Список использованных источников:

1. Э. Таненбаум, М. Ван Стеен. Распределенные системы. Принципы и парадигмы // СПб.: Питер, 2003. 877 с.

ИСПОЛЬЗОВАНИЕ ЛЕКСИЧЕСКОЙ ОБФУСКАЦИИ VHDL-ОПИСАНИЙ ДЛЯ ВНЕДРЕНИЯ ВОДЯНЫХ ЗНАКОВ.

Видничук В.Н.

Белорусский государственный университет информатики и радиоэлектроники

г. Минск, Республика Беларусь

Иванюк А.А. – д.т.н., доцент

В статье описывается метод лексической обфускации VHDL- описаний, его основные положения а также метод внедрения водяных знаков в это описание. Демонстрируются множества символов используемых для обфускации описаний.

В наши дни быстрыми темпами растёт производство цифровой техники и в связи с этим растёт проблема пиратства. Для решения этой проблемы предлагается рассмотреть метод внедрения водяного знака в лексическую обфускацию VHDL-описаний.

Обфускация (от англ. obfuscate – делать неочевидным, запутанным, сбивать с толку) – широко известная методика защиты исходных кодов программ от обратного проектирования [1].

Основной целью лексической обфускации является затруднение понимания исходных кодов программы. Лексическая обфускация это один из многих способов затруднить понимания функционирования программы.

Частным случаем лексической обфускации является переименование идентификаторов VHDL-описания на похожие друг на друга. Название идентификатора можно представить в виде множества символов:

$$< Id >= \{[A..z] \cap [\ \ ..\ \ ] \cap [0..9]\}$$

где, идентификатор не должен начинаться с цифры и может состоять из латинских и кириллических символов. В связи с этим набор символов при выполнении метода переименования идентификаторов можно представить следующим множеством:

$$< Id' >= \{O, O', 0, Q, 1, l, I, j\},$$

в котором O' является кириллическим символом O и O латинская являются разными символами, однако в латинском языке могут использоваться диакрические знаки например: Í

$$< Di >= \{\acute{O}, \tilde{O}, \grave{I}\ \acute{I}, \quad , \quad , \quad , \quad , \hat{I}, \ddot{I}\},$$

при синтезе VHDL-описания данные символы приводятся к латинской O, однако имеют разные коды символов, поэтому при выполнении анализа лексически обфусцированного кода могут возникать ошибки и непонятности. Отсюда, следует возможность использования диакрических символов для формирования названий идентификаторов повышенной сложности.

Далее приведён пример использования диакрических символов для затруднения понимания кода: идентификатор OIOOIIOIIOIIOIÖ при синтезе будет равняться идентификатору OIOOIIOIIOIIOIO или OIOOIIOIIOIIOIOÓ, но при попытке найти идентификатор OIOOIIOIIOIIOIÖ с помощью поиска или сторонних программ найдётся только он сам. Следовательно, с помощью использования данных символов можно запутывать не только человека, но и программные средства анализа кода или, например, идентификатор OIO'OIIOIIOILÖ и идентификатор OIOOIIOIIOILÖ для человека будут выглядеть как идентичные, однако при синтезе они будут являться разными идентификаторами.

При использовании метода переименования идентификаторов следует знать, что все идентификатора должны быть похожи и отличаться друг от друга на 1 символ, в связи с этим существует возможность внедрения водяного знака в VHDL-описание путём выбора мест замены символов в соответствии с ключом водяного знака.

Данный метод можно применять несколькими способами, например, генерируется первичный идентификатор для переименования, состоящий из нулей и единиц, символ выбирается с помощью равномерного распределения. В итоге получается идентификатор вида: 100101001011 это делается для того, чтобы расстояние между двумя идентификаторами не превышало 1. После этого

осуществляется случайная замена всех единичных идентификаторов на символы, находящиеся в множестве:

$$<1_{mn}>=\{1, l, I, j\},$$

А для замены нулевых символов используется множество:

$$<0_{mn}>=\{O, O', 0, Q\},$$

В итоге получается уникальный идентификатор, состоящий из похожих символов: IOQ1O’lO0jO’I1. Далее анализируется количество уникальных идентификаторов в описании и формируется ключ водяного знака: 010010000010 данный ключ определяет какие из символов первоначально сгенерированного идентификатора могут изменяться при формировании названий остальных идентификаторов. Новые идентификаторы генерируются с использованием множеств пподстановки 1_{mn} и 0_{mn}. Для примера были сгенерированы остальные идентификаторы для данного описания: IO’Q1O’lO0jO’I1, IOQ1QlO0jO’I1, IOQ1O’lO0jO’j1 проанализировав изменения, в которых можно определить водяной знак, зашифрованный в данных идентификаторах.

