УДК 681.3

Ю.В. ЛАДЫЖЕНСКИЙ, Ю.В. ПОПОВ, Г.А. ТЕСЛЕНКО

Донецкий национальный технический университет, Украина

ПРОГРАММНАЯ СИСТЕМА ДЛЯ РАСПРЕДЕЛЕННОГО ЛОГИЧЕСКОГО МОДЕЛИРОВАНИЯ С ДИНАМИЧЕСКИМ ПРОТОКОЛОМ СИНХРОНИЗАЦИИ

Разработана программная система для распределенного логического моделирования. Рассмотрены основные компоненты программной системы. Приведено описание используемого динамического протокола синхронизации.

синхронизация, динамический протокол, логическое моделирование

Введение

Сложность и размеры современных проектов цифровых систем требуют эффективных по производительности средств верификации. Имитационное моделирование является важным инструментом при верификации проектов цифровых систем. Оно используется для функционального и временного моделирования проекта и проводится с целью выявления ошибок, возникающих на стадиях проектирования.

Современные методы верификации цифровых систем с использованием моделирования характеризуются большими временными затратами и последовательным способом выполнения операций.

Использование алгоритмов параллельного и распределенного вычислений [1,2] является перспективным направлениям для ускорения моделирования. Создание на их основе эффективных по быстродействию параллельных и распределенных программных и аппаратных средств моделирования цифровых устройств является актуальной научно-технической проблемой.

Целью работы является исследование динамических алгоритмов синхронизации с целью повышения быстродействия распределенного логического моделирования цифровых систем.

1. Структура программной системы для распределенного логического моделирования

Программная система для распределенного логического моделирования состоит из трех подсистем [3]: подсистема подготовки схемы и входного воздействия для моделирования, подсистема моделирования и подсистема анализа результатов моделирования (рис. 1).

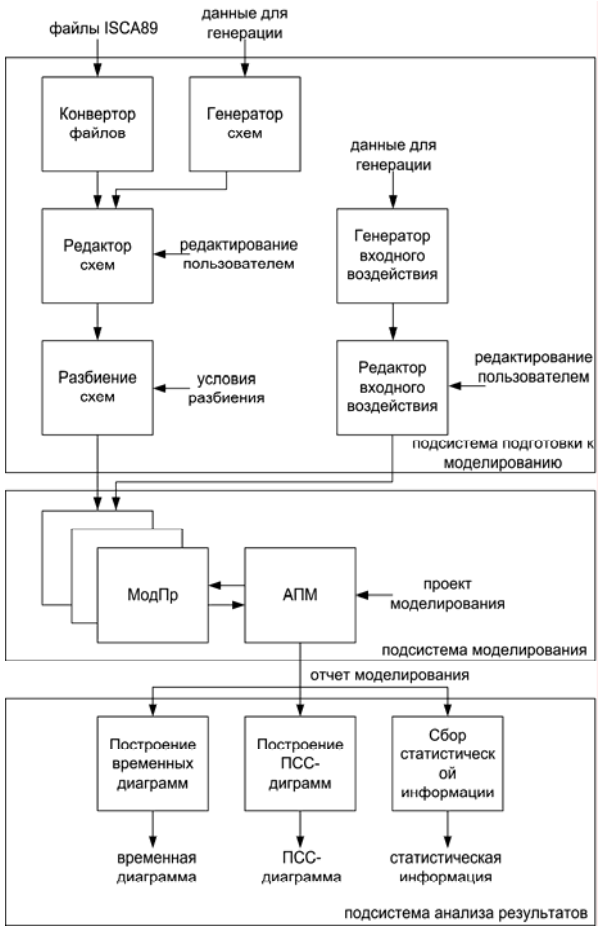

Рис.1. Структура программной системы для распределенного моделирования

Подсистема подготовки схемы и входного воздействия для моделирования предназначена для создания и редактирования схем и входного воздействия.

В подсистеме моделирования каждый моделирующий процессор выполняет моделирование заданного участка схемы с заданным воздействием. В результате моделирования создается отчет, который является входными данными для подсистемы анализа результатов моделирования. Эта подсистема позволяет построить временные диаграммы, диаграммы причинно-следственных связей и получить статистическую информацию из отчетов моделирований.

Программная система позволяет последовательно моделировать цифровые системы, либо выполнять распределенное моделирование с использованием консервативного, оптимистического, комбинированного и динамического алгоритмов.

