СИНТЕЗ VHDL КОДА ДЛЯ ПРОЕКТИРОВАНИЯ ГЕНЕРАТОРОВ ПСЕВДОСЛУЧАЙНЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ НА КЛЕТОЧНЫХ АВТОМАТАХ

Д. Е. Храбров, И. А. Мурашко
Кафедра «Информационные технологии», Гомельский государственный технический университет имени П.О. Сухого
Гомель, Республика Беларусь
E-mail: science@dexp.in, iamurashko@tut.by

Предложен алгоритм автоматизированного синтеза VHDL кода для проектирования генераторов псевдослучайных последовательностей на клеточных автоматах.

Введение

Генератор псевдослучайных тестовых воздействий является одним из наиболее важных элементов встроенного самотестирования (англ. Built-in Self-test, BIST) [1]. Существует множество методов генерирования псевдослучайных последовательностей. В этой работе рассматриваются генераторы на клеточных автоматах [2,3].

Целью работы является синтез программного средства для автоматизации проектирования генераторов псевдослучайных последовательностей. Программное средство должно уметь получать конфигурации клеточного автомата, который может генерировать последовательность максимальной длины. Этой проблеме и посвящена данная работа.

I. Генерация кода по конфигурации клеточного автомата

Эвристический алгоритм получения конфигурации клеточных автоматов для создания генераторов псевдослучайных последовательностей описан в [4], который в свою очередь базируется на [5]. В основу положены три основных метода синтеза: синтез по Каттеллу, эвристический синтез и метод Монте-Карло. Все методы описаны в [4], основа эвристического синтеза подробнее показана в [6].

Имея конфигурацию клеточного автомата можно однозначно генерировать его VHDL код. Упрощённая блок-схема генерирования VHDL кода показана на рисунке 2. Код подключения библиотек, конфигурации и описания компонентов общий для любых генераторов на клеточных автоматах. Описание триггеров и соединительных линий выводится в цикле от 1 до количества триггеров, в описании компонентов отличается только номер. Далее для VHDL кода выводятся описания двухвходового и трёхвходового элемента XOR. Следующим этапом описываются соединения.

Результатом работы генератора являются значения триггеров на каждой итерации. Выходные значения ($TRiOUT$) собираются с линий связи, называемых $DUMMY$ ($TRiOUT\_DUMMY$). Триггеры имеют 3 порта: C, D и Q. Вход синхронизации каждого триггера C соединён с линией связи CLK. На вход данных D каждого триггера поступает значение с предстоящего элемента XOR.

Далее в VHDL коде идёт описание элементов XOR, через которые и происходит соединение триггеров между собой. При необходимости, элемент XOR может иметь тип $XOR3$ и иметь 3 входа, или может отсутствовать совсем. В таком случае ко входу триггера будет напрямую подключен выход предыдущего триггера.

II. Реализация в Xilinx ISE

Разработанное программное средство принимает на вход единственное значение – размерность результирующего генератора. Далее задаётся уточняющий вопрос, генератор с какими именно характеристиками нужен. И на выходе получается VHDL описание генератора, которое можно использовать в Xilinx ISE.

VHDL-код поведенческого уровня занимает порядка 50 строк кода. Этот же автомат, но реализованный на структурном уровне, занимает 165 строк кода.

Реализация LFSR может быть выполнена в разных вариантах. Например, возьмём характеристический полином $x^7 \oplus x^5 \oplus x^3 \oplus x \oplus 1$. Пример одной из реализаций приведен на рисунке 1. Данная реализация требует 3 двухвходовых элемента XOR.

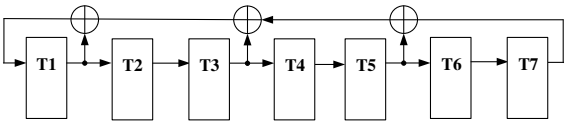

Рис. 1 – LFSR $x^7 \oplus x^5 \oplus x^3 \oplus x \oplus 1$, реализация 1

Ещё одна возможная реализация данного LFSR – использование одного, но четырёхвходового элемента XOR. Поскольку часто реализации подобных элементов логики реализованы на таблицах поиска (англ. LUT – lookup table), то та-

кая реализация возможна и элемент будет иметь единичную тактовую задержку.

Моделирование проводилось в Xilinx ISE 14.2, ПЛИС Xilinx Spartan-3 XC3S200. Кроме того, устройства так же были реализованы на отладочной плате Digilent Spartan-3 Board. При моделировании в Xilinx ISE при наличии только одного элемента *XOR* максимальная частота устройства была 313 МГц. Если же два элемента *XOR* в генераторе стояли последовательно, то частота падала до 210 МГц.

