Посвящаю моей жене, Татьяне Серафимовне

Предисловие

Технический прогресс неминуемо ставит новые вызовы перед разработчиками цифровой аппаратуры, при этом усложняются технические системы и решаемые ими задачи, что требует автоматизации процесса разработки аппаратуры на более высоких уровнях проектирования. В связи с этим появляются новые проблемы при разработке аппаратных средств, особенно сложных систем с большим числом компонентов и разнообразными соединениями между компонентами. Важную роль в современных цифровых системах также играют транзакции, которые поддерживаются аппаратными блоками, обеспечивающими обмен данными по стандартным или фирменным протоколам передачи данных.

С другой стороны, все эти сложные цифровые блоки и системы необходимо тестировать, желательно на всех уровнях проектирования. Одним из способов верификации устройств и систем является событийное моделирование, выполняемое на компьютере с помощью программных средств. Отметим также, что возрастание сложности цифровых систем сопровождается ужесточением требований к срокам разработки и повышению надежности проектов.

Практика инженерного проектирования показывает, что возможностей языка описания аппаратуры Verilog [1] становится уже недостаточно для разработки современных цифровых систем. В результате активной деятельности различных инициативных групп разработчиков были предложены усовершенствования языка Verilog, что привело к появлению нового языка проектирования цифровой аппаратуры SystemVerilog.

Язык SystemVerilog полностью наследует язык Verilog и предоставляет новые возможности для разработки больших и сложных проектов на самых верхних уровнях проектирования: системном, абстрактном и уровне транзакций. Кроме того, язык SystemVerilog включает новые языковые конструкции верификации проектов, обеспечивающие новые возможности для описания тестового окружения сложного проекта. В результате получился язык с очень широкими и разнообразными возможностями, которые трудно в деталях описать в одной книге. В предыдущей книге [2] были рассмотрены конструкции языка SystemVerilog для синтеза, используемые при аппаратной реализации проекта. В данной книге описываются конструкции языка SystemVerilog, которые предназначены для моделирования и используются при формальной верификации проекта.

Разработка больших и сложных проектов требует значительно больших усилий при верификации (проверке) проекта. Практика показывает, что в успешных компаниях, которые создают сложные цифровые системы в указанные сроки, на одного разработчика аппаратуры приходится два инженера по верификации. Использование языка SystemVerilog в моделировании позволяет автоматизировать процесс формальной верификации проекта. В настоящее время на основе языка SystemVerilog разработано ряд методик верификации, которые позволяют значительно сократить время и усилия, затрачиваемые на верификацию проектов, а также повысить надежность проектов.

Для лучшего понимания излагаемого материала желательно, чтобы читатель был знаком с языком Verilog [1] или языком SystemVerilog для синтеза [2]. Даже если читатель не знаком ни с одним из указанных языков, ничего страшного. Однако в этом случае книгу придется прочитать несколько раз.

Язык SystemVerilog слишком сложный, чтобы его можно было изложить последовательно. В связи с этим в книге часто упоминаются еще не рассмотренные конструкции языка. Поэтому начинающим пользователям языка SystemVerilog книгу рекомендуется читать несколько раз: первый раз для общего знакомства с языком, а затем изучать отдельные главы и разделы по мере необходимости.

Материал книги не ориентирован на какую-либо систему проектирования или элементную базу. Книга может быть интересна как разработчикам цифровых систем на FPGA (field programmable gate array), так и на ASIC (application specific integrated circuit).

Книга включает 19 глав и одно приложение.

Глава 1 является вводной. В ней описываются основные понятия, которые относятся к моделированию цифровых систем с использованием языка SystemVerilog. Приводится краткая история языка SystemVerilog, описываются основные проблемы верификации сложных проектов, определяется, что такое *тестовый стенд* (testbench). Здесь также дается представление фундаментальных понятий, на которых базируется формальная верификация: утверждения, функциональное покрытие и рандомизация. Описывается, как выполняется моделирование проекта в языке SystemVerilog. Кроме того, приводятся системные задачи языка SystemVerilog для отображения информации.

