XXXI Неделя науки СПбГПУ. Материалы межвузовской научной конференции. Ч. VI: С.35-36, 2003.

УДК 658.512.011.56: 681.3.06

Л.В. Дворянский (4 курс, каф. ИУС), К.В. Орлов (асп., каф. ИУС),
В.П. Котляров, к.т.н., проф.

ЭФФЕКТИВНЫЙ ПОДХОД К РАЗРАБОТКЕ OMI-СОВМЕСТИМЫХ ПОВЕДЕНЧЕСКИХ МОДЕЛЕЙ

Современный процесс разработки микропроцессорных приложений предполагает широкое использование имитационных моделей со сложной многокомпонентной структурой. При разработке имитационных моделей используются различные языки высокого уровня, например Verilog HDL, VHDL, C, C++ и т.д. При этом наибольшее применение в разработке и тестировании большинства моделей программно-аппаратных комплексов находят Verilog, C и C++. Эти языки программирования имеют очень много общего, как в синтаксисе, так и в семантике, но обладают и существенными различиями. Например, Verilog содержит средства для описания сигналов, событий, параллельных процессов. С другой стороны, C позволяет описывать сложные структуры данных и работать с динамически распределяемой памятью, а C++ предоставляет эффективные средства для объектно-ориентированного программирования. Исходя из этого, Verilog используется в основном для написания RTL и gate-level моделей, а C/C++ удобен для разработки поведенческих моделей с высоким уровнем абстракции.

Благодаря наличию языковых средств, которые позволяют эффективно моделировать аппаратуру на низком уровне, Verilog стал стандартом де-факто в этой области. С C/C++ ситуация не столь однозначна. Из-за отсутствия встроенной поддержки конструкций, присущих специализированным языкам моделирования (например, сигналов и параллельных процессов) разработка моделей на чистом C/C++ неэффективна. Эту проблему позволяет решить использование специализированных библиотек, реализующих различные низкоуровневые объекты для моделирования (сигналы, порты, события и т.д.) и переключение параллельных процессов. Однако, не смотря на все преимущества, эти библиотеки не содержат необходимых средств для интеграции моделей с уже существующими симуляторами, вследствие чего область использования таких моделей очень ограничена.

Стандарт OMI (Open Module Interface – IEEE Std 1499-1998) призван разрешить проблему совместного использования моделей, написанных с использованием различных языков программирования и различных библиотек. Этот стандарт жестко специфицирует интерфейс между моделями и симуляторами, что позволяет OMI-совместимым моделям использоваться в любых OMI-совместимых симуляторах. На данный момент многие Verilog и VHDL симуляторы поддерживают стандарт OMI (в том числе NC-Verilog, NC-VHDL и Verilog-XL), а некоторые могут выполнять OMI модели, используя специальные прослойки (например, такая прослойка существует для всех Verilog-симуляторов). Таким образом, включив в библиотеку средства, позволяющие модели взаимодействовать с симулятором посредством OMI интерфейса, мы можем существенно расширить область применения модели.

Область применения C++ моделей также ограничена использованием платформозависимых реализаций переключения контекста (а переключение контекста – всегда платформо-зависимая задача). Таким образом, разработчикам компонентов, предназначенных для использования на большом числе платформ, зачастую приходится отказываться от применения концепции параллельных процессов и других средств, присущих языкам имитационного моделирования, что существенно увеличивает сложность и время разработки моделей.

В большинстве C++ симуляторов процесс – это функция специального вида, которая выполняется в своем собственном контексте (на отдельном стэке, т.к. компиляторы C++ обычно размещают локальные переменные на стэке) и содержит вызовы функции, которая производит переключение контекстов. Функция переключения сохраняет регистры на стэке и выполняет переход к следующему процессу. Переключение стэков и сохране-

ние/восстановление регистров реализуется либо с помощью средств операционной системы, либо с использованием кода на ассемблере. Эти решения очень зависимы от платформы, а первое плюс ко всему еще и вносит большие задержки в выполнение модели.

Предлагается платформо-независимое решение этой задачи, в основе которого лежит отказ от хранения локальных переменных процесса на стэке, вследствие чего отпадает необходимость в использовании платформо-зависимого кода для переключения контекста, и для переключения между процессами можно выходить из функции в операторе wait и вызывать функцию следующего процесса. При этом также отпадает необходимость в сохранении и восстановлении регистров, так как во всех переменных процесса после возврата из функции будут корректные значения. Код модели можно изначально писать без использования локальных переменных в процессах, но предпочтительнее преобразовать локальные переменные процесса в поля некоторого класса с помощью некоего препроцессора C++ кода, что позволит разрабатывать код моделей в виде, поддерживаемом основными C++ симуляторами.

Таким образом, для того, чтобы иметь максимально широкую область применения, разрабатываемые поведенческие модели должны удовлетворять следующим требованиям: не содержать в себе платформо-зависимого кода и обладать стандартизированным интерфейсом. Применяя описанные выше методы (использование специального препроцессора для трансляции кода модели с целью получения бесстэковых процессов и реализации средств взаимодействия модели с симулятором посредством OMI интерфейса), мы можем получить систему, позволяющую эффективно разрабатывать переносимые модели на C++ с возможностью простой интеграции с существующими моделями, написанными на других языках для разработки имитационных моделей, например на VHDL и Verilog.