# ТЕХНОЛОГИЯ СБОРА И АНАЛИЗА РЕЗУЛЬТАТОВ РАСПРЕДЕЛЕННОГО ЛОГИЧЕСКОГО МОДЕЛИРОВАНИЯ

Ладыженский Ю.В., Попов Ю.В.
Кафедра ПМиИ, ДонНТУ
ly@cs.dgtu.donetsk.ua

## *Abstract*

***Ladyzhensky Y.V., Popoff Y.V. A technology for gathering and analysis of distributed logical simulation results.*** *Analysis of synchronization protocols for logic simulations is fulfilled. Algorithms for experimental research of parallel simulations are developed. A system architecture is proposed. Software was implemented and tested in a local network.*

## *Введение*

Для повышения скорости и надежности распределенного моделирования дискретных логических схем необходимо исследовать работу различных протоколов синхронизации логических процессов при моделировании разных схем [1, 2], в зависимости от способов разбиения схемы и тестовых последовательностей. Для исследования необходима программная система, выполняющая моделирование, сбор информации о ходе моделирования и анализ полученных данных.

Анализ последних исследований показал, что в настоящее время для ускорения процесса моделирования исходную схему разрезают на части и каждую часть моделируют на отдельном процессоре. В [1] рассмотрены основные алгоритмы распределенного моделирования. Основное внимание уделено системе распределенного логического моделирования цифровых устройств с использованием консервативного протокола синхронизации. Не рассмотрены системы с оптимистическим протоколом синхронизации. В [2] предложена система для исследования методов параллельного логического моделирования ПЛИС. В [3] приведено сравнение существующих методов параллельного логического моделирования на мультипроцессорах с распределенной памятью. Основной теоретический материал, необходимый при программировании многопоточных, параллельных и распределенных программ рассмотрен в [4]. В работах не разработана архитектура программной системы и не в полном объеме решена задача разработки технологии сбора и анализа результатов распределенного логического моделирования (РЛМ).

Цели и задачи исследований, рассмотренных в статье, включают в себя разработку архитектуры программной системы для исследования

различных протоколов синхронизации при РЛМ. Должна быть рассмотрена реализация сервера анализа результатов распределенного логического моделирования. Сервер должен обеспечивать построение диаграмм причинно-следственных связей, временных диаграмм и обработку статистической информации. Должен быть разработан журнал отчета моделирующего процессора специального формата.

## *1. Архитектура программной системы для исследования протоколов синхронизации при РЛМ*

Архитектура программной системы для исследования различных протоколов синхронизации при РЛМ приведена на рис. 1. Программная система включает в себя следующие основные подсистемы [3]:

- подсистема ввода исходных данных;
- подсистема распределенного логического моделирования;
- сервер анализа результатов РЛМ.

