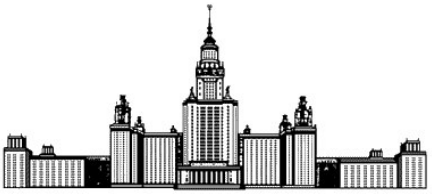

МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ М. В. ЛОМОНОСОВА
ФАКУЛЬТЕТ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ И КИБЕРНЕТИКИ
КАФЕДРА МАТЕМАТИЧЕСКОЙ КИБЕРНЕТИКИ

Отчет по преддипломной практике

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

студента 418 группы

Трубицына Юрия Алексеевича

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

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

Москва, 2016

Оглавление

1. Введение

Основной темой является задача восстановления функциональности для частично заданных СФЭ, которая возникает при попытках скрыть или защитить схему от несанкционированного копирования. 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 третий способ защиты - маскировка интегральных схем.

Раздельное производство интегральных схем – набор методологий производства интегральных схем, отличительной особенностью которых является производство интегральной схемы, когда схема изготовляется на нескольких различных фабриках [6]. В основном, в литературе выделяют два основных подхода к раздельному производству интегральных схем:

1. раздельное производство различных слоев интегральной схемы [4];
2. подходы, основанные на 2.5D и 3D интеграции [5].

Современная интегральная схема состоит множества слоев, которые последовательно изготовляются на поверхности кремниевой пластины при помощи фотолитографического процесса. При этом нижние слои используются для производства транзисторов, то есть основных логических элементов схемы, а верхние уровни задействованы для расположения слоев металлизации, которые используются для соединения логических элементов интегральной схемы. На современном уровне технологий типичная интегральная схема может содержать от 10 до 12 слоев металлизации. При этом минимальное расстояние между проводниками и их линейные размеры на разных слоях отличается. В связи с технологическими ограничениями на верхних слоях расстояние между проводниками и их ширина и высота больше, чем на нижних слоях. При раздельном

производстве производство интегральной микросхемы разделяется на два этапа, которые могут быть выполнены на разных фабриках. На первом этапе на кремниевой пластине создаются транзисторы и создаются нижние слои металлизации (например, до уровня M4 включительно). На втором этапе недостающие слои металлизации наносятся на, полученную на первом этапе, заготовку. Выбор уровня металлизации на котором происходит разделение производства является произвольным и зависит от целей и требований заказчика. Подходы, основанные на 2.5D и 3D интеграции, предполагают интеграцию нескольких кремниевых пластин внутри одного корпуса. При этом пластины располагаются одна над другой и соединяются при помощи специальных прорезов в кремнии, так называемых, TSV (through silicon vias). Чаще всего эти прорезы располагают в узлах регулярной сетки с некоторым фиксированным шагом. 2.5D интеграция предполагает соединение нескольких интегральных схем на специальной коммутирующей пластине, так называемом интерпозере, который состоит только из слоев металлизации, то есть держит только проводники и не содержит логических элементов. Раздельное производство имеет ряд технологических сложностей. Так, например, после первого этапа требуется транспортировка кремниевых пластин на другую фабрику. Из-за своей хрупкости пластины могут деформироваться и даже трескаться, а также могут возникать повреждения верхнего слоя, что может привести к невозможности нанесения последующих слоев на другой фабрике. Кроме того, в начале второго этапа производства требуется очень аккуратная настройка оборудования и выравнивание кремниевых пластин или отдельных интегральных схем, для обеспечения высокого уровня выхода годных. Раздельное проектирование существенно затрудняет успешность обратного проектирования, так как злоумышленник, который работает на производстве, может получить только часть интегральной схемы, которая не будет корректно работать, пока она не будет соединена с другой частью, произведенной на другой фабрике.

В данной работе используется следующий метод - 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 ациклический 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 связана с соответствующей 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. При этом вершины, инцидентные удаленным ребрам помечаются особым образом.

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: $\overline{\alpha} \leq \overline{\beta} \Leftrightarrow \forall i \in \{1, ..., n\} : \alpha_i \leq \beta_i$.

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.

