УДК 004.94

Д.А. Гавриш, С.Н. Саранча

Харьковский национальный университет радиоэлектроники, Харьков

МЕТОДЫ РАСПРЕДЕЛЕННОГО МОДЕЛИРОВАНИЯ ДИСКРЕТНО-СОБЫТИЙНЫХ СИСТЕМ

Рассмотрены методы распределенного моделирования дискретно-событийных систем. Моделирование электронного логического компонента как частный случай моделирования дискретно-событийной системы. Определены дальнейшие перспективы усовершенствования алгоритмов моделирования.

Ключевые слова: *дискретно-событийная система, имитационное моделирование.*

Введение

Моделирование и анализ поведения во времени динамических систем являются важными областями в науке и технике. Более «реалистичной» реализации поведения во времени динамической системы присуща большая сложность вычислений. Проведение имитационных экспериментов является трудоемкими по нескольким причинам. Во-первых, проектирование более сложных систем требует более глубоких навыков и, как правило, значительных усилий. Во-вторых, как только модель описана, выполнение моделирования может занять много времени. Это определяется либо целью моделирования, либо природой описанной модели. Для сбора статистической информации об объекте требуется выполнить серию запусков моделирования с целью определения необходимого соответствия параметров. Другой составляющей повышения эффективности моделирования является изучение как можно большего количества различных параметров модели путем циклического выполнения моделирования. Имитационное моделирование может требовать значительных вычислительных ресурсов.

Представляется возможным увеличить скорость моделирования за счет использования большего количества вычислительных ресурсов, в частности, нескольких процессоров, работающих параллельно. Так как модель отражает реальные системы, которые состоят из параллельно работающих компонентов, этот скрытый параллелизм может сделать использование параллельного моделирования эффективным. Кроме того, для выполнения независимого повторения одной и той же имитационной модели с различными входными параметрами, распараллеливание кажется тривиальным.

Анализ литературных данных. На текущий момент можно найти много публикаций, касающихся области оптимизации и ускорения процесса моделирования. До появления параллельных дискретно-событийных систем (DES) в процессе моделирования сложных систем использовались SIMD вычислительные системы. DES производит декомпозицию моделируемой системы и выполняет обработку событий в отдельных блоках, именуемых логическими процессами. Как правило, логические процессы распределяются по различным рабочим станциям. Для проведения распределенного моделирования необходимо определить протокол синхронизации, который служит для синхронизации модельного времени и передачи событий между логическими процессами. Передача сообщений гарантирует сохранение причинно-следственных связей между событиями в моделируемой системе.

По принципу организации синхронизации модельного времени между логическими процессами различают классический и оптимистический протокол распределенного моделирования. В контексте консервативного протокола моделирования можно выделить работы K.M. Chandy, J. Misra, D.A. Jeffer-

son. Также в этих работах рассмотрены способы предотвращения и восстановления клинча, которые базируются на использовании временных и отправки нулевых сообщений в консервативном протоколе. Для оптимистического протокола были рассмотрены: процесс отката модельного времени (также называемый «Искривление времени» – Time Warp), оптимистическое временное окно, управление памятью, вычисление глобального модельного времени, метод управления параллельными процессами, который базируется на вероятностном предсказании поведения системы, а также адаптивные системы управления.

1. Представление цифровой системы

Описание электронного компонента на языках VHDL или Verilog/SystemVerilog представляет собой сложную иерархическую структуру, в которой каждый компонент может содержать в себе набор составных элементов. Данная структура применяется для упрощения процесса проектирования за счет использования модульности.

В языке VHDL компонент описывается парой entity – architecture. В entity хранятся константы генерации и описание портов компонента, а в архитектуре описывается его поведение или структура.

В языке Verilog/SystemVerilog компонент описывается как module, в котором содержится как описание интерфейсов компонента, так и его структура либо поведение.