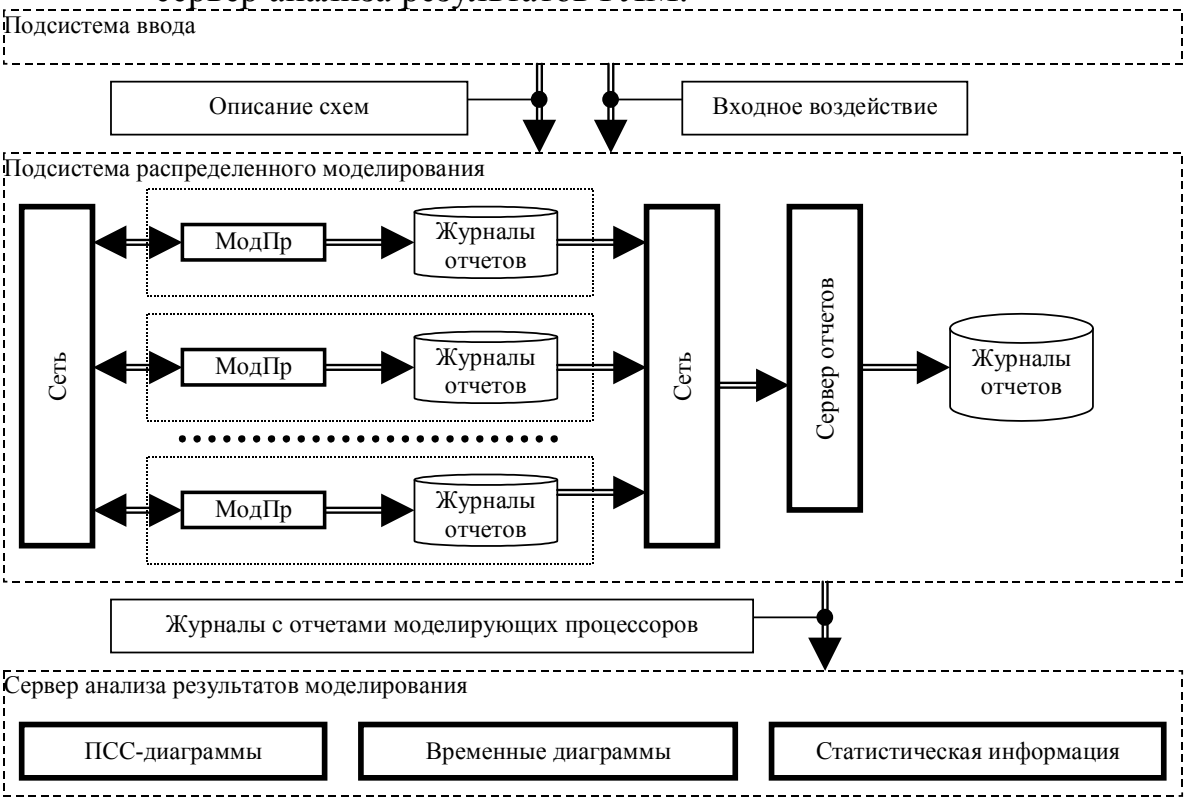


Рис. 1 - Архитектура программной системы для исследования различных протоколов синхронизации при РЛМ

Подсистема распределенного логического моделирования содержит набор моделирующих процессоров (МодПр), на которых происходит моделирование. Во время моделирования на каждом МодПр ведется собственный журнал отчета. После окончания моделирования журналы отчетов собираются на сервере отчетов для последующей передачи на сервер анализа результатов моделирования.

Сервер анализа результатов распределенного логического моделирования содержит следующие объекты:

- диаграммы причинно-следственных связей (ПСС-диаграммы);
- временные диаграммы;
- статистическая информация.

## *2. Журнал отчета моделирующего процессора*

Для анализа работы любой программной системы необходима информация о всех состояниях системы и обо всех событиях, которые произошли в этой системе. Задача анализа усложняется тем, что требуется анализировать работу распределенного приложения.

Одним из самых распространенных методов анализа работы последовательных программ является трассировка. Во время трассировки программа выполняется шаг за шагом. После выполнения очередного шага программа приостанавливается для анализа текущего состояния системы. Применение трассировки для анализа системы распределенного логического моделирования осложняется следующими факторами:

- задержки, возникающие при остановке во время трассировки одной копии программы могут существенным образом влиять на ход работы остальных копий программ;
- применение трассировки не позволяет выявлять причинно-следственные связи, возникающие при работе распределенного приложения;
- трассировка не дает информации о реальных задержках, возникающих при выполнении программы;
- большинство стандартных отладчиков не позволяют получить статистическую информацию о ходе выполнения программы;

Для того, что бы решить перечисленные выше задачи, можно в каждой копии МодПр во время моделирования записывать последовательность состояний в файл. Такой файл будем называть журналом отчета МодПр. Сформулируем основные требования к журналу отчета:

- задержки, возникающие в программе в результате использования журнала отчета, должны быть минимальными. Для этого необходимо минимизировать количество записываемой в журнал информации, записывать журнал в локальный файл в потоковом режиме и собирать в один файл только после завершения моделирования;
- журнал должен давать информацию о реальных задержках, возникающих во время работы программы. Для этого к каждой записи журнала добавляется реальная временная метка;
- формат журнала должен быть легко расширяем для исследования новых или модифицированных протоколов синхронизации логических процессов.

Для удобства работы файл журнала отчета создается в текстовом формате. Для минимизации информации в журнал отчета записываются только события, происходящие во время моделирования. Не будет вноситься информация о переменных состояния, которые не изменились. Во время моделирования могут возникнуть следующие основные типы событий:

- возникло событие в моделируемой схеме (внутреннее или внешнее);
- отправлено или получено сообщение по сети;
- произошла ошибка;
- изменилось состояние системы;
- пройдена контрольная точка программы.

