УДК 681.3.012

В. И. Волчихин, Н. П. Вашкевич, Р. А. Бикташев

ПЛАНИРОВЩИК ЗАДАЧ С АППАРАТНОЙ ПОДДЕРЖКОЙ ДЛЯ МНОГОПРОЦЕССОРНЫХ СИСТЕМ

Аннотация. Представлены результаты исследований по аппаратно-программной реализации планировщика задач для многопроцессорных операционных систем с пространственным разделением задач. Представлено формальное описание алгоритмов синхронизации взаимодействующих процессов при планировании задач на основе аппарата недетерминированных автоматов, предложен вариант структурной и функциональной реализации планировщика. Проведено моделирование алгоритмов на языке VHDL и проанализированы полученные результаты.

Ключевые слова: многопроцессорная операционная система, планирование задач, недетерминированные автоматы, формализация алгоритмов, синхронизация процессов.

Abstract. The paper presents research results of hardware and software implementation of the task scheduler for multiprocessor operating systems with spatial separation of tasks. The article adduces formal description of synchronization algorithms of interacting processes for scheduling tasks based on the nondeterministic automata apparatus. The authors suggest a variant of structural and functional implementation of the scheduler. The algorithms have been simulated with the VHDL language and the results have been analyzed.

Key words: multiprocessor operating system, task scheduling, non-deterministic automata, formalization of algorithms, synchronization of processes.

Введение

Традиционные операционные системы вносят значительные накладные расходы на выполнение трудоемких функций операционных систем и, в частности, на синхронизацию процессов, связанных с планированием задач, поскольку реализуются программным способом в пространстве ядра с применением механизмов критических секций, семафоров, рандеву, мониторов и др. [1]. Такие способы хорошо отработаны, но требуют значительных временных затрат на выполнение системных вызовов. Например, вхождение процесса в монитор и реализация очереди блокированных процессов, возникающей из-за конкуренции множества процессоров при доступе к планировщику, требуют выполнения программных прерываний, которые существенно увеличивают время ожидания прикладных задач (процессов) и существенно снижают общую производительность многопроцессорной системы. Известно, что на выполнение планирования уходит до десятка тысяч процессорных тактов, что соответствует временным потерям в несколько микросекунд [2].

Системные затраты времени можно значительно уменьшить применением аппаратной поддержки для выполнения функции синхронизации, связанной с планированием задач, что приведет к увеличению коэффициента использования процессоров, сокращению времени ответа, надежности и безопасности систем управления.

Существует два основных способа построения планировщиков задач в многопроцессорных системах: с разделением времени и разделением пространства [3]. Первый способ предполагает использование глобальной очере-

ди готовых к обработке задач, второй – локальной очереди для каждого процессорного узла.

В планировщиках с разделением времени существует явление перезагрузки кэш-памяти, связанное с переключением задач, когда прерванная задача с высокой вероятностью может быть направлена на продолжение обслуживания в другой процессорный узел. Названное явление увеличивает частоту кэш-промахов и неизбежно приводит к снижению производительности многопроцессорной системы [3].

В планировщиках с разделением пространства у каждого процессора имеется своя очередь задач, причем планировщик взаимодействует только с одним процессором и его функция ограничивается выборкой очередной задачи и назначения ее освободившемуся процессору. При прерываниях по истечении кванта (переключениях контекста) задача остается в той же очереди, в которой она находилась ранее. Таким образом, в вычислительной системе действует одновременно N планировщиков, в результате чего задачи выбираются из очередей бесконфликтно, поступают в процессоры параллельно, что создает условия для повышения производительности.

Аппаратная поддержка алгоритма планирования задач реализована в части синхронизации взаимодействующих процессов. Другие функции, такие как сохранение и восстановление контекста, перемещение процессов из очереди готовых задач в очередь ожидающих и т.п., реализуются традиционным путем и в данной модели не рассматриваются.

1. Структурная организация планировщика

Предлагаемый планировщик задач является распределенным и выполняется в виде независимого аппаратного устройства в составе многопроцессорной системы (рис. 1).