2. Подсистема подготовки к моделированию

Подсистема подготовки входных данных для моделирования содержит следующие программные компоненты:

– конвертор файлов ISCAS89 в SSM-формат. В файлах внутреннего формата SSM программной системы содержится описание схемы;

– редактор многоуровневого описания схем. Редактор позволяет создавать новые элементы схем. Поведение этих элементов описывается таблицей истинности, либо наследуется из описания других схем. Готовые элементы могут использоваться для создания схем более высокого уровня;

– компонент для разбиения схемы на части, каждая часть моделируется на отдельном процессоре;

– генератор случайных схем позволяет создавать описание новых схем случайным образом. Набор параметров указывается в специальном файле проекта генератора схем.

– редактор входного воздействия;

– генератор случайного входного воздействия.

Исходными данными для моделирования являются файл входного воздействия и множество SSM файлов с описанием схемы. Все файлы имеют текстовый формат. Возможно добавление дополнительных программных компонент, расширяющих возможности по обработке и подготовке исходных файлов данных.

3. Подсистема моделирования

Подсистема моделирования состоит из следующих программных компонент:

– множество моделирующих процессоров (МодПр), каждый из которых реализует алгоритм синхронизации вычислений. Каждый моделирующий процессор запускается на отдельном компьютере;

– администратор процесса моделирования (АПМ), специальная программа для управления моделирующими процессами. Администратор позволяет осуществлять выбор схемы, входного воздействия, а также задавать необходимые опции для моделирования. После завершения моделирования администратор собирает отчеты всех моделирующий процессоров. Для обмена данными между администратором процесса моделирования и моделирующими процессами может использоваться локальная сеть или Интернет [4].

4. Подсистема анализа результатов моделирования

Результатом моделирования является файл отчета. Для формирования содержимого файла используется специальный язык диаграмм причинно-следственных связей (ПСС-диаграмм).

Подсистема анализа результатов моделирования содержит следующие программные компоненты:

- программа для построения временных диаграмм;

- программа для построения диаграмм причинно-следственных связей;

- программа для сбора статистической информации о процессе моделирования.

Диаграммы причинно-следственных связей предназначены для визуального представления процесса моделирования и исследования поведения протоколов синхронизации. ПСС-диаграмма содержит список переменных состояний для каждого процесса моделирования (например, значения сигналов в узлах схемы), список сообщений, переданных по сети; операции, выполненные процессором моделирования (например, обработка и генерация событий), физические значения временных отчетов.

Анализ ПСС-диаграмм позволяет найти ненужные задержки в процессе моделирования, избыточные сообщения, передаваемые по сети, определить наиболее времяемкие процедуры в реализации протокола синхронизации. Такие диаграммы также могут использоваться для упрощения процесса отладки любого распределенного приложения. Для визуального представления сигналов на входах и выходах схемы предназначены временные диаграммы. Использование временных диаграмм позволяет проверить корректность моделирования. Результат на временной диаграмме не зависит от использованного протокола синхронизации при моделировании и метода разбиения схемы на части.

5. Динамический протокол синхронизации

Особенностью динамического протокола, используемого в системе моделирования является возможность переключения от консервативного поведения к оптимистическому и обратно в процессе моделирования. Логический процесс (ЛП) автоматически изменят свой тип на консервативный в случае возникновения частых откатов, либо на оптимистический в случае частых блокировок при моделировании.

С целью повышения эффективности моделирования протокол синхронизации может быть реализован аппаратно. Структурная реализация динамического протокола синхронизации в виде специализированного процессора для распределенного моделирования рассматривается в [5].

Ниже приведены условные обозначения, используемые при описании алгоритмов, используемых при работе системы моделирования: $+e_i@t_i$ – положительное событие e_i в момент времени t_i; $-e_i@t_i$ – отрицательное событие e_i в момент времени t_i; IQ – очередь входных событий; H – очередь истории событий; RQ – очередь отрицательных событий; Si – текущее состояние логического процесса; optimism_degree – степень оптимизма при оптимистическом типе синхронизации; rollback_count – счетчик откатов; deadlock_count – счетчик тупиков; negaive – отрицательный тип события; positive – положительный тип события; safe – «безопасный» тип события (t_i < LVT); negative_event_sent – количество отправленных отрицательных событий; positive_event_simulated – количество обработанных положительных событий; LP – логический процесс; LP_kind – тип логического процесса.