Для формального описания журнала отчета введем следующие обозначения:

$Z_{i,P}^{j}$ - $i$-я запись $j$-го типа журнала отчета на процессоре $P$;

$T\left(Z_{i,P}^{j}\right)$ - реальное время добавления в журнал отчета $i$-й записи $j$-го типа на процессоре $P$.

Любой индекс может быть опущен. тогда $i$ означает номер записи в сквозной нумерации записей, без учета информации, соответствующей отсутствующим индексам. Например, $Z_i^j$ - $i$-я запись $j$-го типа.

Тип записи определяет, какого типа событие произошло, например отправлено сообщение-маркер или выбрано событие из локального списка событий. Реальное время добавления записей в журнал исчисляется в единицах реального времени, которое прошло после начала процесса моделирования. Записи в журнале для одного процессора приводятся в порядке их появления в реальном времени: $T\left(Z_k\right) \le T\left(Z_{k+1}\right)$.

В связи с погрешностями при измерении реального времени в журнале отчета могут появиться две разные записи с одинаковой реальной временной меткой. Поэтому неравенство является нестрогим. Однако записи с меньшим номером всегда добавляются раньше записей с большим номером, никакие две записи не могут быть добавлены одновременно.

При слиянии нескольких журналов в один файл все записи в нем перенумеровываются по порядку. Для того, что бы отличить данные разных процессоров, запись $Z_{0,P}$ всегда задает имя процессора $P$.

В МодПр имеется возможность настройки фильтров для внесения в журнал отчета только записей определенных типов. Установка фильтров для разрешения внесения записей всех типов ведет к большим погрешностям при измерении реального времени наступления событий и к большому размеру файла журнала отчета. Погрешности при измерении реального времени возникают в связи с затратами времени на запись информации в журнал. Запрет некоторых типов записей сокращает количество информации о ходе моделирования.

## *3. Диаграммы причинно-следственных связей*

Объекты диаграмм причинно-следственных связей (ПСС-диаграмм) предназначены для визуального представления результатов моделирования и графов причинно-следственных связей событий. ПСС-диаграммы предоставляют информацию о том, где, когда и какие события произошли, какие изменения в состоянии МодПр в связи с этими событиями имели место в удобном для восприятия виде. Анализ ПСС-диаграмм при последовательном и распределенном моделировании позволяет оценить адекватность моделирования, отсутствие ненужных задержек при моделировании и лишних сообщений в сети, определить наиболее времяемкие места в протоколе синхронизации логических процессов [4].

На ПСС-диаграммах представлены обработанные события, отправленные и полученные сообщения в сети, значения сигналов в узлах моделируемой схемы, значения виртуального и реального времени для каждого события, списки необработанных элементов. Для каждого события приводится полная информация о состоянии МодПр, на котором это событие произошло. Параллельно отображаются состояния остальных МодПр.

Горизонтальной осью на ПСС-диаграмме является ось реального времени $T$. Все события расположены на ПСС-диаграмме таким образом, что если провести на ней вертикальную линию, то все события, расположенные левее этой линии произошли в реальном времени не раньше событий, расположенных правее этой линии. Если один из процессоров простаивает, то на ПСС-диаграмме появляется пустое место. Распределение событий по ПСС-диаграмме производится с учетом следующих правил:

- для процессора $P_k$ все события записаны в порядке добавления в журнал: $\forall i, j, Pk : i > j \Rightarrow T\left(Z_{i,Pk}\right) > T\left(Z_{j,Pk}\right)$;
- для нескольких процессоров находятся записи о передаче сообщения по сети и его приеме. События располагаются таким образом, что бы событие, соответствующее приему сообщения, было всегда правее события, соответствующего отправке сообщения:

  $$\forall i, j, Pk, Pl : Z_{i,Pk}^{sent} \rightarrow Z_{j,Pl}^{received} \Rightarrow T\left(Z_{i,Pk}^{sent}\right) < T\left(Z_{j,Pl}^{received}\right)$$

  где $Z_{i,Pk}^{sent} \rightarrow Z_{j,Pl}^{received}$ означает, что на процессоре $Pk$ было отправлено сообщение, а на процессоре $Pl$ это же сообщение было получено.