Каждый компонент, в свою очередь, содержит набор параллельных логических процессов и набор объектов, которыми эти процессы оперируют. В общем виде моделируемый электронный компонент имеет структуру, представленную на рис. 1.

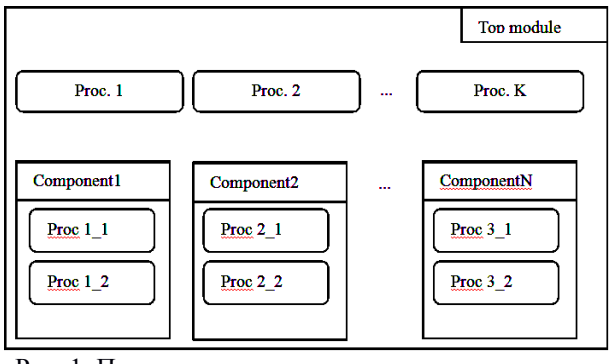

Рис. 1. Пример описания электронного компонента

Таким образом, цифровая система на высоком уровне абстракции может быть представлена в виде $Sys = \{Ent, Arc\}$, где *Ent* – описание интерфейса системы, включающее набор портов – внешних сигнальных соединений и набор констант-параметров, а *Arc* – описание законов поведения системы, которые могут быть представлены в виде алгоритмического или структурного описания, системы булевых функций и т.д. При этом использование в языке VHDL *Ent* и *Arc* может быть однозначно сопоставлено с соответствующими языковыми конструкциями, а при использовании языка Verilog можно говорить о модуле (module), который совмещает в себе функции указанных конструкций.

Описание интерфейса представляет собой объединение информации о настроечных параметрах *Params* и внешних сигнальных соединениях *Ports*. Описание интерфейса осуществляется в виде $Ent = \{Params, Ports\}$ где $Params = |Param_p|$. Каждый $Param_p$ описывается в виде $Param_p = \{PName_p, PType_p, PDefault_p, PValue_p\}$, где $PName_p$, $PType_p$, $PDefault_p$, $PValue_p$ – имя, тип, значение по умолчанию и текущее значение параметра $Param_p$ соответственно. Аналогично $Ports = |Port_t|$ где каждый порт определяется в виде $Port_t = \{PName_t, PMode_t, PType_t, PResolution_t, PDefault_t\}$, где $PName_t$, $PMode_t$, $PType_t$, $PResolution_t$, $PDefault_t$ – имя порта, его режим работы (ввод, вывод, двунаправленный), тип, используемая функция разрешения, значение по умолчанию для порта $Port_t$.

Описание законов функционирования системы представляется в виде архитектурного тела *Arc*, которое содержит набор параллельных операторов и процессов, т.е. $Arc = \{Pr_i\}$. Показано, что все параллельные операторы языка VHDL и Verilog имеют однозначное соответствие процессу с определенным набором последовательных операций и списком чувствительности. К параллельным операторам относятся: оператор назначения сигнала (в простой или расширенной форме оператор назначения сигнала по условию или по выбору), оператор процесса, оператор логического блока, оператор создания экземпляра компонента и оператор generate for / generate if.

Таким образом, каждый процесс представляется в виде последовательности операторов $Pr_i = \{Stmt_{ik}\}$, к которым относятся операторы изменения значений локальных переменных процесса и/или глобальных сигналов, операторы условий и циклов, операторы вызовов функций и процедур, операторы вывода отладочных сообщений *report* / *assert*.

2. Событийное и временное моделирование

Существует два вида процесса моделирования, которые отличаются по принципу увеличения модельного времени. В синхронной дискретной системе модельное время увеличивается на временные промежутки равной длины, или другими словами, моделируемая динамичная система дискретизирована мелкими промежутками времени. Выбор точности моделирования и длительности времени моделирования подчиняется закономерности: время од-

ного такта должно быть достаточным для обеспечения требуемой точности моделирования (чем меньше длительность одного такта, тем больше точность). Уменьшение длительности такта увеличивает длительность процесса моделирования. В асинхронном моделировании события беспорядочно разбросаны во времени, и управляемые временем концепции порождают неэффективные алгоритмы моделирования.

