МНОГОЯДЕРНАЯ АРХИТЕКТУРА
ПРОБЛЕМНЫЕ АСПЕКТЫ

В.Слюсар
swadim@inbox.ru

Увеличение числа параллельно работающих ядер в одной микросхеме открывает перед разработчиками процессорных систем новые возможности. Вместе с тем, многоядерная архитектура обусловливает и ряд проблем. Речь идет о развитии не только операционных систем, но и аппаратной поддержке интерфейсов ввода-вывода. Статья рассказывает об основных трудностях и направлениях работы на пути эффективного внедрения архитектур с параллельными вычислениями.

ФОРМЫ МНОГОПРОЦЕССОРНОЙ ОБРАБОТКИ. СИММЕТРИЧНАЯ ОБРАБОТКА

Существует несколько методов повышения быстродействия вычислительных средств с многоядерной архитектурой [1, 2]. К ним относятся:

- симметричная многопроцессорная обработка (symmetric multiprocessing, SMP);
- асимметричная многопроцессорная обработка (asymmetric multiprocessing, ASMP);
- многопроцессорная обработка с назначением приложению конкретного ядра (Bound Multiprocessing, BMP);
- технология виртуализации (virtualisation technology, VT).

При SMP единственная операционная система (ОС) динамически распределяет задачи между доступными процессорными ядрами, оптимизируя использование ресурсов. Ядра задействуются равномерно, и прикладные программы могут выполняться параллельно на всем множестве ядер (рис.1). При этом достигается максимальное быстродействие системы. Важно, что для синхронизации приложений вместо сложных механизмов и протоколов межпроцессорной коммуникации применяют стандартные функции ОС. Таким образом, проще реализовать проекты с распараллеливанием программных потоков. Общая для совокупности ядер ОС позволяет с помощью служебных инструментов собирать статистику, единую для всей архитектуры. Соответственно, можно облегчить отладку и оптимизацию приложений на этапе разработки или масштабирования для других форм многопроцессорной обработки.

Сегодня SMP широко применяют в многопроцессорных суперкомпьютерах и серверных приложениях. Однако если необходимо детерминированное исполнение программ в реальном времени, например при визуализации мультимедийных данных, возможности сугубо симметричной обработки весьма ограничены. Может возникнуть ситуация, когда приложения, выполняемые на различных ядрах, обращаются к одному ресурсу ОС. В этом случае доступ получит только одно из ядер. Остальные будут простаивать до высвобождения критической области. Естественно, резко снижается произ-

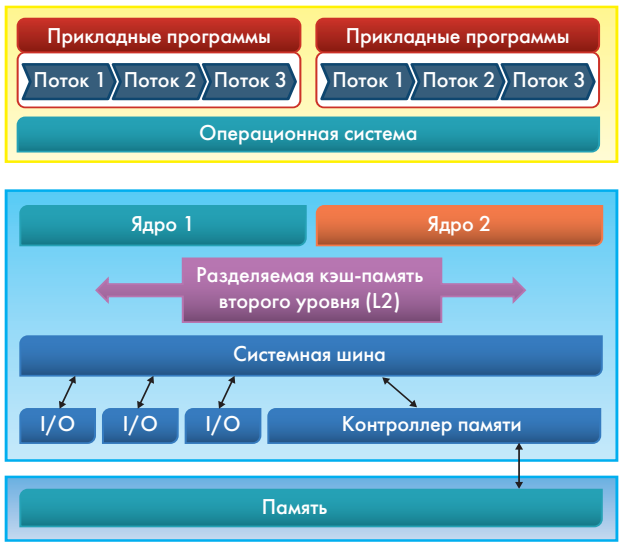

Рис.1. Принцип реализации симметричной многопроцессорной обработки (SMP) на примере двухъядерного процессора

водительность приложений реального времени. Такой опасности подвержены в первую очередь архитектуры с SMP, реализованные на ОС с монолитным ядром, например Windows CE. Основная часть их функциональных возможностей сосредоточена в единственном исполняемом файле. Из-за его большого размера системные запросы требуют длительного времени выполнения. По этой причине многие поставщики ОС занялись внедрением в свои разработки микроядерной архитектуры [3].