Объект ПСС-диаграммы имеет следующие свойства:

- хранилище журнала отчета моделирующего процессора. Журнал отчета содержит исходные данные для построения ПСС-диаграммы;
- фильтр выводимых данных. При визуализации диаграммы можно указать, какие данные выводить: извлеченные/добавленные события в

локальный список событий; значения свойств; сообщения различных типов, переданные по сети;
- устройство графического вывода. В качестве такого устройства может использоваться экран или графический файл в формате Windows Metafile.

Для работы с ПСС-диаграммами имеются следующие методы:
- очистить хранилище журнала отчета моделирующего процессора. Этот метод применяется перед загрузкой нового журнала отчета;
- загрузить журнал отчета моделирующего процессора;
- установить фильтр выводимых данных;
- изобразить ПСС-диаграмму на экране. При этом ПСС-диаграмма выводится таким образом, что бы названия осей всегда оставались видимыми на экране;
- сохранить ПСС-диаграмму в графическом файле. При этом ПСС-диаграмма автоматически разрезается на несколько частей. Можно указать соотношение высоты и ширины рисунка в результирующем файле. Названия осей можно поместить в отдельный файл или в каждый сгенерированный файл.

Пример ПСС-диаграммы с комментариями приведен на рис. 2.

На рис. 2 можно обнаружить переданное по сети сообщение о событии c2=0@4. Это сообщение было передано от процессора P2 в момент времени 33 и было получено процессором P3 в момент времени 22, т.е. было получено раньше, чем было отправлено. Такое противоречие появляется в связи с погрешностью в измерении реального времени момента начала моделирования. Поскольку нет способа привязки к глобальным реальным часам, каждый процессор ведет отсчет времени от момента получения управляющего сообщения о начале моделирования. К процессору P3 это управляющее сообщение пришло на 11 единиц времени раньше, чем к процессору P2.

## *4. Временные диаграммы сигналов*

Временные диаграммы сигналов предназначены для визуального изображения последовательности сигналов в узлах и на выходах моделируемой схемы. Временная диаграмма является основным результатом моделирования. Однако она не зависит от способа моделирования, не будет изменяться при моделировании на нескольких или на одном процессоре. Анализируя временные диаграммы сигналов, можно оценить корректность моделирования, но, в отличие от ПСС-диаграмм, нельзя оценить скорость моделирования, выявить причинно-следственные связи, возникающие в ходе распределенного моделирования.

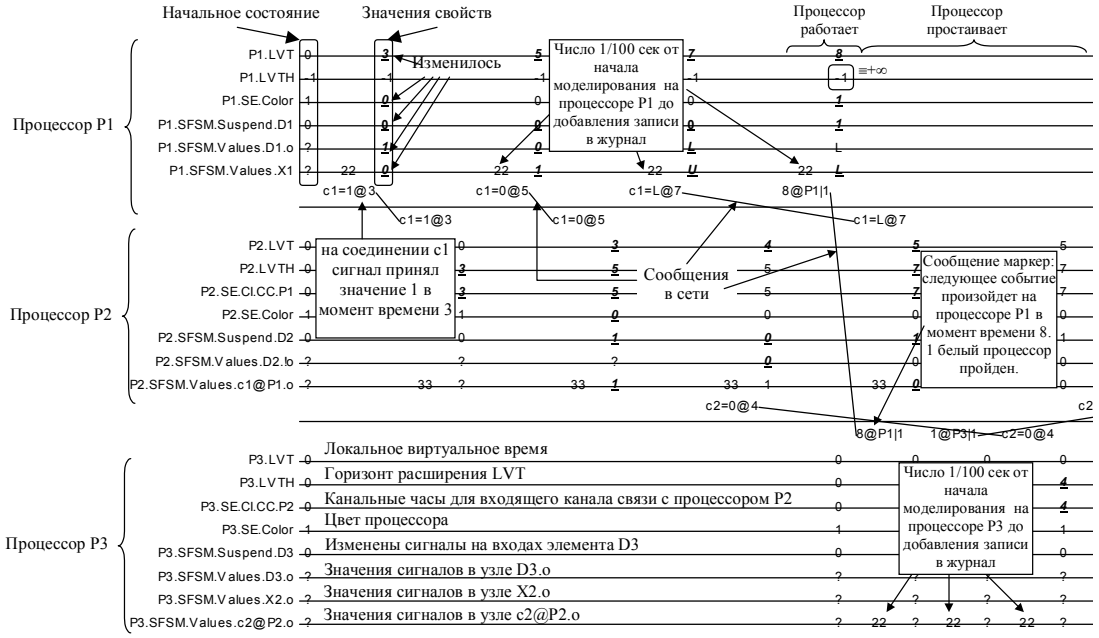


Рис. 2. Пример ПСС-диаграммы с комментариями

Объект Временная диаграмма имеет свойства:

- списки событий. Эти списки являются исходными данными для построения временных диаграмм;
- списки видимых источников событий, для которых будут изображаться временные диаграммы сигналов. Источниками событий могут быть изменения в узлах схемы; выходные сигналы; события, добавленные или извлеченные из локального списка событий; сообщения о событиях, переданные или принятые по сети;
- масштаб клеток диаграммы. Этим свойством задается соотношение высоты одной клетки диаграммы к ее ширине. Чем больше масштаб, тем шире будут клетки;
- название оси для каждого источника событий. По умолчанию в качестве названия выбирается полное имя узла схемы, который соответствует данному источнику события. Полное имя узла включает в себя имя процессора, имя элемента и имя выхода элемента, с которого подается сигнал на этот узел.

Пример временной диаграммы сигналов приведен на рис. 3.

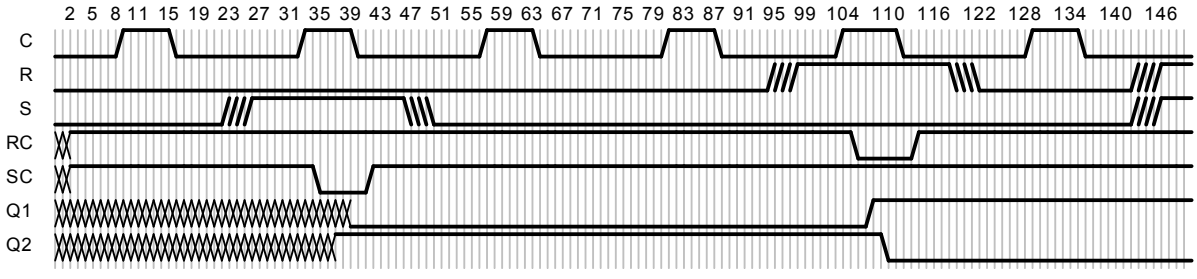


Рис. 3. Пример временной диаграммы сигналов

Для работы с временными диаграммами и их анализа имеются следующие методы:

- очистить списки событий. Применяется перед загрузкой нового журнала отчета;
- добавить в списки событий события из других источников событий. Этот метод можно применить, если нужно на одной диаграмме одновременно изобразить диаграммы сигналов для узлов разных схем, результаты моделирования которых записаны в разных файлах журналов отчетов моделирующих процессоров;
- отобрать источники событий. Вызывается непосредственно перед добавлением новых событий для указания нужных источников событий. Позволяет загружать в память только те события из журнала отчета, которые необходимы для дальнейшего анализа;
- спрятать все источники событий. Очищает списки видимых источников событий;
- вывести списки событий из данного источника событий и из всех видимых источников событий в выбранный момент виртуального времени;
- сохранить временную диаграмму сигналов в графических файлах. При этом диаграмма автоматически будет разрезана на несколько страниц. Размер страниц задается пользователем. Названия осей можно вывести на каждой странице либо на отдельной странице.

### *5. Статистическая информация*

Информация, получаемая в результате статистического анализа журналов отчетов моделирующих процессоров, позволяет:

- определять загрузку процессоров, анализировать распределение загрузки по процессорам;
- выделять “особые” записи журнала отчета: комментарии, записи с сообщениями об ошибках;
- определять узкие места в реализации протокола синхронизации: какие методы чаще вызываются, в каких происходит наибольшая средняя задержка в реальном времени; между какими парами событий происходили наибольшие задержки, какая точка в реализации протокола выполнялась чаще всего;
- определять общее число записей журнала по типам, например сколько событий было обработано всего и по каждому процессору; сколько сообщений было отправлено и получено по сети, сколько из них приходится на сообщение маркер, сколько – на сообщения о событиях; какие каналы связи были наиболее загружены.

