ТЕСТОВАЯ СИСТЕМА ДЛЯ ФУНКЦИОНАЛЬНОЙ ВЕРИФИКАЦИИ ДИСКРЕТНЫХ УСТРОЙСТВ С ПАРАЛЛЕЛИЗМОМ ПОВЕДЕНИЯ

Черемисинова Л. Д., Черемисинов Д. И.
Объединённый институт проблем информатики Национальной академии наук Беларуси
Минск, Республика Беларусь
E-mail: {cld, cher}@newman.bas-net.by

Рассматривается задача, возникающая при моделировании аппаратных реализаций дискретных систем с параллелизмом поведения. Предлагается метод построения тестовой системы для генерации модельных тестов на основе спецификации на проектирование. В основе метода лежит построение TLM моделей описаний на языке параллельных алгоритмов управления.

Введение

В настоящее время сложность аппаратуры возросла до такого уровня, что требуется развитие новых методов и средств функционального тестирования, которые автоматически генерируют тестовые последовательности и оценивают правильность поведения аппаратной реализации системы. При этом, генерируемые тесты не должны быть жестко привязаны к этой реализации. В настоящее время наиболее развитыми подходами и средствами разработки функциональных тестов являются технологии AVM (Advanced Verification Methodology) [1] компании Mentor Graphics и OVM (Open Verification Methodology) [2] Mentor Graphics и Cadence Design Systems на основе языков SystemVerilog или SystemC. Можно упомянуть также академическую разработку – технологию UniTESK (Unified TEsting and Specification tool Kit) [3] Института системного программирования РАН. В ней основной упор делается на автоматизацию процесса генерации тестовых последовательностей на основе автоматных моделей. Тестовые воздействия находятся в процессе обхода графа состояний конечного автомата, моделирующего тестируемую систему.

В настоящей работе так же, как и в UniTESK, рассматривается задача генерации тестовых последовательностей для функциональной верификации аппаратуры систем управления. Отличие состоит в том, что рассматриваются системы управления с параллелизмом поведения. При разработке тестовой системы используется объектно-ориентированный подход на основе моделирования на уровне транзакций – TLM (Transaction-Level Modeling) [4].

I. Верификация на основе TLM модели

TLM представляет собой подход к моделированию обмена данными в цифровых системах, при котором организация связи между функциональными компонентами отделена от их реализации. На уровне транзакций упор делается на функциональность обмена данными и меньше на их фактическую реализацию. При моделировании на уровне описания аппаратуры, как правило, используется тактовая синхронизация, а на уровне TLM время не используется, синхронизация выполняется, когда данные передаются между процессами. Все выполняемые при этом операции изменения состояний, передача данных и вычисления являются транзакциями. Модель уровня транзакций описывает компонент системы набором процессов, которые определяют его поведение и взаимодействуют одновременно. Каждая транзакция описывает преобразование данных, которые являются общими для взаимодействующих процессов.

Транзакции создаются динамически в процессе моделирования в соответствии с заданными условиями. Произведенные ими изменения состояний данных становятся видимыми для остальных процессов после их завершения.

TLM модели используются в современных средствах верификации проектов аппаратуры. Такой подход позволяет отделить процесс отладки системы и генерации тестовых последовательностей от ее аппаратной реализации.

II. Постановка задачи

В настоящей работе принят подход к верификации схемных реализаций систем управления на основе модели (model checking). С целью минимизации размера теста и в то же время максимизации вероятности обнаружения ошибок, ставится задача охвата тестами тех ситуаций, которые возможны в тестируемой системе в процессе ее штатной работы и отражены соответственно в ее спецификации, поэтому тестовые последовательности извлекаются из спецификации системы алгоритмическим путем и применяются к аппаратной реализации системы с тем, чтобы убедиться, что последняя ведет себя корректно, сравнивая ее выходные реакции с теми, которые предусмотрены в спецификации.

Рассматривается случай реактивных систем [5], состоящих из одновременно и независимо работающих компонентов. Важнейшим свойством таких систем является также присущий им параллелизм происходящих в них процессов.

Спецификация на проектирование систем с параллелизмом поведения задается на языке ПРАЛУ описания параллельных алгоритмов управления [5]. Этот язык используется для задания временной упорядоченности событий, возникающих при работе не только самого устройства управления, но и системы в целом, включая его внешнее окружение и описание поведения объекта управления. Реализация устройства рассматривается как черный ящик, для которого доступны только входы и выходы.