В отличие от концепции монолитной ОС, функциональные возможности микроядерной архитектуры рассредоточены по большему числу файлов. Основной исполнительный модуль содержит ядро с минимальным набором управляющих функций и может передавать управление процессами другим модулям. Вся обработка данных, функции ввода-вывода и т.д. производятся во внешних по отношению к микроядру процессах [3]. Сокращение размеров ядра ведет к уменьшению длины критических потоков команд. Следовательно, минимизируется время простоя процессора. Обычно микроядро контролирует доступ к аппаратному уровню; синхронизирует параллельные процессы; резервирует и освобождает память; создает, закрывает и планирует потоки исполняемого кода, поддерживая семафоры для их синхронизации; иногда контролирует и операции ввода-вывода. В идеале микроядро следует реализовать в виде менеджера запросов, который распределяет по внешним процессам заявки на доступ к ресурсам.

АСИММЕТРИЧНАЯ ФОРМА ОБРАБОТКИ

Кардинальным решением проблемы борьбы за ресурсы ОС является технология асимметричной многопроцессорной обработки (рис.2). Вместо единой (как при SMP) ОС на каждом ядре функционирует своя, отдельно установленная, системная оболочка. Поэтому ASMP пригодна для приложений, которые нельзя эффективно выполнить в рамках единой ОС. В простейшем случае оболочки являются экземплярами одной ОС. В более сложном варианте на каждое ядро можно установить другую систему (рис.2), например реального времени. В классической системе с ASMP каждое приложение выполняется полностью на одном ядре, даже если другие свободны. По этой причине отдельные ядра могут функционировать в крайне напряженном режиме или, наоборот, иметь недостаточную нагрузку. Как бы то ни было, главное преимущество такого подхода – надежное и независимое выполнение каждого приложения. В качестве примера рассмотрим радиолокационную станцию с многоядерным процессором для цифровой обработки сигналов. На одном из ядер запускается операционная система реального времени (ОСРВ) для ввода первичных радиолокационных данных. Другое ядро отвечает за передачу данных и визуализацию параметров обнаруженных целей. Такое распределение задач не всегда оптимально с точки зрения быстродействия. Однако критическим

Рис.2. Принцип реализации асимметричного мультипроцессинга (ASMP)

фактором является надежность и эффективность выполнения асимметричных процедур. При этом обработка данных объединяется по принципу "2 в 1".

Отметим, что ASMP – единственная технология, в которой различные ОС можно использовать параллельно на физическом уровне. Поэтому она лучше всего подходит для архитектур с жестким разделением аппаратных ресурсов ядер [4]. Распределение аппаратных ресурсов обычно происходит при запуске ЭВМ и является статическим, неизменным во времени. На физическом уровне этот процесс затрагивает также средства хранения данных, обработку прерываний и др.

Недостаток асимметричной обработки – сложность взаимодействия между процессами, исполняемыми на разных ядрах. Эффективность работы зависит от применения сложных средств межпроцессорной коммуникации. Одно из направлений дальнейшего развития – возможность жесткого закрепления конкретных ядер за приложениями. Например, нумерация ядер подобно регистрам классического одноядерного процессора. Это сократит время динамического перераспределения периферийных ресурсов. Кроме того, можно "привязать" встроенные в процессор контроллеры ввода-вывода к соответствующим аппаратным модулям.

ПЕРСПЕКТИВА: ОБЪЕДИНЕНИЕ МЕТОДОВ ОБРАБОТКИ

Как ASMP, так и SMP обладают рядом недостатков. Эти недостатки отчасти исправлены в разработке компании QNX Software Systems – многопроцессорной обработке с назначением приложению конкретного ядра [2]. BMP – это модификация традиционной симметричной формы обработки. В ней тоже одна ОС контролирует все аппаратные и системные средства, динамически распределяя их между приложениями. Однако во время инициализации приложения пользователь может указать, чтобы оно выполнялось только на за-

данном ядре, независимо от того, простаивают другие ядра или нет. При этом разработчик берет на себя задачи оптимального распределения ядер между приложениями, контроля переполнения общей для всех ядер кэш-памяти второго уровня и т.д.

С приходом на смену двухъядерным процессорам многоядерных* появится возможность реализовать многозадачность "4 в 1" и "8 в 1". Самое широкое распространение получат процессоры, сочетающие методы ASMP и SMP. При этом скорее всего будут задействованы механизмы произвольного объединения ядер в кластеры и функции межкластерного распараллеливания задач. Так, в восьмиядерном процессоре три ядра могут реализовывать параллельное выполнение приложений с помощью ASMP, а двухядерный и трехядерный кластеры – использовать симметричную обработку. Внедрением подобной технологии занимаются специалисты AMD, разрабатывая архитектуру "агрегированного ядра". По данным [5], в ней несколько физических ядер выступают как одно виртуальное. Пока неясно, в какой мере эта концепция воплотит идею множественной кластеризации ядер. Однако сам факт разработок стимулирует интерес фирм-производителей многоядерных процессоров к подобным решениям.

