XXIX Неделя науки СПбГТУ. Материалы межвузовской научной конференции. Ч.V: С.26-27, 2001.

УДК 658.512.011.56, 681.3.06

К.Г. Зыбин (5 курс, каф. ИУС), П.Б. Егоров (асп., каф. ИУС),
В.П. Котляров, к.т.н., проф.

СИМУЛЯТОР ДЛЯ ИМИТАЦИОННЫХ МОДЕЛЕЙ В СТАНДАРТАХ OMI

Целью работы являлось создание симулятора поддерживающего стандарт OMI для создания имитационных моделей.

OMI стандарт определяет интерфейс между моделью и приложением (симулятором), которое использует данную модель. Стандарт описывает набор программных интерфейсов и правил их использования, которые позволяют модели взаимодействовать с приложением и ограничивают доступ к внутренней информации модели. Стандартизация данного интерфейса обеспечивает возможность подключения моделей разных производителей к одному и тому же приложению.

Модель подключается к приложению через специальный менеджер, который и обеспечивает поддержку данного стандарта. При этом к одному приложению может подключаться несколько менеджеров моделей. Менеджер модели может подключать к себе библиотеки, которыми может пользоваться модель. Например, в этих библиотеках могут быть описаны стандартные части электронных компонентов. К одному менеджеру возможно подключение нескольких моделей. В качестве модели к менеджеру может подключаться другой менеджер. Таким образом, стандарт OMI разрешает создавать древовидную структуру из моделей и менеджеров, что обеспечивает возможность использования в качестве частей модели другие, изготовленные ранее, модели и библиотеки. При этом возможно в процессе создания модели постепенно увеличивать степень ее детализации. Подключаемые модели не обязательно должны быть написаны на одном и том же языке.

Взаимодействие между моделью и приложением осуществляется через процедуры вызова (callback's), порядок вызова которых определен стандартом, но есть возможность не вызывать некоторые из них, или наоборот, ввести новые процедуры такого же типа. Процедуры вызова определены как набор функций описанных на языке ANSI C. Таким образом, сборка модели и приложения происходит на уровне сборки объектных файлов.

Весь этап работы с моделью разбит на 5 частей: загрузка модели, создание модели, инициализация модели, симуляция, завершение работы. Каждому этапу соответствуют свои процедуры вызова.

Загрузка модели (Bootstrap) - этап, на котором устанавливается взаимодействие между моделью и приложением. На данном этапе также выполняется проверка версий и возможность взаимодействия между моделью и приложением. На данный момент существует только одна версия данного стандарта, поэтому проверка версий сводится к тривиальной процедуре. Этап загрузки модели должен быть выполнен первым.

Создание модели (Elaboration) - процесс создания экземпляров моделей, с которыми будут производиться действия во время симуляции. Это следующий после загрузки этап.

Инициализация модели (Initialization) - Установление начальных значений для объектов модели, имеющих состояния. Начальные значения могут устанавливаться как самой моделью, так и приложением. В последнем случае используются значения по умолчанию, не определенные стандартом. Этот этап происходит после создания модели.

Симуляция (Simulation) - процесс моделирования поведения системы. В этот момент происходит обмен данными между портами модели и перевод ее из одного состояния в другое.

Завершение работы (Termination) - процесс закрытия OMI сессии. Он может производиться в любое время после начала работы приложения.

Симуляционный цикл делится на 6 состояний, циклически сменяющих друг друга, в каждом из которых возможно взаимодействие с моделью. Это деление избыточно, но за счет этого имеется возможность подключать к приложению модели написанные на разных языках программирования или моделирования. Для этого также в стандарт введена поддержка типов данных языков Verilog и VHDL. Стандарт OMI поддерживает два типа симуляции: поцикловая (cycle-based) и управляемая событиями (event-driven). В принципе возможно одновременное подключение моделей обоих типов к приложению.

Результаты. В процессе работы было создано ядро симулятора, которое поддерживало симуляцию управляемую событиями. Данное ядро поддерживало все типы данных описанных в стандарте. При этом поддерживалось создание резольвированных сигналов, т.е. сигналов, к которым подключалось больше двух портов модели. Ядро взаимодействовало с тестовыми моделями по описанному в стандарте протоколу.
Для проверки работы ядра через простейший менеджер к нему были подключены модели, написанные на языке C++. Модели прошли полный цикл симуляции. Результаты поведения этих моделей соответствовали поведению реальных устройств.