведем рассчетные формулы для остальных пунктов. Введем ряд обозначений:

1. $d^+(v)$ - входная степень вершины v, равна количеству входных ребер;
2. $d^-(v)$ - выходная степень вершины v, равна количеству выходных ребер;
3. I - множество всех входных вершин СФЭ;
4. O - множество всех выходных вершин СФЭ;
5. V - множество всех вершин СФЭ;
6. $D(v)$ - глубина вершины v;
7. $D(\Sigma)$ - глубина СФЭ Σ;
8. $CS(v)$ - значащие переменные вершины v;

Расчет средней входной степени: $\frac{\sum\limits_{v \in V} d^+(v)}{|V|}$. Выходной аналогично.

Расчет средней глубины, нормированной на максимальную глубину: $\frac{\sum\limits_{v \in V} D(v)}{D(\Sigma)}$.

Вычисление среднего количества значащих переменных, нормированного на общее количество переменных: $\frac{\sum\limits_{v \in V \setminus I} CS(v)}{|I|}$.

4.3. Реализация и тестирование алгоритмов вычисления признаков схемы

В классе SFE реализованы и протестированы следующие методы для расчетов признаков СФЭ:

1. SFE::getPercentageTypeGate(typeGate t) - процент каждого возможного функционального элемента из множества NOT, AND, OR, XOR, NAND, NOR, XNOR;

2. SFE::getMaxInputDegree() / SFE::getMaxOutputDegree() - максимальная входная/выходная степень;

3. SFE::getMinInputDegree() / SFE::getMinOutputDegree() - минимальная входная/выходная степень;

4. SFE::getMiddleInputDegree() / SFE::getMiddleInputDegree() - средняя входная/выходная степень;

5. SFE::getPercentageMiddleDepth() - средняя глубина, нормированная на максимальную глубину;

6. SFE::getPercentageMiddleSignVar() - среднее количество значащих переменных, нормированное на общее количество переменных.

4.4. Реализация алгоритма построения случайных СФЭ

Для тестирования полученных алгоритмов, а также для генерации обучающих выборок СФЭ для машинного обучения, был реализован отдельный алгоритм, генерирующий случайную СФЭ, представленную в виде файла-описания на Verilog. Алгоритму нужно подать на вход максимальное количество входов, выходов и элементов, не являющихся ни входами, ни выходами, а также имя выходного файла. На основании этих данных алгоритм генерирует случайную СФЭ.

5. Полученные результаты

Разработана реализация на языке C++ СФЭ на ЭВМ с возможность вычисления описанных выше признаков и считывания описания с файла-описания. Разработан генератор случайных СФЭ.

6. Приложение

https://github.com/yura03101995/PracticeMSU.git
По данной ссылке можно найти весь исходный код преддипломной практики.

Литература

[1] Chipworks: *"Intel's 22-nm Tri-gate Transistors Exposed"*, http://www.chipworks.com/blog/technologyblog/2012/04/23/intels-22-nm-tri-gate-transistors-exposed/, 2012

[2] R. Torrance and D. James: *"The state-of-the-art in semiconductor reverse engineering"*, IEEE/ACM Design Automation Conference, pp. 333–338, 2011

[3] DARPA: *"Integrity and Reliability of Integrated Circuits (IRIS)"*, http://www.darpa.mil/Our_Work/MTO/Programs/Integrity_and_Reliability_of_Integrated_Circuits_(IRIS).aspx, 2012

[4] J. P. Baukus, L. W. Chow, R. P. Cocchi, P. Ouyang, and B. J. Wang: *"Camouflaging a standard cell based integrated circuit"*, US Patent no. 8151235, 2012

[5] J. P. Baukus, L. W. Chow, R. P. Cocchi, P. Ouyang, and B. J. Wang: *"Building block for a secure CMOS logic cell library"*, US Patent no. 8111089, 2012

[6] SypherMedia: *"Syphermedia library circuit camouflage technology"*, http://www.smi.tv/solutions.html