6. Оптимистический режим работы

Алгоритм работы ЛП в оптимистическом режиме показан на рис. 2.

```
1  e_i@t_i = IQ.get_min();
2  if (e_i@t_i is positive);
3     IQ.extract_min();
4     if (t_i<H[last].t_i) rollback_cout++;
5     while (t_i<H[last].t_i);  //отставшее событие
6        IQ.reinserted(H[last].e_i); //положительный откат
7        rollback(H[last--]);
8        RQ.inserted(-e_i@t_i);
9     <S_o,{e_o@t_o}>:=simulate(S_i,e_i@t_i);
10     positive_event_simulated++;
11    if (t_o=RQ[first].t_i); // «ленивая» отмена событий
12       if (e_o ≠ RQ[first].e_i)
13          send negative pair {-e_i@t_i}
14          negative_event_sent++;
15          RQ.extract_first();
16    H[last++]:=<t_i,e_i,S_i,{e_o}>;
17    send {e_o@t_o};
18 else if (+e_i@t_i ∈ H); // отрицательный откат
19    IQ.extract_min();
20    while (+e_i@t_i ≠ H[last].e_i)
21       rollback(H[last--]);
22   rollback(H[last--]);
23       if (nega-
tive_event_sent/positive_event_simulated>N)
24          optimism_degree++;
25       else if
         (posi-
tive_event_simulated/negative_event_sent) >N
26          optimism_degree--;
```

Рис. 2. Алгоритм работы ЛП в оптимистическом режиме

Из очереди IQ выбирается очередное событие $e_i@t_i$. Если событие положительное и время t_i меньше чем значение локального времени (временной метки последнего промоделированного события), то e_i нарушает порядок локальной причинности событий. В результате значение rollback_count увеличивается на 1 и выполняется положительный откат, с использованием алгоритма «ленивой отмены» (строки 10-14, рис. 2). Для каждого промоделированного события j

из H со значением временной метки $H[j].t_i > t_i$, входное событие e_i заново вставляется в очередь IQ (строка 6, рис. 2). При этом отрицательное событие $-e_i@t_i$ для отмены неверного события e_i заносится в очередь RQ (строка 7, рис. 2). Далее выполняется моделирование события $e_i@t_i$ (строка 8, рис. 2), информация о промоделированном событии записывается в очередь H (строка 15, рис. 2). После моделирования очередного положительного события значение positive_event_simulated увеличивается на 1. При этом, если новое событие в момент времени t_o, соответствует старому в очереди $RQ[].t_o$(строки 10-11, рис. 2), то оно считается неверным. Для такого события формируется отрицательное событие (строка 12, рис. 2). После формирования и отсылки каждого отрицательного события значение negative_event_sent увеличивается на 1.

В случае, если событие $e_i@t_i$ отрицательное и в H есть соответствующее ему положительное событие $+e_i@t_i$ (строка 17, рис. 2), то соответствующая ему информация удаляется из очереди H.

7. Динамическое изменение степени оптимизма

Степень оптимизма определяет на сколько, в единицах виртуального времени, значение локального времени ЛП может превышать глобальное. При работе в режиме оптимистической синхронизации ЛП может изменяться значение степени оптимизма в процессе моделирования. В общем виде алгоритм показан на рис. 3.

В оптимистическом режиме работы после моделирования очередного положительного события значение счетчика positive_event_count увеличивается на 1, а при посылке отрицательного увеличивается значение счетчика negative_event_count.

```
1  while (simulation loop)
2  if (negative pair {-e_i@t_i} is sent)
3     negative_event_sent++;
4  else if (positive event {+e_i@t_i} is simulated)
5     positive_event_simulated++;
6  if (negative_event_sent /
   positive_event_simulated)>N
7     optimism_degree++;
8  else if (positive_event_simulated /
9  negative_event_sent) > N
10    optimism_degree--;
11 negative_event_sent:=0;
12 positive_event_simulated:=0;
```

Рис. 3. Алгоритм динамического изменения степени оптимизма

При этом выполняется следующая проверка: если отношение количества обработанных положительных событий и сформированных отрицательных событий превышает значение N, то выполняется инкремент значения optimism_degree.

В случае, если отношение отрицательных и положительных событий больше N, то выполняется декремент значения optimism_degree. Уменьшая или увеличивая значение N можно изменять скорость изменения степени оптимизма, в результате чего можно повысить скорость моделирования.