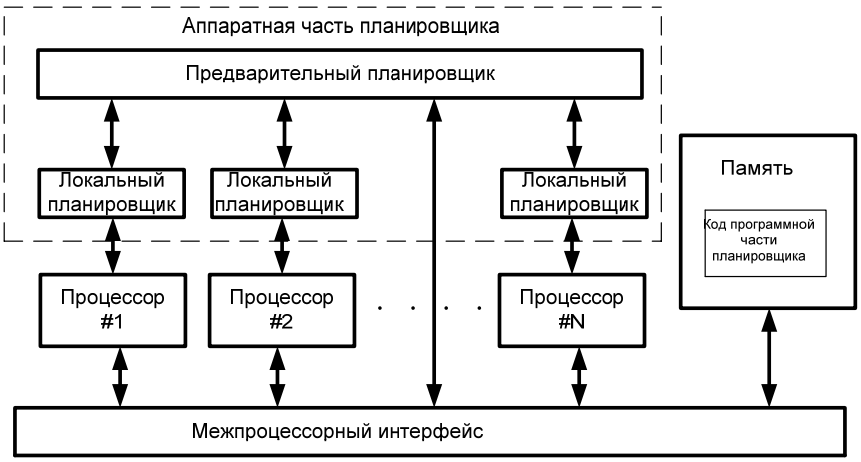

Рис. 1. Схема подключения планировщика задач
с аппаратной поддержкой в многопроцессорную систему

Предполагается, что аппаратный планировщик содержит две схемы: общий предварительный планировщик и локальный планировщик. Первый принимает извне новую задачу и передает ее локальному планировщику. Для реальной системы это означает, что в новой задаче выделяются разделы па-

мяти, присваивается соответствующий идентификатор, который помещается в очередь одного из локальных планировщиков.

Модель многопроцессорной системы с использованием аппаратного планировщика представлена на рис. 2. Предварительный планировщик является общим для всех локальных планировщиков. Он выполняет функцию выбора по заданному критерию обслуживающего процессора и передачи в него идентификатора новой задачи. Если процессор занят обслуживанием текущей задачи, то новая задача устанавливается в очередь локального планировщика. Если все очереди локальных планировщиков заполнены, то новая задача не допускается к обслуживанию. Каждой задаче выделяется квант процессорного времени. Если процессор, обслуживающий задачу, выполнил ее за предоставленный квант, она удаляется из системы, иначе возвращается на дообслуживание в тот же процессор.

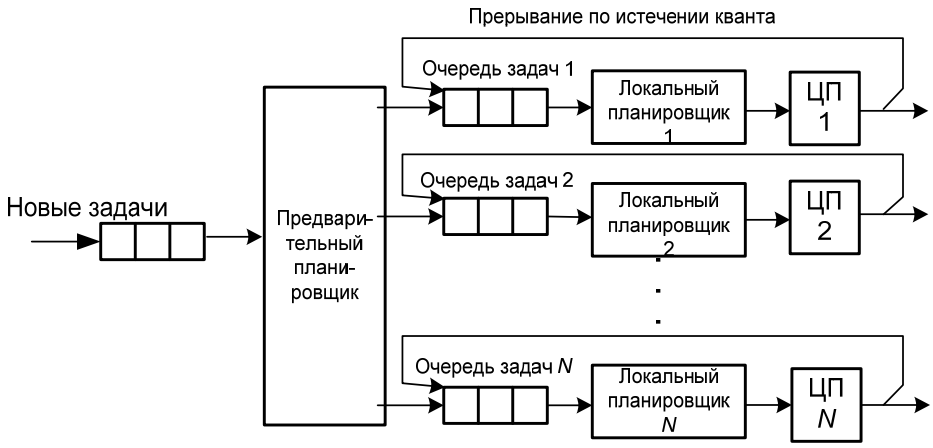

Рис. 2. Модель многопроцессорной системы, использующей планировщик с пространственным разделением

Первоначально поступившая задача помещается в очередь планировщика типа FIFO и находится в ней до тех пор, пока не получит необходимые ей ресурсы, в число которых входит и место в очереди к одному из процессоров. Если имеется свободное место в одной из очередей, то выбранная планировщиком задача занимает его, причем число мест в очереди ограничено некоторым числом. Принятая на обслуживание задача находится в очереди до тех пор, пока не поступит на выполнение в процессор. По окончании кванта текущая задача прерывается, после чего планировщик просматривает локальную очередь, и если в ней имеются заявки на обслуживание, то назначается на выполнение задача, стоящая в голове списка. Незавершенная задача по окончании кванта помещается в конец той же очереди, где она находилась ранее. Если задача завершена, то результат выдается пользователю, а в очереди освобождается одно место. Если очередь пуста, процессор переходит в режим ожидания.