Рассмотрим реализации на клеточных автоматах и LFSR, имеющие характеристический полином $x^{13} \oplus x^{12} \oplus x^{11} \oplus x^9 \oplus x^8 \oplus x^7 \oplus x^6 \oplus x^5 \oplus x^4 \oplus x^3 \oplus x^2 \oplus x \oplus 1$. Реализация на клеточных автоматах имеет вектор правил [5557757757577] и работает на максимальной частоте 313 МГц, так так последовательно расположенных элементов *XOR* в такой реализации нет. Худшая реализация на LFSR имеет 11 последовательно стоящих *XOR* и максимальную частоту 60 МГц. Перераспределение элементов *XOR* так, чтобы максимальное количество последовательно соединённых элементов *XOR* было меньше 5, поднимает максимальную частоту до 140 Мгц. Xilinx ISE имеет в стандартной библиотеке элементы *XOR* на максимум 9 входов. То есть для реализации представленного генератора нужно минимум два элемента *XOR*. Это поднимает максимальную частоту до 170 МГц, что медленнее реализации на клеточных автоматах.

При значительном увеличении размерности клеточного автомата максимальная частота генератора уменьшается не настолько значительно. Например, клеточный автомат на правилах (90, 240) и размерностью 503 триггера имеет максимальную частоту 220 МГц.

Заключение

В работе рассмотрена автоматизация синтеза генераторов псевдослучайных последовательностей на клеточных автоматах. Предложена методика получения конфигураций клеточных автоматов по единственному параметру – размерности. Разработанное приложение позволяет генерировать VHDL код, который можно использовать как в проектировании цифровых устройств, включая программируемые логические интегральные схемы и микроконтроллеры, так и в программной реализации.

1. Agrawal, V. Essentials of Electronic Testing for Digital, Memory, and Mixed-Signal VLSI Circuits / V. Agrawal, M. Bushnell. – Springer, 2000. – P. 712.
2. Hortensius, P. D. Parallel random number generation for VLSI systems using cellular automata / P. D. Hortensius // IEEE Transactions on Computers. – 1989. – Vol. 38 (10). – P. 1466–1473.
3. del Reya, A. Martin. Reversibility of linear cellular automata / A. Martin del Reya, G. Rodriguez Sanchez // Applied Mathematics and Computation. – 2011. – Vol. 217. – P. 8360–8366.
4. Мурашко, И.А. Эвристический алгоритм проектирования генераторов псевдо-случайных последовательностей на клеточных автоматах / Д.Е. Храбров, И.А. Мураш-ко // Материалы международной научной конференции «ИТС 2015», Минск, 28 октяб-ря 2015 г. / Белорусский государственный университет информатики и радиоэлектро-ники – Минск, 2014 – С.156–157.
5. Храбров, Д. Е. Применение клеточных автоматов с расширенным набором правил для генерирования псевдослучайных тестовых последовательностей / Д. Е. Храбров, И. А. Мурашко // Проблемы физики, математики и техники. – 2014. – № 1 (18). – С. 98–104.
6. Мурашко, И. А. Применение клеточных автоматов с расширенным набором правил для генерирования псевдослучайных тестовых последовательностей / И. А. Мурашко, Д. Е. Храбров // Вестник московского государственного университета приборостроения и информати-ки. – 2013. – № 47. – С. 78–93.

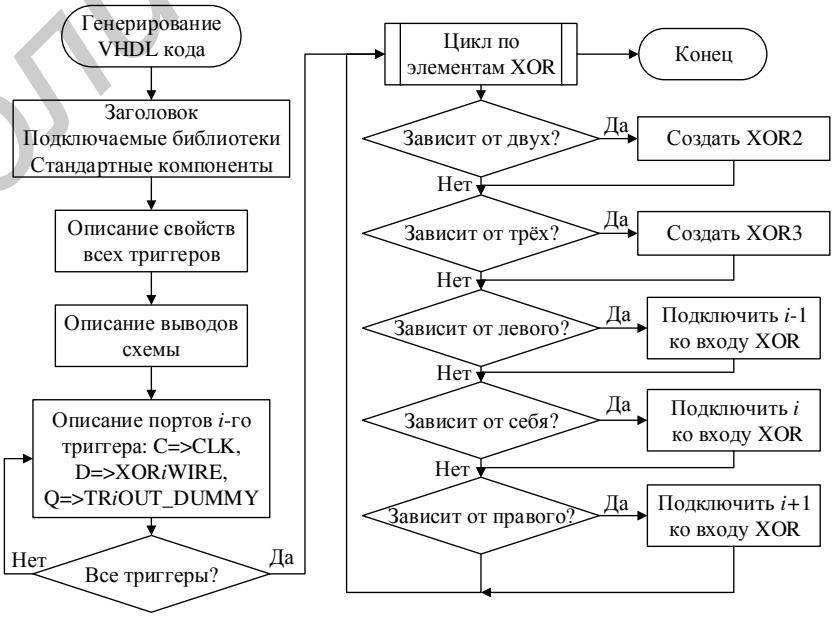

Рис. 2 – Блок-схема генерирования VHDL кода