8. Консервативный режим работы

Алгоритм работы ЛП в консервативном режиме показан на рис. 4.

```
1  e_i@t_i:= IQ.get_min();
2  if (e_i@t_i is positive and safe)
3     IQ.extract_min();
4     <S_o,{e_o@t_o}>:=simulate(S_i, e_i@t_i);
5     event_simulated++;
6      send {e_o@t_o};
7  else if (deadlock is appeared)
8     deadlock_count++;
```

Рис. 4. Алгоритм работы ЛП в консервативном режиме

Если очередное обрабатываемое событие является положительным и безопасным, то выполняется его моделирование. Событие считается безопасным, если значение временной метки меньше значения глобального виртуального времени. Следует отметить, при работе в режиме переключения типа ЛП событие является безопасным не только если оно не нарушает порядок причинности, но и если оно получено от оптимистического ЛП. После моделирования очередного события значение счетчика event_simulated увеличивается на 1. Если в результате моделирования возникает тупик, то выполняется декремент значения счетчика deadlock_count.

9. Промежуточный режим

Переключение из консервативного режима в оптимистический происходит путем установки поля, отвечающего за тип ЛП. При переключении из оптимистического режима в консервативный ЛП в начале переходит в промежуточный режим. В этом режиме обрабатываются только отрицательные и безопасные события (строка 2, рис. 5). Когда в очереди H останется только одно положительное событие, ЛП переключается в консервативный режим. Такое действие необходимо, так как при оптимистическом поведении очередь H может содержать события, для которых может быть вызван откат в будущем.

```
1  e_i@t_i:= IQ.get_min();
2  if (e_i@t_i is negative or safe)
3      execute_optimistic();
4  if (e_i@t_i is positive and H.size = 1)
5  LP_kind:= conservative;
```

Рис. 5. Алгоритм работы ЛП
в промежуточном режиме

10. Механизм переключения типа ЛП

Переключение типа ЛП выполняется в соответствии с алгоритмом на рис. 6. Тип синхронизации при моделировании определяется значением LP_kind. Переключение в консервативный режим выполняется при условии, что отношение количества откатов rollback_count и количества промоделированных событий event_simulated больше значения switching_criterion_opt. Аналогично переключение в оптимистический режим происходит, если отношение количества тупиков deadllock_count больше значения switching_criterion_cons. Изменяя значения критериев переключения switching_criterion_opt и switching_criterion_cons можно получить оптимальное соотношение между временем работы ЛП в оптимистическом и консервативном режимах, т.е. повысить эффективность распределенного логического моделирования.

```
1  while (simulation is progress)
2      if (LP_kind) is optimistic
3          execute_optimistic();
4          if (rollback_count / event_simulated >
           switching_criterion_opt)
5              LP_kind:= transient;
6              rollback_count:=0;
7              event_simulated:=0;
8      else if (LP_kind is conservative)
9          execute_conservative();
10         if (deadlock_count / event_simulated >
           switching_criterion_cons)
11             LP_kind:= optimistic;
12             deadlock_count:= 0;
13             event_simulated:= 0;
14         else if (LP_kind is transient)
15             execute_transient();
```

Рис. 6. Алгоритм переключения типа ЛП

Заключение

Разработанная программная система позволяет выполнять распределенное моделирование цифровых систем, исследовать алгоритмы синхронизации, используемых при распределенном моделировании. Предложенный протокол динамической синхронизации позволяет повысить эффективность распределенного моделирования по сравнению с консервативным или оптимистическим протоколами.

Литература

1. Shi C.J.R., Lungeanu D. Distributed simulation of VLSI circuits via lookahead-free self-adaptive optimistic and conservative synchronization // Proc. ICAAD. – Nov 1999. – P. 500-504.

2. Lungeanu D., Shi C.-J.R. Parallel and distributed VHDL sim. // Proc. DATE. – March 2000. – P. 658-662.

3. Ladyzhensky Y.V., Popoff Y.V. Software system for event-driven logic simulation // IEEE EWDWT, Odessa, September 15-19, 2005. – P.119-122.

4. Ladyzhensky Y.V., Popoff Y.V. Architecture of internet access to distributed logic simulation system // IEEE EWDWT, Sochi, September 15-19, 2006. – P. 339-343.

5. Ладыженский Ю.В., Тесленко Г.А. Аппаратный метод повышения эффективности алгоритмов распределенного логического моделирования цифровых систем // Наукові праці ДНТУ. – Донецьк: ДонНТУ, 2006. – Вип. 106. – С. 77-81.

Поступила в редакцию 16.02.2007

Рецензент: д-р техн. наук, проф. В.С. Харченко, Национальный аэрокосмический университет им. Н.Е. Жуковского «ХАИ», Харьков.