3. Постановка задачи

1. Выделить набор признаков СФЭ, которые будут использоваться для решения задачи распознавания;
2. Реализовать и протестировать алгоритмы вычисления признаков СФЭ;
3. Реализовать алгоритм построения случайных СФЭ с фиксированным значением выделенных признаков.

4. Основная часть

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

Были выделены следующие признаки:

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

Рассчитывать первые три пункта достаточно просто: доля возможного функционального элемента из множества NOT, AND, OR, XOR, NAND, NOR, XNOR это есть число вершин, помеченных этим функциональным элементом деленное на общее число вершин; максимальная/минимальная полустепень исхода/захода вершин - очевидно. Приведем рассчетные формулы для остальных пунктов. Введем ряд обозначений:

1. $d^+(v)$ - полустепень захода вершины v, равна количеству входных ребер;
2. $d^-(v)$ - полустепень исхода 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. $CS(v)$ - существенные переменные вершины v;

Расчет средней полустепени захода: $\frac{\sum\limits_{v \in V} d^+(v)}{|V|}$. Исхода 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: $\frac{\sum\limits_{v \in V \setminus I} CS(v)}{|I|}$.

4.2. Реализация способа представления СФЭ на ЭВМ

Для вычисления выбранных характеристик СФЭ был реализован следующий набор классов: 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 синтаксический анализатор СФЭ, заданных при помощи языка Verilog. Анализатор представлен в виде класса Parser, в котором реализован метод, возвращающий только значащие лексемы.

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.

4.3. Реализация алгоритмов вычисления признаков схемы

В классе SFE реализованы следующие методы для рассчетов признаков СФЭ:

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

2. SFE::getMaxInputDegree() / SFE::getMaxOutputDegree() - максимальная полустепень исхода/захода вершин - проходим по всем вершинам, считаем полустепень исхода/захода, храним максимальную;

3. SFE::getMinInputDegree() / SFE::getMinOutputDegree() - минимальная полустепень исхода/захода вершин - проходим по всем вершинам, считаем полустепень исхода/захода, храним минимальную;

4. SFE::getMiddleInputDegree() / SFE::getMiddleInputDegree() - средняя полустепень исхода/захода вершин - проходим по всем вершинам, считаем сумму полустепеней исхода/захода, делим на количество вершин, умноженное на максимальную полустепень исхода/захода;

5. SFE::getPercentageMiddleDepth() - средняя глубина, нормированная на максимальную глубину - проходим по всем вершинам, считаем сумму глубин, делим на количество вершин, умноженное на глубину СФЭ;

6. SFE::getPercentageMiddleSignVar() - среднее количество существенных переменных, нормированное на общее количество переменных - проходим по всем вершинам, считаем сумму количества существенных переменных для каждой вершины, делим на количество вершин, умноженное на количество переменных.

Помимо вычисления определенных нами признаков, в классе SFE имеются функции для вычисления принадлежности к классам T_1, T_0, M. Вычисление проводится строго по определению этих классов.

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.

В данном алгоритме вершины следуют в строгом порядке: сначала входы, потом внутренние вершины(ни входы, ни выходы), затем выходы. Сначала случайным образом выбирается тип функционального элемента(ФЭ), затем определяется количество входов в зависимости от типа ФЭ, и, наконец, записываются входы данного ФЭ, которые берутся из множества элементов, строго предшествующих данному. Такой подход позволяет избежать циклов и построить 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. Весь исходный код доступен по ссылке:
https://github.com/yura03101995/PracticeMSU.git

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] Jeyavijayan (Jv) Rajendran, Ozgur Sinanoglu, and Ramesh Karri: *"Is split manufacturing secure?. In Proceedings of the Conference on Design, Automation and Test in Europe (DATE '13)"*, EDA Consortium, San Jose, CA, USA, 1259-1264, 2013

[5] Frank Imeson, Ariq Emtenan, Siddharth Garg, and Mahesh V. Tripunitara: *"Securing computer hardware using 3D integrated circuit (IC) technology and split manufacturing for obfuscation. In Proceedings of the 22nd USENIX conference on Security (SEC'13)."*, USENIX Association, Berkeley, CA, USA, 495-510, 2013

[6] R.W Jarvis and M. G. McIntyre: *"Split manufacturing method for advanced semiconductor circuits."*, US Patent no. 7195931, 2004