Событийное дискретное моделирование производит запуск системы моделирования в моменты появления событий. В дальнейшем такой способ обозначается как дискретно-событийное моделирование (DES – discrete event simulation). В DES во время выполнения последовательных повторяющихся процессов возникают события в модельном времени (иногда его называют «виртуальным временем», VT «virtual time»). Эти события находятся в упорядоченном во времени списке (event list EVL), хранящем события, которые произойдут в будущем с указанным временем. Текущее состояние системы определяется модельным временем и состоянием переменных $S= (s_1, s_2, \dots s_n)$ (пример проиллюстрирован на рис. 2). Ядро моделирования (SE – simulation engine) управляет процессом моделирования; постоянно извлекая первое событие из списка (элемент списка с наименьшей временной меткой), выполняет обработку события посредством изменения состояния переменных и/или планируя новые события в EVL. В процессе моделирования возможно удаление устаревших (не актуальных) событий. Моделирование выполняется до тех пор, пока имеются запланированные в будущем события и не достигнуто некоторое предопределенное время завершения моделирования.

Событийная DES использует ядро моделирования, которое продемонстрировано на рис. 2 и использует соответствие между набором событий, происходящих в физической системе, и выполнением переходов в сети Петри (PN – Perri Net); всякий раз, когда происходит событие в реальной (физической) системе, выполняется переход в модели. Список событий, следовательно, осуществляет переходы в определенный момент модельного времени, учитывая, что выборка событий является не вытесняемой. Состояние системы PN, обозначенное как (S), изменяется во время обработки события, т.е. выполняется переход: событие с наименьшей временной меткой удаляется из списка запланированных событий, а S меняется согласно соответствующего движения токенов. Новое состояние, однако, может сделать доступными новые переходы (в некоторых случаях может даже сделать недоступными доступные переходы). Тогда EVL должен быть скорректирован следующим образом: планируется новый доступный переход с выполнением его с указанным временем в будущем, со вставкой минимального значения к EVL (в то время как недоступные переходы будут удалены). Окончательно VT принимает значение момента времени (timestamp – ts) перехода, который выполнился.

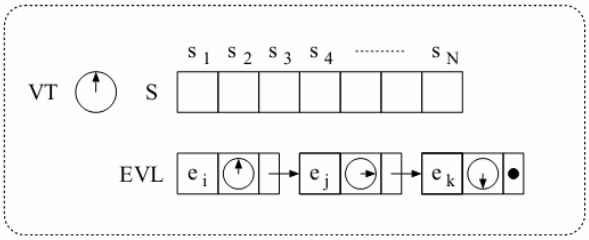

Рис. 2. Ядро моделирования
для дискретно-событийных систем

При моделировании дискретно-событийной системы могут возникнуть как параллельные, так и взаимоисключающие события. Особенностью параллельных событий является то, что их появление не взаимосвязано и возможно выполнять обработку этих сообщений параллельно. С другой стороны, с целью улучшения последовательного моделирования они могут быть удалены из EVL на одном шаге моделирования. При обработке взаимоисключающих событий моделирование одного из них (кроме изменения состояния) приводит к удалению другого из EVL. Обработка таких событий не может выполняться параллельно.

3. Уровни параллелизма/ распределения процесса моделирования

В зависимости от организации параллельного моделирования можно выделить следующие уровни параллелизма:

1. Уровень приложений. Наиболее очевидным ускорением моделирования с возможностью обеспечения широкомасштабных исследований является введение независимых репликаций одинаковых моделей с возможностью ввода различных параметров для доступных процессоров. Таким образом, связь между процессорами не нужна, и можно ожидать высокую эффективность процесса моделирования. Распределение процесса моделирования может быть неосуществимым по причине ограниченности доступной памяти на отдельных обрабатывающих устройствах.