Другой способ объединить технологии многопроцессорной обработки – переход к ОС с несколькими микроядрами. Каждое из них исполняется на своем ядре процессора, причем микроядра могут быть идентичными или же контролировать непересекающиеся аппаратные ресурсы. При этом только часть ядер или кластеров процессора задействована под микроядра. Остальные обрабатывают внешние процессы единой ОС. Таким образом, данная идея подразумевает ASMP на уровне микроядер, и симметричную обработку – на уровне внешних по отношению к ним программных потоков и драйверов.

ТЕХНОЛОГИЯ ВИРТУАЛИЗАЦИИ

В нынешней "двухядерной реальности" интеграция симметричного и асимметричного методов достигается иначе – например, с помощью технологии виртуализации (virtualization technology, VT). Она была разработана еще в конце 60-х годов прошлого века. Суть VT – в рамках одной ОС функционирует множество виртуальных машин, причем на каждой машине действует своя операционная среда (рис. 3). Она может быть экземпляром основной ОС – так называемым виртуальным разделом (partition). Запуск виртуальных ОС, а также обработку их запросов производит монитор виртуальных машин (VMM)**. Эта программа работает непосредственно с аппаратным обеспечением, обрабатывает запросы виртуальных ОС и эмулирует ответы от реального аппаратного обеспечения.

Виртуальные разделы ОС изолированы друг от друга – как в отношении используемого процессорного времени, так и оперативной памяти. Поэтому процессы, конфликтующие в рамках единой ОС, но разнесенные по разным виртуальным машинам, работают стабильно. Можно считать, что на уровне ядер программные процессы выполняются по технологии SMP. А с точки зрения пользователя реализуется принцип асимметричной обработки на ОС виртуальных машин. VT освобождает разработчиков приложений от необходимости регулирования межпроцессорных взаимодействий и упрощает распределение аппаратных ресурсов.

Благодаря технологии виртуализации в одной системе могут функционировать приложения, ранее требовавшие отдельного ПК: брандмауэры, серверы баз данных и т.п. Высокая степень абстракции и замкнутости виртуальных оболочек процессов облегчает переносимость приложений, например с одного виртуального сервера на другой. В процессе работы совокупности виртуальных машин можно изменить виртуальное разделение ресурсов с помощью VMM (рис.3). Это открывает новые возможности для удаленного администрирования и реконфигурации архитектуры системы.

Новые технологии Intel способствуют становлению аппаратной поддержки VT в качестве стандарта для всех платформ и ОС.

VT можно применять и на отдельном ядре в рамках ASMP. Так, при наличии достаточного числа ядер, каждой или некоторым виртуальным машинам может назначаться для выполнения свое процессорное ядро. При этом VMM – общий для всех ядер, и первичной является технология виртуализации. Данный метод аналогичен технологии BMP, однако за конкретным ядром закрепляется не приложение, а целая виртуальная машина со своей ОС.

Рис.3. Технология виртуализации (VT)

* Образцы четырехъядерных процессоров Core 2 Quad от Intel доступны для тестирования с сентября 2006 года.

** VMM – Virtual Machine Monitor. Также используется термин "супервизор" – Hypervisor

Другой подход – функционирование отдельного VMM и нескольких виртуальных машин на каждом ядре или кластере. Такие методы обработки успешно применяются на основе ОСРВ в сложных системах управления и приложениях реального времени. Следует отметить, что в силу недостаточного быстродействия известных систем, например Windows XP Embedded или Linux, на них не следует запускать ОСРВ в режиме виртуальной машины. Основной системой должна стать другая ОСРВ с функциями поддержки виртуальных разделов. К сожалению, выбор таких систем на рынке ПО весьма ограничен.