В главе 2 рассматриваются типы данных и все, что с ними связано. Здесь описываются основные типы: сети и переменные, а также агрегированные типы: массивы (в том числе динамические массивы, ассоциативные массивы и очереди), структуры и объединения. Здесь же рассматриваются перечисляемые и пользовательские типы, а также константы, приведение типов и пакеты, с помощью которых данные проекта можно объявить только один раз, а затем эти объявления использовать в разных модулях. Отдельно рассматривается вопрос способа хранения данных с точки зрения объема используемой памяти и скорости выполнения моделирования.

Выражения, операции и операторы назначения представлены в главе 3.

Выражения и операции составляют основу любого языка программирования или проектирования. В языке SystemVerilog операции языка Verilog расширены операциями языка программирования C, операцией членства множества, операциями распределения и потоковыми операциями. Новыми в языке SystemVerilog также являются агрегатные выражения, в которых можно использовать упакованные структуры и массивы. Отдельно рассматриваются операторы назначения, которые совпадают с аналогичными операторами языка Verilog. Язык SystemVerilog добавляет новые операторы назначения с выполнением отдельных операций, как в языке C.

Глава 4 посвящена процессам и потокам. Процесс в языке SystemVerilog — это общее понятие совокупности действий по обработке данных. Процессы создаются процедурами или процедурными операторами. Язык SystemVerilog расширяет структурные процедуры и добавляет тонкое управление процессами. Очень близкими по сути к процессам являются *потоки* (threads). Потоки представляют собой легкие процессы, которые могут совместно использовать общий код и память и потреблять гораздо меньше ресурсов, чем обычный процесс. Так же как и процесс, поток представляет собой совокупность некоторых действий по обработке данных. Если понятие процесса используется как в синтезе, так и в моделировании, то термин «поток» чаще применяется в моделировании при описании кода в виде программ.

В главе 5 описываются процедурные операторы. Язык SystemVerilog вводит квалификаторы уникальности и приоритета в условный оператор и оператор выбора, расширен также список операторов цикла и операторов перехода.

В главе 6 описываются модули. Модуль является главной структурной единицей языков Verilog и SystemVerilog. Новым в языке SystemVerilog является удобство передачи сигналов через порты модулей, когда имена сигналов совпадают с именами портов. Кроме переменных и сетей язык SystemVerilog позволяет передавать через порты модулей массивы, структуры, объединения и интерфейсы. Дополнительно переменные могут передаваться по ссылке. Новым также является то, что в качестве параметров модулей могут выступать типы данных.

В главе 7 рассматриваются задачи и функции языка SystemVerilog. Язык SystemVerilog снимает большинство ограничений для задач и функций языка Verilog, что приравнивает использование задач и функций к языкам программирования, а также упрощает разработку больших и сложных проектов.

В главе 8 представлены интерфейсы. Интерфейсы — это новый механизм для описания соединений между компонентами большого проекта. Интерфейс разрешает группу имен портов заменить одним именем, позволяя разработчику сосредоточиться на функциональности системы, а не на проверке правильности подсоединения сигналов к портам модулей. Интерфейсы могут иметь собственную функциональность, соответствующую протоколу передачи данных. Язык SystemVerilog также позволяет создавать

древовидную структуру интерфейсов для соединения компонентов сложной системы.

Глава 9 посвящена объектно-ориентированному программированию (ООП) языка SystemVerilog, в ней рассматриваются классы. Дело в том, что тестовые стенды обычно описываются в виде программ. Поэтому вполне логичным является использование ООП при написании программ. С помощью классов описывается окружение *тестируемого проекта* (design under test — DUT), *тестовый стенд* (testbench), а также элементы тестового стенда. Особенно успешно классы используются для рандомизации проекта.

В главе 10 рассматривается синхронизация моделирования и блоки синхронизации. Взаимодействие тестового стенда и DUT во времени представляет собой достаточно сложный процесс, который может породить *гонки сигналов* (races). Для устранения гонок между сигналами тестового стенда и DUT в языке SystemVerilog служат блоки синхронизации. Блок синхронизации определяет сигналы тестового стенда, синхронизированные с заданным тактовым сигналом (синхросигналом). Это позволяет переходы во времени определить в терминах циклов и транзакций, а также отделить детали синхронизации от элементов тестового стенда. В результате становится возможным описание синхронных событий без явного использования синхросигналов или указания времени.