2. Уровень подпрограмм. Среда моделирования изучает, в каких случаях нужно упорядочить зависимости между репликациями на протяжении итерации. То есть, входные параметры репликации i определены выходными параметрами репликации i-1. Распределение подпрограмм, входящих в состав имитационного эксперимента, может быть эффективным в случае генерации случайных чисел, обра-

ботки событий, обновления состояний системы, сбора статистики работы системы. В связи с довольно небольшим количеством задач среды моделирования, количество процессоров, которые могут быть использованы, и, следовательно, степень достижимого ускорения, ограниченно подпрограммным уровнем распределения.

3. Уровень компонентов. Ни один из приведенных выше уровней распределения не дает возможности использовать параллелизм, который имеется в моделируемой физической системе. Для этого она должна быть разделена на компоненты или подмодели. Такая декомпозиция непосредственно раскрывает внутренний параллелизм модели или, как минимум, предоставляет возможность получить его во время процесса моделирования. Проблема декомпозиции может быть решена при помощи проектирования объектно-ориентированной системы, где экземпляры классов соответствуют компонентам реальной системы, которые представляют собой вычислительные задачи, выполняющиеся параллельно.

4. Уровень событий, централизованная модель. Реализация EVL позволяет параллельно обрабатывать события. На схеме приводится централизованное хранение данных (EVL) на ведущем процессоре. Ускорение может быть обеспечено при помощи параллельной обработки одновременных событий (которые требуют большое количество процессорного времени) на наборе подчиненных свободных процессоров. Ведущий процессор в данном случае заботится о сохранении актуальности очередей событий, то есть, планировании обработки событий с учетом возможности их параллельной обработки. Для осуществления этого необходимы знания о событийной структуре, которая может быть извлечена из модели. Распределение на уровне событий с централизованным EVL особенно подходит для многопроцессорных систем с общей памятью. EVL может быть представлен в виде общей структуры данных, который доступен для всех процессоров. Одновременно обрабатывающиеся события обычно имеют одно и то же модельное время или ограничены малым временным интервалом.

5. Уровень событий, децентрализованный EVL. Наиболее рекомендуемым методом проведения параллельного моделирования является планирование обработки различными процессорами событий из произвольных компонентов модели и различного модельного времени. Планирование может осуществляться регулярным или неструктурированным образом. Действительно, возможно ожидать высокую степень параллелизма за счет применения этого метода, который позволяет одновременно моделировать события с различными временными метками. Реализация этой системы предполагает наличие протоколов для локальной синхронизации, которые могут, в свою очередь, быть причиной увеличения коммуникационных расходов в зависимости от распределения событий в базовой модели во времени.

4. Синхронное моделирование логических процессов

В синхронном моделировании логического процесса все локальные таймеры логических процессов имеют одинаковые значения в любой момент времени. Таким образом, каждый локальный таймер выполняет последовательность дискретных значений (0, Δ, 2Δ, 3Δ, …). Иными словами, процесс моделирования выполняется в соответствии с глобальным таймером. Все локальные таймеры хранят значение глобального таймера. Каждый логический процесс должен обрабатывать все события во временном интервале [iΔ; (i+1) Δ) (где i – номер такта) перед тем, как любой другой логический процесс начнет обработку событий с временной меткой (i+1)Δ и больше. Эта методология значительно упрощает реализацию корректного моделирования, предотвращая проблемы «клинча» и вероятность получения очень большого потока сообщений и/или затрат памяти для работы протоколов синхронизации при асинхронном моделировании. Более того, это дает возможность эффективно использовать механизм барьерной синхронизации, который доступен в большинстве сред параллельной обработки. Дисбаланс в работе логических процессов на каждом такте, с другой стороны, естественно приводит к простоям и, следовательно, представляет собой источник повышения эффективности.