2. Формализация алгоритмов синхронизации

Алгоритм синхронизации при планировании аналогичен задаче «спящего парикмахера» [3, 4]. Он содержит клиентскую и серверную части, взаимодействующие с общим ресурсом (локальной очередью задач), причем кли-

ентская часть осуществляет постановку задачи в очередь, а серверная – выбирает задачу из головы очереди и запускает ее на обслуживание в процессор. Действия клиента и сервера синхронизируются по схемам «писатели-читатели» и «рандеву», причем первая обеспечивает корректную работу с очередью задач как с общим ресурсом, вторая фиксирует момент наличия выбранной задачи из очереди и момент готовности процессора к обслуживанию этой задачи.

Формальное описание алгоритма взаимодействия процессов в данной задаче базируется на использовании моделей недетерминированных автоматов [5]. Модели представляются в виде систем канонических уравнений, описывающих все реализуемые события управляющего алгоритма [6].

Для описания алгоритма введены основные частные события:

S_I^t, S_S^{pj} – события, свидетельствующие о том, что поступила на обслуживание новая задача и имеется свободный процессор соответственно;

S_{FQ}^t – событие, свидетельствующее о том, что в очереди имеются свободные места;

S_Q^t, S_{SZ}^{pj} – события, свидетельствующие о том, что в очереди имеется хотя бы одна задача и хотя бы один ожидающий процессор соответственно;

S_{ZPj}^t, S_{PZ}^{pj} – события, свидетельствующие о том, что от планировщика поступил запрос задачи от j-го процессора, а j-й процессор выдал подтверждение запроса планировщику соответственно;

S_{GPj}^t, S_{PT}^{pj} – события, свидетельствующие о том, что задача готова к обслуживанию, а j-й процессор задачу принял соответственно;

S_A^{pj}, S_E^{pj} – события, свидетельствующие о том, что j-й процессор выполняет задачу и выполнение ее закончено соответственно;

S_{PK}^{pj} – событие, свидетельствующее о том, что произошло переключение контекста по истечении кванта процессорного времени;

S_{TO}^t, S_{RT}^{pj} – события, свидетельствующие о том, что необходимо снять задачу с исполнения и выдать результат пользователю соответственно;

S_{OF}^t – событие, свидетельствующее о том, что необходимо удалить задачу из очереди.

На основании словесно представленного алгоритма управления процессами и введенных событий, реализуемых в этом алгоритме, система канонических уравнений, описывающих эти события, будет иметь следующий вид:

– для процесса «клиент», реализуемого локальным планировщиком до момента рандеву:

$$S_Q^t(t+1) = S_I^t\, S_{FQ}^t \vee S_A^{pj}\, \overline{S_E^{pj}}\, S_{PK}^{pj} \vee S_Q^t\, \overline{S_{SZ}^{pj}};$$

$$S_{ZPj}^t(t+1) = S_Q^t\, S_{SZ}^{pj} \vee S_{ZPj}^t\, \overline{S_{PZ}^{pj}};$$

$$S_{GPj}^t(t+1) = S_{ZPj}^t\, S_{PZ}^{pj} \vee S_{GPj}^t\, \overline{S_{PT}^{pj}}. \qquad (1)$$

– для процесса «сервер», реализуемого процессором до момента рандеву:

$$S_{SZ}^{pj}(t+1) = S_S^{pj} \vee S_{SZ}^{pj}\ \overline{S_Q^t};$$

$$S_{PZ}^{pj}(t+1) = S_{SZ}^{pj}\ S_Q^t \vee S_{PZ}^{pj}\ \overline{S_{ZPj}^t};$$

$$S_{PT}^{pj}(t+1) = S_{PZ}^{pj}\ S_{ZPj}^t \vee S_{PT}^{pj}\ \overline{S_{GPj}^t}. \qquad (2)$$