В результате статистического анализа журнала отчета МодПр генерируется следующая информация:

- статистика реального времени, специфичная для консервативного протокола синхронизации логических процессов. Эта статистика содержит в себе информацию о загрузках всех процессоров. Для каждого процессора приводится: продолжительности всех периодов простоя $T_i^{idle}$ и обработки событий $T_i^{busy}$; общее время загрузки процессора

$$T^{busy} = \sum_i T_i^{busy};$$

общее время простоя процессора

$$T^{idle} = \sum_i T_i^{idle};$$

общая время решения задачи моделирования:

$$T^{total} = T^{busy} + T^{idle}.$$

Время простоя включает в себя ожидание прихода сообщения из сети и действия, не связанные с обработкой событий. Пример расчета $T_i^{busy}$ и $T_i^{idle}$ рассмотрен на рис. 4;

- “особые” записи журнала отчета, содержат сообщения об ошибках и комментарии, обнаруженные в журнале отчета. Такая информация используется для выявления особых ситуаций, которые возникли во время моделирования;
- общее число записей журнала отчета, классифицированные по процессорам;
- статистика реального времени содержит суммарное время $T$ и среднее время $t$ реальных задержек между добавлениями в журнал записей. Информация приводится по всем возможным парам ($T^{i,j}$, $t^{i,j}$), например, “какая средняя задержка между получением и отправкой маркера”. По каждому типу записи $Z^i$ указывается задержка перед добавлением ($T^{,i}$,$t^{,i}$) и после добавления ($T^i$,$t^i$), например “какое среднее время проходит после выполнения очередной операции перед началом обработки события”.

Суммарное время $T^{i,j}$, которое прошло от момента $T(Z^i)$ добавления в журнал записи $Z^i$ типа $i$ до момента $T(Z^j)$ добавления в журнал записи $Z^j$ типа $j$ считается по формуле:

$$T^{i,j} = \sum_k^{count(Z^i)} \left(T\left(Z_l^j\right) - T\left(Z_k^i\right)\right)$$

где $count(Z^i)$ - количество записей $i$-го типа;

$Z_l^j$ - запись $j$-го типа, которая находится непосредственно перед $Z_k^i$.Учитываются только те записи $Z_l^j$, после которых нет других записей типа $i$: $\nexists m : T(Z_l^j) < T(Z_m^i) < T(Z_k^i)$.

Суммарное время $T^{,i}$, которое прошло непосредственно перед добавлением в журнал записей типа $i$, считается по формуле:

$$T^{,i} = \sum_{k}^{count(Z^i)} \left( T(Z_k^i) - T(Z_{k-1}) \right)$$

Суммарное время $T^i$, которое прошло непосредственно после добавления в журнал записей типа $i$, считается по формуле:

$$T^i = \sum_{k}^{count(Z^i)} \left( T(Z_{k+1}) - T(Z_k^i) \right)$$

### *Заключение*

Предложенную архитектуру программной системы можно использовать для исследования различных протоколов синхронизации при распределенном логическом моделировании. Разработанный журнал отчета моделирующего процессора специального формата позволяет собирать полезную информацию о ходе моделирования. Предложенные средства анализа журнала отчета обеспечивают построение диаграмм причинно-следственных связей, построение временных диаграмм сигналов и обработку статистической информации. Детальное исследование ПСС-диаграмм позволяет определить наиболее узкие места в реализации протокола синхронизации и ошибки в реализации, которые не могут быть выявлены лишь путем анализа временных диаграмм сигналов. Предложенную технологию сбора и анализа результатов распределенного логического моделирования можно использовать для исследования протоколов синхронизации логических процессов с целью их последующей оптимизации.