Существует централизованный и децентрализованный подходы к реализации глобального таймера. В работе [2] предлагается реализовать централизованное управление глобальным таймером на одном выделенном процессоре. Были разработаны алгоритмы, которые определяют момент времени для каждого логического процесса, когда необходимо взаимодействие с другим логическим процессом с целью предотвращения срабатывания в те моменты времени, когда не произошло ни одного события. Как только определен минимальный временной промежуток до следующего возможного события, глобальное время увеличивается на Δ(S) – значение, которое зависит от частичного состояния S. Для реализации распределенного глобального таймера используется структурная (иерархическая) организация логических процессов для определения минимального времени до наступления следующего события. Параллельная операция определения следующего события с минимальным значением временной метки передает этот параметр в корень дерева процессов, который потом распространяет это время вниз по дереву. Существует распределенный алгоритм «snapshot» для избегания

«узких мест» централизованного глобального координатора времени.

Возможно также комбинированное применение синхронного моделирования логических процессов с событийным механизмом управления глобальным временем. Глобальное время увеличивается к минимальному значению временной метки следующего события, так же, как и в случае событийного управления. Логические процессы позволяют только моделировать один дельта цикл, который называется Лобачевским, как с ограниченной задержкой, так и перемещающимся временным окном.

5. Асинхронное моделирование логических процессов

Асинхронное моделирование логического процесса предполагает наличие событий, происходящих в различные моменты времени, которые не зависят друг от друга. Таким образом, параллельная обработка этих событий эффективно ускоряет время выполнения последовательного моделирования.

Главной проблемой асинхронного моделирования логических процессов является возможность появления ошибок причинности, которые основываются на наличии причинных связей между событиями. Действительно, асинхронное моделирование логического процесса гарантирует корректность, если глобальное упорядочивание событий, которое сформировано последовательной DES, состоит из частично упорядоченных событий, сгенерированных во время параллельного выполнения. Джефферсон определил ошибку причинности как обратную задачу увеличения модельного времени Лампорта, т.е. определение значения модельного времени при обработке событий в распределенной системе осуществляется так, как будто все события происходят упорядочено логически во времени. Возникновение ошибок причинности никогда не наступит в асинхронном моделировании логического процесса тогда и только тогда, когда в каждом логическом процессе придерживается порядок обработки событий с увеличением временных меток (lcc – local causality constraint, локальное ограничение причинности). С другой стороны, не всегда необходимо соблюдать lcc потому, что возникновение двух событий в одном и том же логическом процессе может быть параллельным (независимыми друг от друга), и, следовательно, возможна их обработка в произвольном порядке. Существует две категории механизмов асинхронного моделирования логических процессов, которые придерживаются lcc следующим образом:

– консервативный метод. Этот метод строго избегает нарушения lcc, даже если есть ненулевая вероятность того, что порядок событий к ошибке не приведет;

– оптимистический метод. Этот метод использует возможность обработки событий, даже если есть ненулевая вероятность возникновения ошибки, связанной с причинностью событий.

Производительность асинхронного моделирования логических процессов, по сравнению с синхронным, увеличивается как минимум в O(logP) раз. P представляет собой число параллельно выполняемых логических процессов на независимых процессорах. Анализ предполагает, что каждый такт экспоненциально распределяет время своего выполнения $T_{step,i} \approx \exp(\lambda)$ в каждом LP_i $\left(E[T_{step,i}] = \frac{1}{\lambda}\right)$. Как следствие, ожидаемое время моделирования $E[T^{sync}]$ для k тактов синхронного моделирования является k:

$$E\left[\max_{i=1..p}\left(T_{step,i}\right)\right] = k\frac{1}{\lambda}\sum_{i=1}^{P}\frac{1}{i} \le \frac{k}{\lambda}\log(P).$$

Простой теперь является ограничением синхронизации (как в асинхронном моделировании). Ожидаемое время моделирования будет

$$E\left[T^{async}\right] = E\left[\max_{i=1..p}\left(T_{step,i}\right)\right] \succ \frac{k}{\lambda}.$$

Мы имеем $\lim_{k\to\infty,P\to\infty}\frac{E\left[T^{sync}\right]}{E\left[T^{async}\right]} \approx \log(P)$.