Далее в идентификаторах случайным образом изменяются символы: I, O на диакрические принадлежащие множеству Di. Данный метод позволяет усложнить работу с исходным кодом VHDL-описания и запутывает его алгоритмы обработки. В результате выполнения данного метода получаются следующие идентификаторы: IÖQ1O’ÍO0jO’I1, IO’Q1O’Ȭ0jO’I1, IOQ1QlȮ0jO’I1, ÌQ1O’lO0jO’j1 причём при каждом использовании данных идентификаторов в описании один или несколько символов O, I или их диакрические аналоги могут изменяться на другие диакрические символы, что усложняет понимание кода.

По результатам данной работы было разработано программное средство обфускации идентификаторов в VHDL-описаниях и внедрении в них водяного знака. Пример работы данного программного средства предоставлен на следующем рисунке:

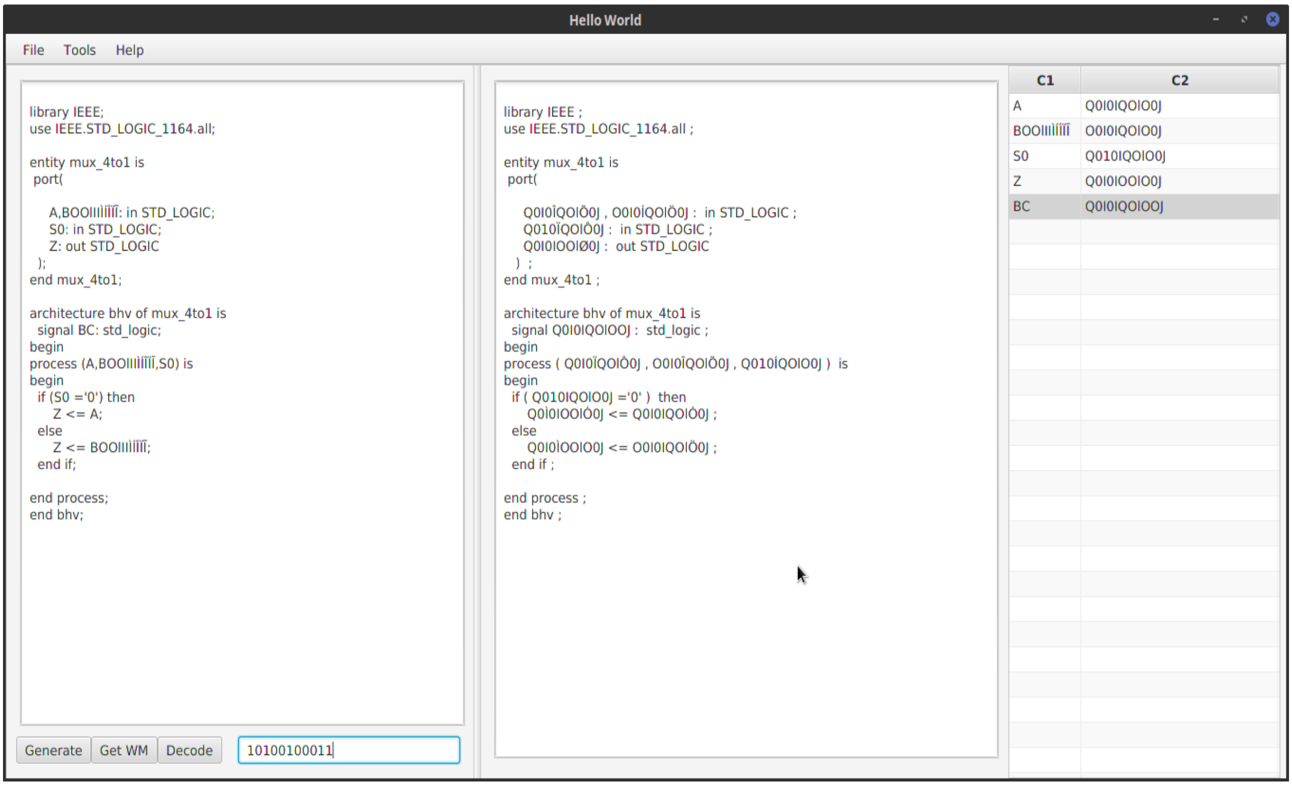

Рисунок 1 – Пример работы программного средств обфускации VHDL-описаний.

Стоит заметить, что полученное обфусцированное описание является полностью синтезируемым и результат синтеза не отличается от оригинального, следовательно, лексическая обфускация идентификаторов прошла успешно.