Система канонических уравнений, описывающая события после рандеву:

$$S_A^{pj}(t+1) = S_{PT}^{pj}\ S_{GPj}^t \vee S_A^{pj}\ \overline{S_{PK}^{pj}};$$

$$S_{TO}^t(t+1) = S_A^{pj}\ S_E^{pj}\ S_{PK}^{pj} \vee S_{TO}^t\ \overline{S_{RT}^{pj}};$$

$$S_{RT}^{pj}(t+1) = S_A^{pj}\ S_E^{pj}\ S_{PK}^{pj} \vee S_{RT}^{pj}\ \overline{S_{OF}^t};$$

$$S_{OF}^t(t+1) = S_{RT}^t\ S_{TO}^{pj};\ S_S^{pj}(t+1) = S_{RT}^{pj}\ S_{OF}^t. \qquad (3)$$

Уравнениям соответствует граф недетерминированного автомата (НДА) (рис. 3), содержащий клиентскую и серверную части алгоритма управления взаимодействующими процессами до момента «рандеву» (до оператора объединения *J*(&)) и после «рандеву».

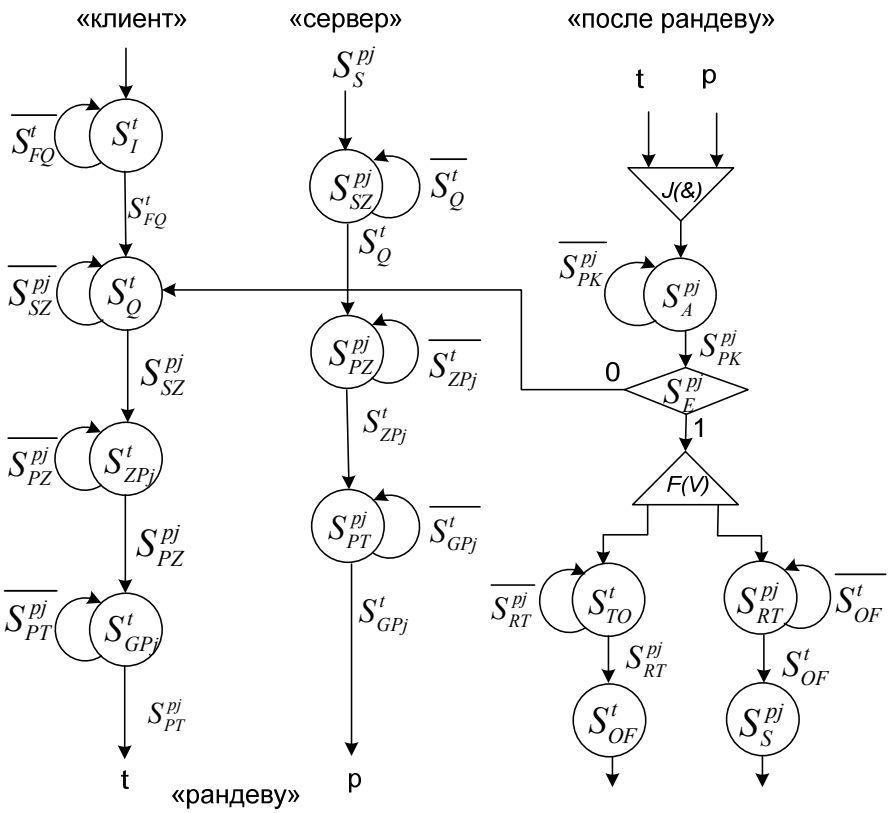

Рис. 3. Граф НДА алгоритма управления взаимодействующими процессами в планировщике с пространственным разделением задач

3. Функциональная организация планировщика

Интерфейс взаимодействия устройства локального планировщика с каждым из процессоров и с предварительным планировщиком представлен на рис. 4.