Временя моделирования k при асинхронном способе увеличится почти в log(P) раз по сравнению с синхронным. Максимально достижимое ускорение в любое время активации моделирования равно $\frac{P}{\log(P)}$. Эти результаты, однако, являются прямым следствием предположения об экспоненциальном времени выполнения шага моделирования, т.е. необходимо сравнить значение k-кратной суммы с максимальным распределением случайных переменных по экспоненциальному закону с ожиданием максимального значения за P k-стадии случайных переменных распределенных по закону Ерланга. За время выполнения шага, равномерно распределенного по $[l, u]$, мы имеем $\lim_{k\to\infty,P\to\infty}\frac{E\left[T^{sync}\right]}{E\left[T^{async}\right]} \approx 2$, или, предположительно, при $T^{sync} \le ku$ и $E\left[T^{async}\right] \ge k\frac{(l+u)}{2}$ разница между синхронным и асинхронным временем выполнения является $\frac{2}{(ku)(k(l+u))} \le 2$, т.е. константная. Таким образом, для локального события время обработки распределяется с конечным увеличением при асинхронном методе и напрямую зависит от количества независимых процессоров.

Конечно, предложенная модель далека от того, что можно было бы наблюдать в реальных процессах на различных платформах, но результаты могут помочь оценить эти два подхода, по крайней мере, со статистической точки зрения.

6. Логический процесс моделирования дискретно-событийных систем

Общим требованием для всех стратегий моделирования с распределением на уровне событий является возможность разделить общую задачу моделирования на набор логических процессов, взаимодействующий между собой (logical processes – LPs), которые пытаются использовать присущий параллелизм между соответствующими компонентами модели и параллельно выполнять этих процессы. Таким образом, мы можем представить логический процесс моделирования как организованное взаимодействие набора LP, каждый из которых моделирует ограниченную область событийной структуры. В общем случае область представляется как набор всех событий в определенном промежутке времени моделирования, или набор событий в заданной части модели.

Основные части логического процесса моделирования представлены на рис. 3:

– набор логических процессов для выполнения параллельной обработки событий синхронно или асинхронно;

– коммуникационная схема обеспечивает обмен данными между логическими процессами для локальной синхронизации.

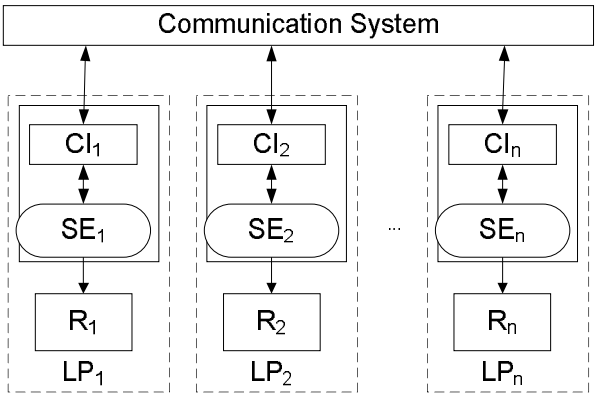

Рис. 3. Логическая схема процесса моделирования

На рис. 3:

CI (Communication Interface) – интерфейс связи;

SE (Simulation Engine) – ядро моделирования;

R (Region, Simulation Sub-Model) – регион, компонент моделирования

LP (Logical Process) – логический процесс

Каждый LP_i является частью области R_i как часть модели, на которой среда моделирования SE_i, управляемая событиями (рис. 2), выполняет локальную (и генерирует удаленную) обработку событий, и таким образом увеличивается локальное время (local virtual time – LVT).