Алгоритм управления на ПРАЛУ представляется неупорядоченной совокупностью линейных цепочек l_i операций языка, каждая из которых открывается метками из μ_i и заканчивается метками из ν_i перехода: «$\mu_i : l_i \to \nu_i$». Порядок выполнения цепочек алгоритма управления в процессе его реализации определяется множеством запуска, его текущие значения N_t задают метки тех цепочек, которые могут выполняться одновременно. Если для цепочки «$\mu_i : l_i \to \nu_i$» выполняется $\mu_i \subseteq N_t$ и реализуется событие, с ожидания которого начинается цепочка l_i, то она запускается.

III. Моделирование описаний параллельных алгоритмов управления

Традиционно протокол моделируется как набор взаимодействующих процессов, где каждый процесс описывается как конечный автомат. Взаимодействие процессов представляется как коммуникация, в которых актами коммуникации служат транзакции через общие структуры данных, называемые каналами. Для устранения недетерминизма поведения атомарные операции процессов, каковыми являются транзакции, линеаризуются, в результате чего любое параллельное выполнение операций эквивалентно их некоторому последовательному выполнению. Для любого другого потока выполнение линеаризуемой операции является мгновенным: операция либо не начата, либо завершена. Синхронизация процессов модели TLM на уровне транзакций осуществляется барьерным механизмом. Барьер – это точки кода, в которых каждый процесс должен приостановиться и подождать достижения барьера всеми процессами группы. В модели TLM, например, на языке SystemC точки барьера задаются вызовами функции wait.

Ключевые моменты предлагаемой TLM модели описания на языке ПРАЛУ:

1. Произведено уточнение семантики основных операций языка – ожидания и действия. Суть уточнения состоит в доопределении частичного порядка реализации операций, задаваемого параллельным алгоритмом, до линейного. Используется модель параллелизма типа «чередование» (interleaving), в которой одновременность понимается как возможность упорядочивать операции произвольным образом.

2. Операции ожидания и действия рассматриваются в виде композиций линеаризуемых элементарных операций. Транзакции в алгоритмах на ПРАЛУ представлены операциями ожидания и действия, имеющими общую переменную и описывающими событие взаимодействия.

3. Структурой данных в модели TLM алгоритма на ПРАЛУ являются векторы текущих и планируемых значений переменных. Реализация операции ожидания алгоритма на ПРАЛУ состоит в выполнении операций приостановки и проверки текущих значений переменных, операции действия – в выполнения операций установки планируемых значений переменных.

4. Введено понятие ветви как совокупности последовательных процессов, начинающихся с некоторой операции. Ветвь является динамическим объектом, порождаемым операцией образования и уничтожаемым операцией прекращения.

5. Синхронизация параллельных ветвей осуществляется с помощью барьерного механизма. Достижение барьера фиксирует такты работы эмулятора и изменения значений переменных. Точки барьера задаются операцией приостановки выполнения ветвей. Структура данных барьера представлена в памяти очередями ОГ готовых для выполнения ветвей и ОЖ ждущих ветвей.

6. При моделировании алгоритма управления из ОГ последовательно извлекаются ветви и выполняются до приостановки, когда начальный фрагмент ветви G: 1) не может выполниться на множестве текущих значений переменных (тогда ветвь переносится в ОЖ); 2) выполнен (в вносится ветвь, начинающаяся с операции, которая должна выполняться в G следующей). При достижении барьера (когда ОГ=$\emptyset$): 1) элементы из ОЖ пересылаются в ОГ; 2) планируемые значения переменных пересылаются в текущие; 3) вводятся новые значения входных переменных.

IV. Заключение

Для предлагаемого метода генерации тестов разработана программная поддержка – средства моделирования и синтезаторы ПРАЛУ в модели аппаратуры на языках Verilog и C.

V. Список литературы

1. Adam Rose, Tom Fitzpatrick, Dave Rich, Harry Foster. Advanced Verification Cookbook. Mentor Graphics Corporation, 2008.
2. OVM Whitepaper. Mentor Graphics Corporation, Cadence, 2007.
3. Кулямин, В.В. Подход UniTesK к разработке тестов / В.В. Кулямин, А.К. Петренко, А.С. Косачев, И.Б. Бурдонов // Программирование. – № 6 (29). – 2003. – С. 25–43.
4. Cai, L. Transaction Level Modeling: An Overview / Lukai Cai, Daniel Gajski // First IEEE/ACM/ IFIP International Conference on Hardware/Software Codesign and Systems Synthesis , 1-3 Oct. 2003.– Newport Beach, CA, USA, 2003.
5. Закревский, А.Д. Параллельные алгоритмы логического управления / А.Д. Закревский. – Минск: Ин-т техн. кибернетики НАН Беларуси, 1999. – 202 с.