Перспективным направлением дальнейших исследований является постановка и детальное исследование реальных экспериментов. Для постановки экспериментов могут потребоваться программные системы для автоматического синтеза тестовых схем и входных воздействий, для разрезания готовых или синтезированных схем на несколько частей разными способами.

```
Z1  1 Processor=P1          ;с записи "Processor=" начинается расчет загрузки
Z2  2 SE.Sync=CM
Z4  4 SE.Executing=0        ;статистика строится по записям "SE.Executing"
Z5  7 SE.Executing=0        ;  учитываются только записи, соответствующие
Z6  11 SE.Marker.Receive=2@P1|1 ;  изменению состояния:
Z7   16  SE.Executing=0                ;   данная запись игнорируется, т.к.
состояние
Z8   22  SE.Marker.Receive=2@P1|1 ;                                  "простой"
сохраняется
Z9  29 SE.Marker.Send=2@P1|1
Z10 37 SE.Executing=0
Z12 37 SE.Executing=0
Z13 46 SE.Executing=1             ; T1idle = T(Z13) - T(Z1) = 46 - 1 = 45
Z14 56 EVL.Add=1@2
Z15 67 SE.Executing=0             ; T1busy = T(Z15) - T(Z13) = 67 - 46 = 21
Z16 79 SE.Marker.Receive=2@P1|1
Z17 106 SE.Executing=1            ; T2idle = T(Z17) - T(Z15) = 106 - 67 = 39
Z18 106 SE.Executing=0            ;эта запись не учитывается, поскольку
Z19 106 SE.Marker.Receive=2@P1|1 ; процессор работал нулевой отрезок времени
Z20 106 SE.Executing=1
Z22 121 SE.Executing=0            ; T2busy = T(Z22) - T(Z17) = 121 - 106 = 15
Z23 137 SE.Marker.Receive=2@P1|1
Z24 154 SE.Executing=1            ; T2idle = T(Z24) - T(Z22) = 154 - 121 = 33
Z25 172 EVL.Add=1@2
Z26 211 SE.Executing=0            ; T3.1busy = T(Z26) - T(Z24) = 211 - 154 = 57
Z27 211 SE.Executing=1
Z28 211 EVL.Add=1@2
Z29 211 SE.Executing=0            ;запись не учитывается, поскольку
Z30 211 SE.Marker.Receive=2@P1|1 ;процессор работал нулевой отрезов времени
Z31 211 SE.Executing=1
Z32 232 SE.Executing=1            ;все эти записи учитываются как один
Z33 255 EVL.Add=1@2               ;этап работы процессора, соответствующий обработке
Z34 279 SE.Executing=1            ;событий в течение T3busy = T3.1busy + T3.2busy = 57 + 146 = 203
Z35 304 EVL.Add=1@2               ;  соседние записи "работал"-"работал" и
Z36 330 EVL.Add=1@2               ;  "простаивал"-"простаивал" объединяются в
Z37 357 SE.Executing=1            ;  в одну запись "работал" или "простаивал"
Z40 357 SE.Executing=1
Z42 357 EVL.Add=1@2
Z43 357 SE.Executing=1            ;  T3.2busy = T(Z43) - T(Z26) = 357 - 211 = 146
```

**(а)**

---

```
P1.Idle=117, 45 39 33   ;117 – общее время простоя на процессоре P1
P1.Busy=239, 21 15 203  ;239 – общее время обработки на процессоре P1
P1.Total=356            ;356 – общее время работы процессора P1
                        ;процессор P1 работал в следующем режиме:
                        ;45 ед.времени (с момента 0 по 45) простаивал
                        ;21 ед. времени (46 ..66 ) обрабатывал события
                        ;39 ед. времени (67 ..105) простаивал
                        ;15 ед. времени (106..120) обрабатывал события
                        ;33 ед. времени (121..153) простаивал
                        ;203 ед. времени (154..356) обрабатывал события
```

**(б)**

Рис. 4. Расчет $T_i^{busy}$ и $T_i^{idle}$: пример журнала отчета моделирующего процессора (а) и соответствующего ему файла с результатами расчета (б)

### *Литература*

1. Ладыженский Ю.В., Попов Ю.В. Система распределенного логического моделирования цифровых устройств с использованием консервативного протокола синхронизации // Наукові праці Донецького національного технічного університету. Серія: інформатика, кібернетика та обчислювальна техніка, випуск 39: - Донецьк: ДонНТУ, 2002. - 282 с. – с. 21-29.
2. Ладыженский Ю.В., Онищенко Д.В. Система для исследования методов параллельного логического моделирования ПЛИС // Моделювання та інформаційні технології. Збірник наукових праць. – Київ: Національна академія наук України. Інститут проблем моделювання в енергетиці імені Г.Є Пухова, 2002. – Випуск 12. – с. 64-70.
3. Luksch Peter. Evaluation of three Approaches to Parallel Logic Simulation on a Distributed Memory Multiprocessor. – Department of Computer Science. – Munich University of Technology (TUM). – 1993. – 11 p.
4. Эндрюс Г.Р. Основы многопоточного, параллельного и распределенного программирования: Пер. с англ. – М.: Издательский дом “Вильямс”, 2003. – 512 с.: илл.

Дата надходження до редколегії: 4.12.2003 р.