В главе 11 рассматриваются программы языка SystemVerilog, с помощью которых описываются тестовые стенды. Программы языка SystemVerilog мало чем отличаются от программ языков программирования, однако акцент делается на взаимодействии программ с модулями, задачами и функциями. Здесь освещается ряд вопросов, нацеленных на устранение гонок сигналов тестового стенда, а также завершение программ.

Глава 12 посвящена утверждениям. Язык утверждений SystemVerilog (assertion SystemVerilog — ASV) является отдельным языком в рамках языка SystemVerilog. Утверждения составляют основу методики формальной верификации ABV (assertion based verification — методика верификации на основе утверждений). Утверждения можно рассматривать как модели спецификации (поведения) проекта. С помощью утверждений выполняется проверка совпадения (соответствие), определение допущений на входах DUT, ограничение проверяемых значений, а также покрытие спецификации проекта.

Функциональное покрытие рассматривается в главе 13. Функциональное покрытие основано на спецификации проекта и измеряет покрытие функций проекта. В языке SystemVerilog функциональное покрытие реализуется с помощью оператора **cover** и языка функционального покрытия FC (Functional Coverage). Язык FC также является отдельным языком в рамках языка SystemVerilog.

Глава 14 посвящена рандомизации (randomization). Рандомизация — это процесс создания случайных данных для моделирования. Рандомизация позволяет автоматизировать выполнение моделирования, а также находить ошибки там, где разработчик никогда их не ожидал. Данные для создания случайных тестов не могут быть абсолютно случайными, обычно требуются

определенные ограничения. Поэтому рандомизация основана на *ограниченных случайных тестах* (constrained-random tests — CRTs). Рандомизация в языке SystemVerilog имеет много специфических особенностей и также может рассматриваться как некоторое подмножество языка SystemVerilog.

В главе 15 рассматриваются *контролеры* (checkers). Контролеры позволяют сгруппировать несколько утверждений вместе в более крупный блок, что обеспечивает модульность и возможность повторного использования утверждений и свойств. Особенностью контроллеров является то, что они позволяют объединить вместе утверждения для моделирования и процедурный код. Благодаря этому контролеры могут служить библиотечными единицами верификации и строительными блоками для создания абстрактных моделей, используемых при формальной верификации.

Глава 16 посвящена синхронизации процессов и связи между процессами. В ней рассматриваются вопросы управления процессами и потоками с помощью *событий* (events), *семафоров* (semaphores) и *почтовых ящиков* (mailboxes).

В главе 17 рассматриваются временные характеристики проектов, реализуемых на микросхемах ASIC. Функционально правильный проект может не работать без учета временны́х характеристик, таких как задержка распространения, ширина импульса, время установки и время удержания. Язык SystemVerilog предоставляет возможности для определения временны́х характеристик проекта. К таким средствам относятся *задержки* (delays), *блоки спецификации* (specify blocks), *проверки синхронизации* (timing checks) и *обратное аннотирование* (back annotation), которое предполагает использование файлов *стандартного формата задержек* (Standard Delay Format — SDF).

Защита исходного кода проекта от несанкционированного доступа рассматривается в главе 18. В языке SystemVerilog эта проблема решается с помощью *защитных конвертов* (protected envelopes) и директив прагмы **protect**.

В главе 19 приводится структура многоуровневого тестового стенда с потоками и межпроцессными связями.

В приложении А перечислены ключевые слова языка SystemVerilog.

Книга в первую очередь предназначена для разработчиков цифровых систем и студентов соответствующих специальностей технических университетов. Материал книги может быть использован преподавателями для чтения лекций, проведения лабораторных и практических занятий. Научным работникам книга может быть полезна при разработке новых методик верификации сложных цифровых систем. Книга содержит много детальных описаний, поэтому может использоваться в качестве справочника по языку SystemVerilog.

Написание данной книги частично финансировалось грантом Белостокского технологического университета из ресурсов для научных исследований Министерством науки и высшего образования Республики Польша.