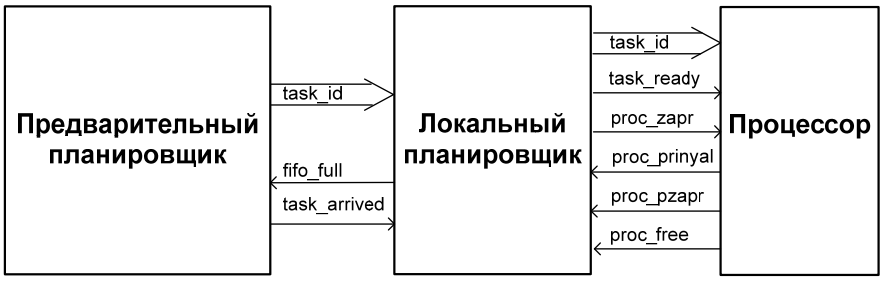

Рис. 4. Интерфейс взаимодействия локального планировщика с предварительным планировщиком и процессором

Функционирование взаимодействующих устройств по предложенному интерфейсу происходит следующим образом. Первоначально задача, получаемая от пользователя, заносится в вычислительную систему под управлением программного планировщика (на рис. 3 не показан). Он же присваивает задаче идентификатор и направляет ее в предварительный планировщик. При получении новой задачи предварительный планировщик пытается поместить ее в один из локальных планировщиков. Если в очереди к локальному планировщику имеется задача, ожидающая обслуживания, то он при условии, что процессор свободен, устанавливает активный уровень сигнала «Запрос» на линии (*proc_zapr*). Процессор выходит из спящего режима и сообщает локальному планировщику о том, что готов к приему и выполнению задачи. Для этого процессор устанавливает активный уровень сигнала «Подтверждение запроса» (*proc_pzapr*), который воспринимает локальный планировщик. Кроме того, процессор снимает сигнал «Свободен» (*proc_free*). Далее локальный планировщик переходит к непосредственной передаче идентификатора задачи процессору. Для этого он извлекает идентификатор задачи из очереди и выставляет его на шину данных процессора (*task_id*), после чего передает в процессор сигнал «Задача готова» (*task_ready*). Процессор воспринимает активный уровень сигнала «Задача готова», тем самым ему гарантируется, что данные на шине достоверны (*task_id*). Процессор фиксирует эти данные в своем внутреннем регистре, и, когда его работа с шиной будет окончена, он выставляет сигнал «Задача принята» (*proc_prinyal*).

Процессор, не занятый обслуживанием задачи (свободный), находится в спящем режиме. Сигнал «Свободен» (*proc_free*), поступающий от процессора к локальный планировщику, принимает активный уровень. Если в очереди имеются готовые задачи, процессор выходит из спящего режима, выбирает задачу из начала очереди (*fifo_read*), получает идентификатор задачи по шине (*task_id*), выставляет сигнал «Задача принята» (*proc_prinyal*) и переходит в режим «Занято». По завершению работы процессор известит локальный планировщик о том, что свободен, установкой активного уровня сигнала «Свободен» (*proc_free*).

Функциональная схема планировщика (рис. 5) состоит из трех блоков. Блок очереди предназначен для хранения задач. Запись в очередь производит предварительный планировщик, а считывание из очереди – процессор, поэтому сигнал записи *wrreq* направляется к каждому блоку очереди. Входная шина данных для передачи идентификатора задач (*task_id*) формируется блоком планирования и является общей для всех блоков очередей. Сигнал для чтения из блока очереди *rdreq* поступает от процессора, причем каждый процессор управляет только своей очередью.

