МЕТОДЫ ЛЕКСИЧЕСКОЙ ОБФУСКАЦИИ VHDL-ОПИСАНИЙ

Сергейчик В. В., Иванюк А. А.
Кафедра ПОИТ, Кафедра ВМиП, Белорусский государственный университет информатики и радиоэлектроники
Минск, Республика Беларусь
E-mail: vovasq@mail.ru, ivaniuk@bsuir.by

Рассматривается лексическая обфускация VHDL-описаний и способы оценки сложности обфусцирующих преобразований.

Введение

В настоящее время быстрыми темпами растёт объём производства цифровых устройств, и в связи с этим особую остроту приобретает проблема пиратства. По некоторым данным ущерб от пиратства и других угроз в области производства аппаратного обеспечения составляет около 1 миллиарда долларов в день [1], что примерно в 10 раз превышает ущерба от пиратства в области ПО [1]. Кроме пиратства, появляются и новые виды угроз. Например, известны атаки на аппаратные реализации криптографических алгоритмов (т.н. Side-Channel Attacks [2]). В ходе этих атак осуществляется измерение токов, магнитных полей в некоторых частях схемы в моменты переключения значений, что позволяет получать значения секретных ключей. Также существует угроза внедрения аппаратных троянов, которые могут изменять функционирование устройства, снижать уровень его защиты, нарушать работоспособность и передавать секретную информацию из него. Поэтому сейчас огромное значение имеет разработка методов защиты от подобных угроз и борьбы с пиратством. Одним из таких методов является обфускация.

I. Лексическая обфускация

Обфускация – широко известная методика защиты исходных кодов программ от обратного проектирования. Основной целью обфускации является запутывание понимания функционирования программы. В идеале сложность и временные затраты на обратное проектирование должны оказаться близки к затратам на разработку системы с нуля. Обфускация очень часто используется совместно с другими методами защиты от пиратства, например, её применяют для сокрытия водяных знаков и отпечатков пальцев. Существует обширная классификация методов обфускации, разработанных для различных языков программирования [3]. Однако эти методы теряют свою актуальность в случае языка VHDL: схемы, полученные в результате синтеза исходных и обфусцированных описаний, идентичны. Поэтому применительно к VHDL выделяют две разновидности обфускации: *лексическую* и *функциональную* [4]. Лексическая затрагивает только уровень исходного описания.

Попробуем формализовать лексическую обфускацию для VHDL. Для этого представим исходный язык описания как:$< V >= \{T, S, D, K\}$, где T – множество терминалов, S – множество выражений, D – множество объявлений, K – множество комментариев. Схему цифрового устройства Sch можно описать так [4]:

$$Sch = \{IP, OP, B, L\},$$

где IP – множество входных портов, OP – множество выходных портов, B – множество функциональных блоков, из которых состоит схема устройства, L – множество проводящих линий, соединяющих внутренние блоки и порты. *Синтез* – процесс интерпретации исходного описания на языке V в схему.

$$DD(V) = DD(\{T, S, D, K\}) = \{IP, OP, B, L\}$$

Тогда *лексическое эквивалентное преобразование* – это замена одного фрагмента кода $V_1 = \{T, S, D, K\}$, результат синтеза которого:

$$DD(V_1) = DD(\{T, S, D, K\}) = \{IP, OP, B, L\}$$

другим фрагментом $V_2 = \{T', S', D', K'\}$, где $V_2 \neq V_1$, результат синтеза которого идентичен предыдущему:

$$DD(V_2) = DD(\{T', S', D', K'\}) = \{IP, OP, B, L\}.$$

Сложность описания можно представить как функцию, зависящую от внутренней структуры V: *C(V) = C({T,S,D,K})*.

Лексическое обфусцирующее преобразование – это лексическое эквивалентное преобразование, для которого дополнительно выполняется свойство: сложность результирующего фрагмента V_2 выше, чем сложность исходного V_1: $C(V_2) > C(V_1)$.

Лексическая обфускация – процесс применения лексических обфусцирующих преобразований к исходному описанию V с целью получения более сложного $V*$, но при сохранении неизменным результата синтеза.

Частными случаями лексических обфусцирующих преобразований могут быть следующие:

1) преобразования, удаляющие или добавляющие избыточные конструкции в описание:

$$V' = V \pm \Delta V;$$

$$DD(V') = DD(V \pm \Delta V) = Sch \pm \emptyset = Sch.$$

Примером таких преобразований могут быть внедрение комбинаций сигналов, минимизируемых при синтезе, удаление комментариев, разрушение форматирования, переименование идентификаторов.

