ФУНКЦИОНАЛЬНОЕ МОДЕЛИРОВАНИЕ ЦИФРОВЫХ СХЕМ С ПОМОЩЬЮ ПРОГРАММ ТЕСТИРОВАНИЯ НА ЯЗЫКЕ С++

Д. Е. Боркивец, А. И. Егоров, А. Г. Кузякин

ФГУП «РФЯЦ-ВНИИЭФ», г. Саров Нижегородской обл.

Введение

Рост сложности задач, решаемых аппаратурой автоматики, неизбежно ведет к увеличению ее функциональной сложности, обуславливающей усложнение схемно-конструкторских решений. Это, в свою очередь, прямо отражается на габаритно-массовых характеристиках (ГМХ) приборов автоматики, порой определяющих применимость их на том или ином типе изделий.

Другой, не менее важной проблемой является обеспечение требуемого уровня стойкости аппаратуры автоматики, определяемого условиями ее применения. Реализация этого требования также приводит к усложнению функционала и, как правило, – к применению аппаратно избыточных схемотехнических решений. Более того, требование стойкости аппаратуры к воздействию внешних дестабилизирующих факторов резко ограничивает номенклатуру пригодных к применению в ней электронных компонентов.

Создание элементной базы нового поколения (высокой степени интеграции с субмикронными проектными нормами) становится основным и критически важным условием для конкурентоспособности электронной аппаратуры.

Разработка полупроводниковой элементной базы, в том числе сверх больших интегральных схем (СБИС) на основе базового матричного кристалла (БМК), для современной электронной аппаратуры является сложным и наукоемким процессом, требующим применения самых современных средств и методов проектирования.

Современные методы проектирования основываются на логическом синтезе структурной схемы из функциональных описаний, разработанных на специализированных языках программирования.

В процессе разработки функционального описания требуется разработка программы тестирования, которая позволяет проверить функционал описания схемы и отслеживать правильность вносимых поправок. Процесс верификации занимает одно из ключевых этапов разработке схемы.

Стандартный маршрут проектирования

Стандартный маршрут проектирования (рис. 1) начинается с предоставления технического задания, которое является исходными данными для разработки функционального описания и тестовой программы. В процессе и после разработки, производится моделирование, которое необходимо для проверки наличия требуемой функции в описании, после чего проверенную модель синтезируют и проверяют снова, чтобы убедиться, что синтез прошел правильно. После этого структурную электрическую схему передают изготовителю [1].

Стоит различать первый этап тестирования от второго. Моделирование на этапе проверки структурной схемы проводится один раз и дает информацию только о соответствии логики до синтеза и после. Моделирование на этапе отладки функционала дает информацию о соответствии результатов функции исходным данным, дает возможность наблюдать изменения, внесенные в описание, проверку влияния внесенного изменения на схему в целом. Оно выполняется итерационно, т. к. с увеличением функционального описания приходится дополнять тест для него и проверять снова, что влечет за собой большие временные затраты на проверку.

Функциональное описание и тестовая программа разрабатываются на языках описания цифровой аппаратуры.

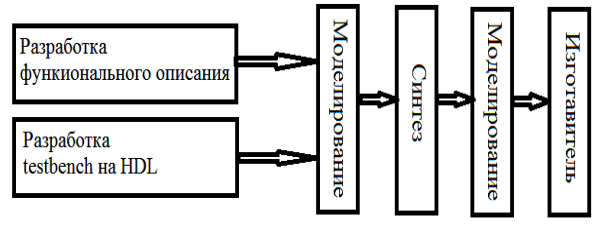

Рис. 1. Стандартный маршрут проектирования

Цели и задачи

Целью данной работы является оптимизация этапа отладки функционального описания разрабатываемой схемы.

Для этого поставлены задачи сокращения времени моделирования и получения доступа к программным средствам на языке С++.

В докладе рассмотрены возможности программного средства, позволяющего проводить функциональное моделирование цифровых схем на языке описания аппаратуры с использованием программ тестирования на языке С++, который позволил упростить процесс и сократить время разработки теста.

Языки описания цифровой аппаратуры hardware description language (HDL)

Развитие современной элементной базы тесно связано с развитием методик проектирования сложных цифровых устройств. Разработка современных устройств невозможна без применения систем автоматизированного проектирования (САПР), которые выполняют значительную часть рутинной работы. Современные САПР обладают достаточным «интеллектом» для того, чтобы вместе с человеком участвовать в разработке схемы. Однако реализация открывающихся возможностей требует использования новых, современных методов проектирования.