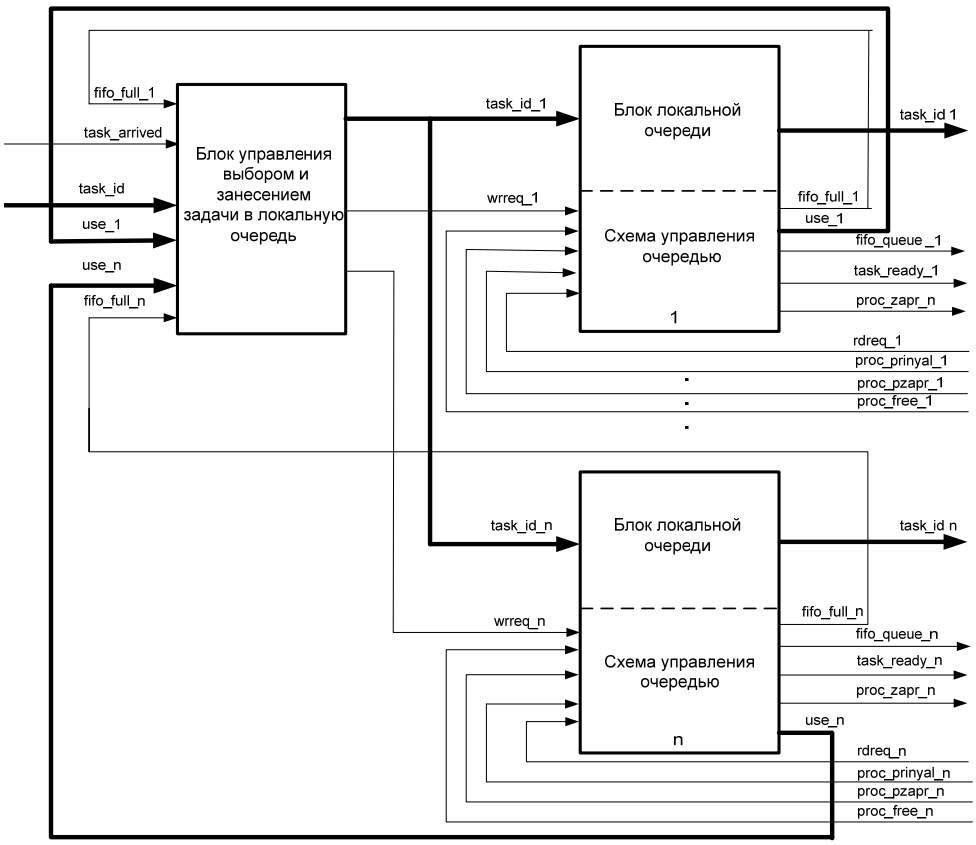

Рис. 5. Функциональная схема планировщика

Кроме того, каждый процессор получает от своего блока очереди сигнал о наличии идентификатора задачи (*fifo_empty*). Если в очереди имеется хотя бы один идентификатор задачи, процессор выбирает его, иначе переходит в режим ожидания. Блок очереди соединяется со «своим» процессором с помощью выходной шины данных. Сигнал *fifo_full* используется блоком управления выбором и занесением задачи в локальную очередь. Он указывает на то, что очередь заполнена, и локальный планировщик перестает ее рассматривать для записи новой задачи.

Схема управления очередью представляет собой управляющий автомат, функционирующий в соответствии с логическими выражениями (1) и (2).

Блок выбора и занесения по заданному критерию выбирает очередь, в которую будет помещен новый идентификатор задачи. В данной работе в

качестве критерия выбора принято минимальное число занятых мест в одной из всей совокупности очередей. Сигнал выбора *wrreq*[*i*] (записи в выбранную очередь) формируется на основании данных, получаемых от локальных планировщиков по шинам *use*[*i*] (*i* = 1, …, *n*). Причем на шине *use*[*i*] фиксируется значение числа занятых ячеек памяти, т.е. указывается, сколько мест занято в соответствующей очереди. Это значение формируется счетчиком записи, входящим в состав схемы управления очередью. Над значениями на шинах *use*[*i*] производится операция сравнения на минимум, по результатам которой формируется двоичный код, указывающий на очередь с минимальным числом занятых мест.

Блок выбора и занесения выбирает из предварительной очереди идентификатор новой задачи *task_id* и под управлением сигнала *task_arrived* осуществляет попытку произвести запись идентификатора в один из локальных блоков очередей. Передача идентификатора осуществляется под управлением сигнала *wrreq* по выходной шине данных *task_id*. Также на этот блок возлагается функция контроля состояния всех процессоров. Если один процессор выходит из строя, то запись новой информации в его очередь следует приостановить.