Каждый LP_i (SE_i) имеет доступ только к жестко определенному подмножеству переменных состояния S_iS, которые пересекаются с переменными состояния, переданными другими логическими процессами.

В каждом LP_i обрабатываются два вида событий: внутренние события, причиной возникновения которых являются S_iS, и внешние события, которые также влияют на локальные состояния других логических процессов S_iS ($i \neq j$).

Коммуникационный интерфейс CI_i, подключенный к SE, обеспечивает распространение событий для их обработки на удаленном логическом процессе и учет причинных связей для моделирования локальных событий, которые инициированы удаленными логическими процессами. Основным механизмом для этого является передача, прием и обработка сообщений о событиях, соединенных с копиями LVT на стороне отправителя.

Существуют два основных класса CIs для логического процесса моделирования: с консервативными и с оптимистичными концепциями определения очередности, в которой необходимо обрабатывать события. Обе они основываются на отправке сообщений с информацией о причинной связи одного события со всеми остальными. CI несет ответственность за предотвращение глобального нарушения причинности событий. В первом случае (в консервативном протоколе) CI взаимодействует с SE таким образом, при котором всегда происходит предотвращение ошибок причинности (путем блокирования SE, если есть возможность обработать «небезопасные» события, для которых причинные зависимости по-прежнему в состоянии ожидания). В оптимистическом протоколе CI взаимодействует с SE, допуская повторное моделирование событий, которые определены как преждевременно обработанные и не соответствуют условиям причинных связей, произведенных другими LPs. В обоих случаях сообщения вызываются и собираются коммуникационными интерфейсами логических процессов, распространение которых в режиме реального времени зависит от технологии, на которой базируется коммуникационная система. Реализация на практике протоколов CI, разработанных в теории, в большой степени зависит от технологий, используемых в конечной мультипроцессорной архитектуре.

Для представления и изменения виртуального модельного времени (VT – virtual time) при моделировании LP рассматривается два способа его реализации синхронным логическим процессом:

– как глобальное время, которое представлено в явном виде (централизованная структура данных);

– неявная его реализация как выполнение процедуры синхронизации.

Ключевой характеристикой является то, что каждый LP (в любой момент в режиме реального времени) представляет то же VT. Это ограничение смягчается в асинхронном моделировании LP, где каждый LP поддерживает локальное виртуальное время (LVT – local virtual time). Таким образом, логические процессы могут иметь различные значения виртуального времени в заданный момент в режиме реального времени.

Выводы

Вопросы, связанные с имитационным моделированием электронного компонента, на сегодняшний день являются актуальными по причине невозможности или очень большой стоимости проведения исследований с реальным электронным компонентом. Для верификации спроектированного электронного компонента, как правило, используются среды моделирования, которые могут воссоздать его работу по имеющемуся описанию на языке проектирования аппаратуры. Процесс моделирования электронного компонента можно рассматривать как процесс моделирования дискретно-событийной системы.

В данной статье были рассмотрены уровни параллелизма при параллельном моделировании. Параллелизм на уровне приложений широко используется в процессе верификации, когда выполняется многократный запуск системы на моделирование с различными входными данными. При этом одновременно может быть запущено большое количество процессов моделирования. Другие виды параллелизма используются в случае, когда невозможно обеспечить процесс моделирования средствами одной рабочей станции. Актуальность использования этих уровней параллелизма обуславливается большой сложностью моделируемых систем и большой длительностью процесса моделирования.

Список литературы

1. Peacock J.K. Distributed Simulation using a Network of Processors [Текст] / J.K. Peacock, J.W. Wong, E.G. Manning // Computer Networks. – 1979. – Vol. 3, No. 1. – P. 44-56.