На сегодняшний день наиболее эффективными считаются методы, использующие текстовые описания проектов вместо традиционных графических редакторов схем. В этом случае проект представляет собой текст, сохраняемый в файле или нескольких файлах, которые в совокупности определяют представления разработчика об устройстве и используются на всех этапах проектирования, в том числе при моделировании устройства и синтезе его принципиальной схемы. Проектирование цифровых схем с использованием языков описания аппаратуры позволяет полностью отказаться от графических редакторов принципиальных схем, при этом повысить скорость работы, надежность и удобство отладки. Тексты описания в большинстве языков проектирования дискретных устройств по составу синтаксических конструкций очень схожи с традиционными языками программирования. Поэтому часто такое текстовое описание называют программой на языке проектирования, или, коротко, HDL-программой, а конструкции, описывающие формирование принципиальной электрической схемы, называют операторами [2].

Другими словами, HDL-программа – это знаковая модель дискретного устройства. Такая модель может быть описана с различной степенью конкретизации. Давая возможность достаточно детально описывать структуру проекта, часто, вплоть до отдельных вентилей, языковое описание также позволяет описывать устройство на уровне алгоритмов его работы, возлагая при этом синтез самой структуры на систему автоматизированного проектирования (САПР). В этом заключается мощное превосходство языков описания аппаратуры перед графическими редакторами схем, где проектировщик вынужден полностью рисовать структуру устройства с максимальной степенью детализации [3].

Существует два наиболее распространенных универсальных языка описания аппаратуры: VHDL, который по своим синтаксическим конструкциям напоминает Паскаль, и Verilog, имеющий некоторое сходство с языком Си. Это сходство весьма условно, поскольку и VHDL, и Verilog являются языками структурного описания электрических схем, а не языками программирования.

Программа тестирования Verilator

Есть множество программ, позволяющих моделировать алгоритмы на языке Verilog. Одной из таких программ является Verilator. Программа существует уже более десяти лет и активно поддерживается.

В отличие от большинства подобных программ, Verilator не является интерпретатором языка Verilog. Это конвертор из Verilog в C++. Полученный файл затем компилируется компилятором gcc и исполняется, как обычная компьютерная программа.

Особенности Verilator:

- позволяет транслировать код на языке verilog в C++-модель;
- имеет открытий исходный код;
- откомпилированный код сравним по скорости моделирования с коммерческими пакетами (рис. 2);
- позволяет генерировать временные диаграммы сигналов и измерять покрытие RTL-кода.

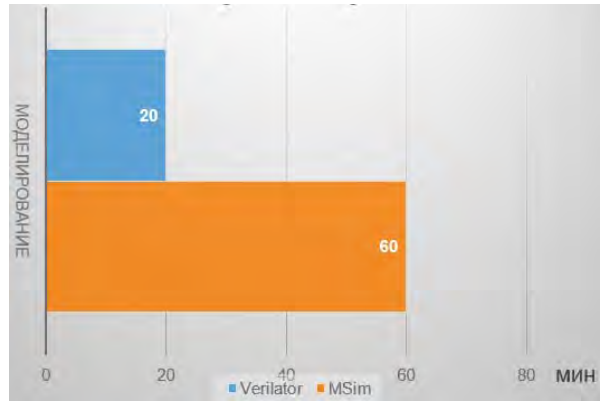

Рис. 2. Моделирование 32-битного SPARC процессора LEON 3

В результате Verilator поддерживает не все возможности Verilog. Поддерживаются синтезируемые конструкции, а также некоторые системные функции, например $display (). Впрочем, поддержка Verilog (а также System Verilog и System C) достаточно хорошая [4].

Недостатки:

- неполная поддержка стандарта Verilog, не синтезируемые конструкции не поддерживаются;
- Verilator игнорирует секции описания временных задержек;
- не моделирует ячейки с тремя состояниями.

Преимущества:

- высокая скорость моделирования;
- возможность написания теста на языке C++.

Тестируемая модель

Отработка маршрута проектирования с использованием Verilator производилась на основе проекта, который представляет собой арифметико-логическое устройство с интерфейсом передачи данных Serial Peripheral Interface (SPI) (рис. 3).