2) преобразования, переупорядочивающие операции, не оказывая влияния на семантику:

$$V = \{p_1, ..., p_n\};$$

$$V' = permutation(p_1, ..., p_n);$$

$$DD(V) = DD(V') = Sch,$$

где p_i - параллельные операторы. Примером служат преобразования переупорядочивания параллельных выражений.

Суть функциональной обфускации в получении эквивалентной схемы[4]:
$DD(V) = Sch; DD(V') = Sch'; Sch \neq Sch'$;
но $F(Sch) = F(Sch')$;

Рассмотрим пример мультиплексора. Существует много способов описать его на VHDL. Некоторые приведены на рисунке 1.

```
1)  process (a, b, sel)
    begin
        if (sel = '1') then
            x <= a;
        else
            x <= b;
        end if;
    end process;

2)  process (a, b, sel)
    begin
        case sel is
            when '1' => x <= a;
            when others => x <= b;
        end case;
    end process;

3)  x <= a when sel = '1'
         else b;

4)  with sel select
        x <= a when '1',
        b when others;

5)  x <= (sel and a) or (not sel and b);

6)  process(a,b,sel)
        variable index : integer;
        variable temp : std_logic_vector(2 downto 0);
        variable rom : std_logic_vector (7 downto 0) := "11001010";
    begin
        temp := sel & a & b;
        index := conv_integer(temp);
        x <= rom(index);
    end process;
```

Рис. 1 – Способы описания мультиплексора

Первые 4 способа описывают поведение устройства. Они являются эквивалентными, однако трудно выделить среди них наиболее сложный. Поэтому в данном случае нельзя сказать, что замена одного из способов на другой в проектном описании приведёт к лексической обфускации. Самым низкоуровневым описанием будет описание в виде логических элементов (рис. 1.5). Такое описание визуально сложнее для восприятия. Это связано с раскрытием низкоуровневых деталей реализации и разрушением абстракции переключателя. По этой причине его можно считать не просто очередным эквивалентным описанием, а результатом лексической обфускации описания. Кроме того, мультиплексор можно представить в виде элемента памяти, адресуемого входными сигналами (рис 1.6). Это описание сложнее для понимания, чем предыдущие. Его можно считать примером лексической обфускации, т.к. в нём создаётся фиктивная абстракция – ROM и осуществляется её адресация.

Для оценки сложности исходных кодов используются различные метрики. Эти метрики показывают качество исходного кода, лёгкость сопровождения [5]. Эти же метрики принято использовать и для оценки эффективности обфусцирующих преобразований. Для эффективной оценки сложности нужно учесть специфику и особенности VHDL. Рассмотрим следующие метрики:

M1. число операторов;
M2. среднее число операторов на процесс;
M3. число процессов;
M4. число сигналов и переменных;
M5. сцепление процессов – связанность процессов по средством одинаковых сигналов в списке чувствительности;
M6. средний размер списка чувствительности процессов;
M7. число объявлений – число объявленных типов, сигналов, переменных и т.п. Для расчёта общей сложности описания используем формулу: $C(V) = \sum_{i=1}^{n} a_i * M_i$, где a_i - весовой коэффициент, выбранный экспертами для метрики, M_i - рассчитанное значение i-ой метрики.

В таблице 1 приведены результаты расчёта метрик для описаний при $a_i = 1$ для всех метрик M_i.

Таблица 1 - Результат оценки сложности

	Вариант					
Способ оценки	1	2	3	4	5	6
M1	4	4	3	3	5	8
M2	3	3	0	0	0	7
M3	1	1	0	0	0	1
M4	4	4	4	4	4	7
M5	1	1	0	0	0	1
M6	3	3	0	0	0	3
M7	0	0	0	0	0	3
$C(V)$	16	16	7	7	9	30

II. Выводы

Большие выразительные возможности и особенности языка VHDL создают условия для разработки новых методов обфускации, они же приводят к необходимости поиска способов оценки сложности преобразований.

1. Hardware Security Mechanisms for Authentication and Trust [Электронный ресурс] / Электронные данные. – Режим доступа: http://sciencestage.com/v/35221. – Дата доступа: 12.09.2013.
2. Majzoobi, M. Introduction to hardware security and trust / M. Majzoobi, F. Koushanfar, M. Potkonjak. – New York: Springer, 2011. – 427 p.
3. Collberg, C. A Taxonomy of Obfuscating Transformations / C. Collberg, C. Thomborson, D. Low – Auckland: Department of Computer Science, 1997.
4. Иванюк, А. А. Проектирование встраиваемых цифровых устройств и систем : монография / А. А. Иванюк. – Минск : Бестпринт, 2012. – 337 с.
5. Software Complexity Measurement [Электронный ресурс]. – Электронные данные. – Режим доступа: Kearney.pdf.