В результате проведённого исследования получилось сформировать метод внедрения водяного знака в лексическую обфускацию VHDL-описаний, а также разработать прототип программного средства для реализации данной функции.

Список использованных источников:

1. Collberg, C. A Taxonomy of Obfuscating Transformations / C. Collberg, C. Thomborson, D. Low – Auckland: Department of Computer Science, 1997. – 36 p.

2. Сергейчик В. В. Методы лексической обфускации VHDL-описаний / В. В. Сергейчик, А. А. Иванюк // Информационные технологии и системы 2013 (ИТС 2013) : материалы международной научной конференции, БГУИР, Минск, Беларусь, 23 октября 2013 г. = Information Technologies and Systems 2013 (ITS 2013) : Proceeding of The International Conference, BSUIR, Minsk, 24th October 2013 / редкол.: Л. Ю. Шилин [и др.]. - Минск : БГУИР, 2013. – С. 198-199.

3. Garg S., Gentry C., Halevi S., Raykova M., Sahai A., and Waters B. «Candidate indistinguishability obfuscation and functional encryption for all circuits.» FOCS 2013.

СРАВНЕНИЕ СВЕРТОЧНЫХ И РЕКУРРЕНТНЫХ НЕЙРОННЫХ СЕТЕЙ В ЗАДАЧЕ АНАЛИЗА ТОНАЛЬНОСТИ ТЕКСТА

Витковский А.В.

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Жвакина А.В. – к.т.н.,доцент

В обработке естественных языков существует задача определения тональности текста. В настоящее время для решения этой задачи используются искусственные нейронные сети. Для определения тональности могут быть использованы сети с различной архитектурой. В данном исследовании сравниваются сверточная нейронная сеть и сеть с долгой краткосрочной памятью.

Обработка естественного языка – направление искусственного интеллекта и математической лингвистики. Оно изучает проблемы компьютерного анализа и синтеза естественных языков. Анализ тональности текста является одной из задач, решаемых в рамках обработки языка. Суть задачи заключается в автоматическом распознавании в тексте его эмоциональной окраски. Классификация бывает бинарной, при которой определяется позитивную и негативную окраску имеет текст. Также существует классификация на несколько классов, при которой окраска может быть нейтральной или разной степени позитивности или негативности.

Две различные архитектуры нейронной сети будут сравниваться при решении задачи бинарной классификации. В качестве выборки используется выборка отзывов с сайта IMDB. Входными данными для обоих сетей будут являться тексты, где каждой слово закодировано целым числом, при этом значение зависит от частоты появления слова в выборке данных.

Сверточная нейронная сеть (Convolutional Neural Network, CNN) является специальной архитектурой, в основном применяемой для распознавания образов на изображениях. Название архитектура сети получила из-за наличия операции свёртки, суть которой в том, что каждый фрагмент изображения умножается на матрицу (ядро) свёртки поэлементно, а результат суммируется и записывается в аналогичную позицию выходного изображения. Несмотря на основное применение, ее также можно использовать для работы с текстом. Если в случае изображений, фильтр сверточного слоя применяется для нескольких соседних пикселей, то в случае текста фильтр можно применять для нескольких соседних слов.

Долгая краткосрочная память (Long short-term memory, LSTM) – один из видов рекуррентных нейронных сетей. В рекуррентных нейронных сетях связи между нейронами имеют направленную последовательность. Такие сети могут использовать свою внутреннюю память для обработки последовательностей произвольной длины. LSTM-сеть содержит LSTM-модули. LSTM-модуль – это рекуррентный модуль сети, способный запоминать значения как на короткие, так и на длинные промежутки времени. Такое поведение обусловлено тем, что LSTM-модуль не использует функцию активации внутри своих компонентов.

Для обоих нейронных сетей первый слой одинаков и является слоем, осуществляющим замену целых чисел, обозначающих слова, на векторное представление слов. Этот слой обучается в ходе тренировки модели в обоих случаях. В обоих случаях функцией потерь является перекрестная энтропия, обучения происходит при помощи метода обратного распространения ошибки с алгоритмом Adam. Объем тестовых и валидационных данных также одинаков для обоих моделей. Различными являются только сами модели нейронных сетей.

Модель сверточной нейронной сети состоит из трех каналов, каждый канал состоит из сверточного слоя, слоя активации, слоя подвыборки и полносвязного слоя. каналы в модели параллельны. Данная модель позволяет задать различные размеры фильтра сверточного слоя в каждом канале. В реализованной моедели использовались фильтры по 3, 4 и 5 слов. Каналы объединяются при помощи полносвязного слоя.