Арифметико-логическое устройство разработано в качестве оконечного устройства для отработки реализации SPI и выполняет 4 функции: логическое и, логическое или, исключающее или, отправка 0. Данные на устройство принимаются последовательно. Первым на вход поступает код команды, вторым и третьим операнды, а во время отправки 0 на выход устройства поступает результат.

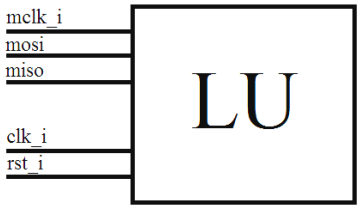

Рис. 3. Графическое представление тестируемой модели: mclk_i, mosi, miso используются для передачи данных по SPI; clk_i – тактовый сигнал; rst_i – сброс

Интерфейс передачи данных SPI

SPI представляет собой четырехпроводную синхронную шину, предназначенную для последовательного обмена данными между микросхемами. Интерфейс в настоящий момент используется всеми производителями. Данный интерфейс отличают простота использования и реализации, высокая скорость обмена и малая дальность действия.

При любом обмене данными по SPI одно из устройств является ведущим (Master´ом), а другое ведомым (Slave´ом). Ведущий переводит периферийное устройство в активное состояние и формирует тактовый сигнал и данные. В ответ ведомое устройство передает ведущему свои данные. Передача данных в обе стороны происходит синхронно с тактовым сигналом (рис. 4) [4].

Физически SPI реализуется на основе сдвигового регистра, который выполняет и функцию передатчика, и функцию приемника.

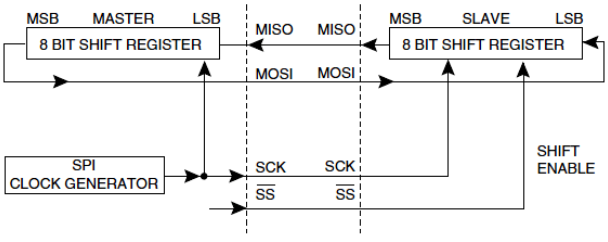

Рис. 4 . Принцип обмена данными по SPI

Сигналы, используемые данным интерфейсом, имеют следующее назначение:

– MOSI – Master Output/Slave Input (выход ведущего / вход ведомого) служит для передачи данных от ведущего устройства к ведомому;

– MISO – Master Input/Slave Output (вход ведущего/выход ведомого)служит для передачи данных от ведомого устройства к ведущему;

– SLK – Serial Clock (сигнал синхронизации) служит для передачи тактового сигнала всем ведомым устройствам;

– SS – Slave Select (выбор ведомого) служит для выбора ведомого устройства (если ведомых несколько).

Тестовое окружение

Представленное логическое устройство включает в себя блок spi-slave(ведомый). Для тестирования требуется возможность правильной передачи данных по стандарту spi. Для этого был разработан и подключен к логическому устройству блок spi-master (рис. 5).

Интерфейс LU имеет:

- тактовый сигнал (clk_i);
- сброс (rst_i);
- разрешение передачи (tr_i);
- разрешение записи (we_i);
- шину входных данных (di_i);
- флаг завершения передачи (re_o);
- шину выходных данный (do_o);
- специализированные выводы для общения с ведомым устройством (mklk_i, mosi, miso).

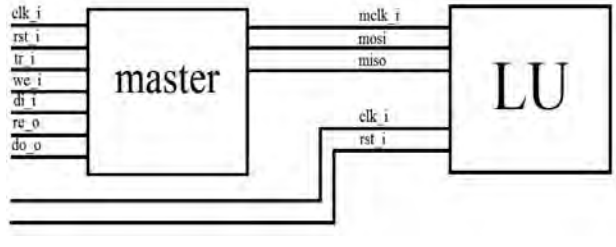

Рис. 5. Схема соединения master SPI с логическим устройством

Тестовая программа

Алгоритм теста, написанного на языке C++, должен соответствовать алгоритму теста, написанного на HDL.

В нашем случае функциональное тестирование реализовывалось прямым перебором всех возможных операндов при всех возможных командах устройства, с последующим сравнением результатов с аналогичными операциями, реализованными средствами языка C++. Способ отображения результатов также программируется в тестовой программе, в нашем случае производится вывод на экран результатов сравнения выходов схемы с эталонными значениями и формирование файла результатов моделирования в виде срезов временной диаграммы.