2. Venkatesh K. Discrete Event Simulation in a Distributed System [Текст] / K. Venkatesh, T. Radhakrishnan, H.F. Li // In: IEEE COMPSAC, IEEE Computer Society Press, 1986. – P. 123-129.

3. Concepcion A.I. Mapping Distributed Simulators on to the Hierarchical Multi bus Multiprocessor Architecture [Текст] / A.I. Concepcion // In: P. Reynolds, Ed., Proc. of the SCS Multiconference on Distributed Simulation, Society for Computer Simulation, 1985. – P. 8-13.

4. Baik D. Perfromance Evaluation of Hierarchical Distributed Simulators [Текст] / D. Baik, B.P. Zeigler // In: Proc. of the 1985 Winter Simulation Conference, SCS, 1985. – P. 421-427.

5. Chandy K.M. Distributed Snapshots: Determining Global States of Distributed Systems [Текст] / K.M. Chandy, J. Lamport // ACM Transactions on Computer Systems. – 1985. – Vol. 3, No.1. – P. 63-75.

6. Lubachevsky B.D. Bounded lag distributed discrete event simulation [Текст] / B.D. Lubachevsky // In: B. Unger, D. Jefferson, Eds., Proceedings of the SCS Multiconference on Distributed Simulation, SCS. – February. – 1988. – 19(3). – P. 183-191.

7. Sokol L.M. MTW: a strategy for scheduling discrete simulation events for concurrent execution [Текст] / L.M. Sokol, D.P. Briscoe, A.P. Wieland // In: Proc. of the SCS Multiconf. on Distributed Simulation, 1988. – P. 34-42.

8. Jefferson D.A. Virtual Time [Текст] / D.A. Jefferson // ACM Transactions on Programming Languages and Systems, July1985. – Vol. 7, No. 3. – P. 404-425.

9. Lamport L. Time, clocks, and the ordering of events in distributed systems [Текст] / L. Lamport // Communications of the ACM. – Jul 1978. – Vol. 21, No. 7. – P. 558-565.

10. Misra J. Distributed Discrete-Event Simulation [Текст] / J. Misra // ACM Computing Surveys. – March 1986. – Vol. 18, No. 1. – P. 39-65.

11. Fujimoto R.M. Parallel Discrete Event Simulation [Текст] / R.M. Fujimoto // Communications of the ACM. – October 1990. – Vol. 33, No. 10. – P. 30-53.

12. Felderman R.E. An Upper Bound on the Improvement of Asynchronous versus Synchronous Distributed Processing [Текст] / R.E. Felderman, L. Kleinrock // In: D.Nicol, Ed., Proc. of the SCS Multiconf. on Dist. Sim. – Jan 1990. – P. 131-136.

Поступила в редколлегию 14.05.2014

Рецензент: д-р техн. наук, проф. О.Ф. Михаль, Харьковский национальный университет радиоэлектроники, Харьков.

МЕТОДИ РОЗПОДІЛЕНОГО МОДЕЛЮВАННЯ ДИСКРЕТНО-ПОДІЄВИХ СИСТЕМ

Д.О. Гавриш, С.М. Саранча

Розглянуті методи розподіленого моделювання дискретно-подієвих систем. Моделювання електронного логічного компоненту як окремий випадок моделювання дискретно-подієвої системи. Визначені подальші перспективи удосконалення алгоритмів моделювання.

Ключові слова: *дискретно-подієва система, імітаційне моделювання.*

DISTRIBUTED METHODS OF DISCRETE-EVENT SYSTEM SIMULATION

D.A. Havrysh, S.N. Sarancha

In the article methods of the discrete-event system distributed simulation have been considered. In order to reproduce behavior of the digital electronic device we can use discrete-event system (DES) simulation. DES representation of the hardware described by VHDL or Verilog has been considered as well. Special attention is paid to the definition of prospects of improving simulation algorithms.

Keywords: *discrete-event system, distributed simulation.*