В табл. 1 представлено соответствие событий на графе алгоритма управления взаимодействующими процессами и сигналов на функциональной схеме планировщика задач.

Таблица 1

Событие	Сигналы на схемах	Событие	Сигналы на схемах
S_I^t	*task_arrived*	S_A^{pj}	Аппаратно не реализовано
S_{FQ}^t	*fifo_full*	S_{PK}^{pj}	---------\\--------
S_Q^t	*task_in_queue*	S_{OF}^t	---------\\--------
S_{ZPj}^t	*proc_zapr_j*	S_{TO}^t	---------\\--------
S_{GPj}^t	*task_ready_j*	S_E^{pj}	---------\\--------
S_{PZ}^{pj}	*proc_pzapr_j*	S_{RT}^{pj}	---------\\--------
S_S^{pj}	*proc_free*	S_{SZ}^{pj}	---------\\--------
S_{PT}^{pj}	*proc_prinyal_j*		

Заключение

Модель устройства планировщика была реализована на языке VHDL в виде четырех программных модулей. Ввод схем и работа производились в свободно распространяемой версии системы Quartus II Version 5.1 Service Pack 1, моделирование производилось в той же системе. Для проведения моделирования использовались два дополнительных блока: блок генерации задач и блок имитации работы процессора.

Модель была протестирована в различных режимах работы и показала высокие результаты по производительности. Полученное значение задержки назначения задачи при условии отсутствия очереди в процессорном узле не превышает одного такта. Таким образом, реализация аппаратного планиров-

щика с пространственным разделением задач создает условия для существенного повышения производительности многопроцессорной системы.

Список литературы

1. **Вашкевич, Н. П.** Аппаратная поддержка диспетчера задач с глобальной очередью в многопроцессорных системах / Н. П. Вашкевич, Р. А. Бикташев, А. И. Меркурьев // Известия высших учебных заведений. Поволжский регион. Технические науки. – 2011. – № 3. – С. 3–14.
2. **Стивенс, У.** UNIX: взаимодействие процессов / У. Стивенс – СПб. : Питер, 2003. – 576 с.
3. **Таненбаум, Э.** Современные операционные системы / Э. Таненбаум. – 2-е изд. – СПб. : Питер, 2002. – 1040 с.
4. **Эндрюс, Г. Р.** Основы многопоточного параллельного и распределенного программирования : пер. с англ. / Г. Р. Эндрюс. – М. : Вильямс, 2003. – 512 с.
5. **Вашкевич, Н. П.** Недетерминированные автоматы в проектировании систем параллельной обработки : учеб. пособие / Н. П. Вашкевич. – Пенза : Изд-во Пенз. гос. ун-та, 2004. – 280 с.
6. **Вашкевич, Н. П.** Формализация алгоритма синхронизации процессов при диспетчеризации задач в многопроцессорных системах с использованием механизма рандеву / Н. П. Вашкевич, Р. А. Бикташев // Информационные технологии. – 2009. – № 12. – С. 12–17.

Волчихин Владимир Иванович
доктор технических наук, профессор, ректор Пензенского государственного университета

E-mail: rectorat@pnzgu.ru

Volchikhin Vladimir Ivanovich
Doctor of engineering sciences, professor, rector of Penza State University

Вашкевич Николай Петрович
доктор технических наук, профессор, кафедра вычислительной техники, Пензенский государственный университет

E-mail: vt@alice.pnzgu.ru

Vashkevich Nikolay Petrovich
Doctor of engineering sciences, professor, sub-department of computer science, Penza State University

Бикташев Равиль Айнулович
кандидат технических наук, профессор, кафедра вычислительных машин и систем, Пензенская государственная технологическая академия

E-mail: bra559620@sura.ru

Biktashev Ravil Aynulovich
Candidate of engineering sciences, professor, sub-department of computer science, Penza State Technological Academy

---

УДК 681.3.012

Волчихин, В. И.

Планировщик задач с аппаратной поддержкой для многопроцессорных систем / В. И. Волчихин, Н. П. Вашкевич, Р. А. Бикташев // Известия высших учебных заведений. Поволжский регион. Технические науки. – 2012. – № 1 (21). – С. 12–20.