Тестирование

Входными данными для работы программы являются функциональное описание, написанное на языке Verilog и тестовая программа, написанная на языке C++.

Следующей командой запускается Verilator, который выполняет транслирование описания на язык C++ и подключение теста к модели:

`Verilator –cc <функциональное описание> -v <verilog библиотека> -exe <программа тестирования> –trace`

Следующей командой выполняется компилирование полученного проекта с помощью программы gcc:

`make –j –Cobj_dir –f V<название функционального описания>.mk V<название функционального описания>`

Запуск полученного исполняемого файла запускает тест, результат которого отображается способом, описанном в тесте. Маршрут выполнения тестирования с использованием Verilator представлен на рис. 6. Файлы результата моделирования можно просмотреть в программе gtkwave, входящей в стандартный набор пакетов UNIX-систем.

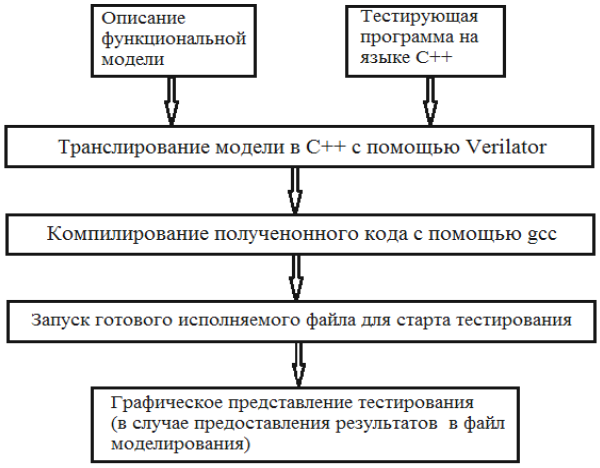

Рис. 6. Маршрут верификации с использованием Verilator

Сравнительный анализ

Для проведения сравнительного анализа необходимо определить время моделирования теста на языке C++ и языке Verilog. Время выполнения тестирования на языке C++ составило 19 сек, на языке Verilog составило 85 сек. Таким образом скорость моделирования тестовой программы на языке C++ в 4,5 раза выше чем у аналогичной программы на языке Verilog.

Преимущества и недостатки использования Verilator

В процессе изучения и работы с использованием Verilator были выявлены следующие плюсы:

– более высокая скорость моделирования, что значительно экономит время тестирования, т. к. оно выполняется итерационно;

– написание теста на C++ позволяет расширить возможности тестирования, в отличиеот более узконаправленного Verilog. Например, C++ допускает передачу параметром в функцию по ссылке и дальнейшее объединение всех подпрограмм в библиотеку. Для написания теста на HDL требуется специализированное программное обеспечение, лицензии для которого стоит приличных денег, требуется подготовленный программист, изучивший HDL язык программирования. Verilator входит в стандартный набор пакетов в операционных системах UNIX. Программисту для написания теста достаточно обладать базовыми навыками программирования на C++, а также изучить некоторые нюансы, связанные с программированием с использованием Verilator и моделированием в целом.

Существующие минусы Verilator не отразились на работе.

Отсутствие поддержки не синтезируемого Verilog не повлияло, т. к., разрабатывая HDL описания, мы не используем не синтезируемые конструкции. Игнорирование секции описания временных задержек не критично, т. к. мы используем Veritator на этапе отладки функционала,

Заключение

В ходе выполнения работы достигнуты поставленные цели, а именно:

– разработано функциональное описание арифметически логического устройства:

– реализована модель интерфейса передачи данных SPI;

– разработана тестирующая программа на языке HDL и C++;

– отработан процесс моделирования на Verilator.

По результатам проделанной работы принято решение включить данное программное средство в маршрут проектирования СБИС на основе БМК. Ожидается, что это приведет к повышению эффективности работы в следующих проектах.

Литература

1. Режим доступа: http://www.compitech.ru/ [Электронный ресурс].

2. Режим доступа: http://window.edu.ru/ [Электронный ресурс].

3. Режим доступа: http://citforum.ru/ [Электронный ресурс].

4. Режим доступа: http://www.veripool.org/ [Электронный ресурс].

Режим доступа: http://chipenable.ru/ [Электронный ресурс].