АРХИТЕКТУРЫ ОС С ИЗОЛИРОВАННЫМИ РАЗДЕЛАМИ

Стандарт надежности ОС – это соответствие спецификациям ARINC (Aeronautical Radio, Incorporated), которые гарантируют совместимость и взаимозаменяемость ОС для авиационной отрасли. Документ ARINC 653: Avionics Application Standard Software Interface определяет архитектуру виртуальных разделов ОС, обеспечивающую безопасность и стабильность работы [8]. В нем введено понятие программного интерфейса APEX (Application/Executive) между ОС и приложениями, исполняемыми в виде отдельного виртуального раздела.

Такой подход применяется в ОСРВ LynxOS-178 фирмы LynuxWorks (http://www.lynuxworks.com) [7]. По сравнению с VT, в LynxOS-178 дополнительно предусмотрено изолированное исполнение каждого приложения (рис.4). При этом виртуальные разделы (partitions) представляют собой совокупность приложения и операционной системы раздела (POS – Partition Operating System). Формирование исполняемого программного кода в каждом разделе производится в обычном режиме (User Mode). Ядро LynxOS-178 – модульная ОС (MOS, Module Operating System), компоненты которой совместно с загрузчиком (Board Support Package, BSP) функционируют в системном режиме (Supervisor). Драйверы устройств также исполняются в режиме виртуальной машины, что повышает надежность функционирования ядра ОС.

Асинхронность прерываний – одна из главных проблем любой ОСРВ. Особую важность она приобретает в случае изолированных виртуальных разделов. Необходимо, чтобы прерывания в одном из них не влияли на время выполнения процессов и обращения к памяти в остальных. Одним из источников прерываний являются таймеры. В LynxOS-178 они управляют диспетчеризацией событий как внутри POS, так и в самой MOS. Чтобы избежать взаимного влияния прерываний разных разделов, в пользовательском режиме прямой доступ к таймерам исключен. Они обрабатываются только в режиме супервизора. Поэтому раздел получает все относящиеся к нему прерывания только при захвате процессорного времени (раздел активен). В противном случае все временные события откладываются и сохраняются для передачи при активизации раздела.

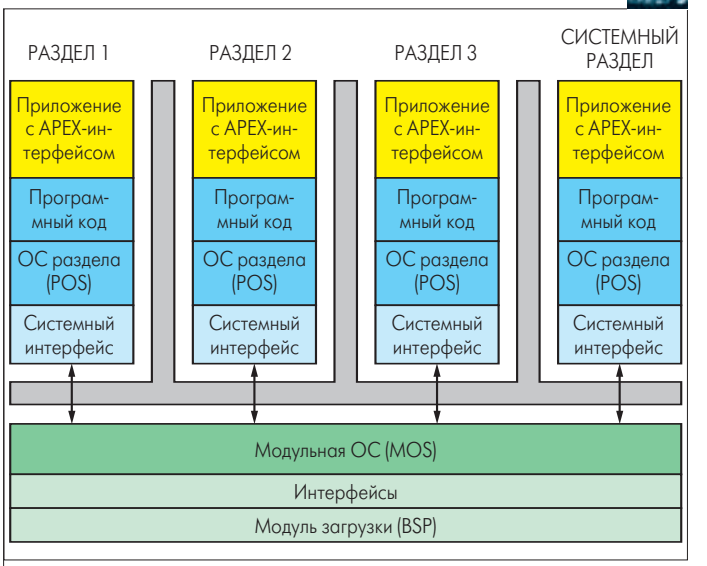

Рис.4. Архитектура ОСРВ с изолированным исполнением приложений

Важным источником прерываний могут быть устройства ввода-вывода. Согласно требованиям ARINC 653, для синхронной передачи данных следует применять метод регулярного опроса устройств. Если операции ввода-вывода требуют обработки прерываний, то соответствующий поток событий обрабатывается на уровне MOS, а затем переадресовывается POS активного раздела. Если при этом виртуальный раздел занимается, например, считыванием непрерывного потока данных, задержка на время обработки прерывания приве-

дет к потере информации. Кроме того, чем дольше прерывание обрабатывается MOS, тем дольше раздел ожидает своей очереди.

Существуют и другие подходы к реализации изолированных разделов. Даже в ОСРВ, удовлетворяющих формальным требованиям ARINC 653, применяют разнообразные механизмы виртуализации. Например, в некоторых программных продуктах [7] встроена не только одноуровневая диспетчеризация, т.е. управление виртуальными разделами, но и, в отличие от LynxOS-178, двухуровневая – с изоляцией как разделов, так и процессов внутри них. Разнообразие методов обусловлено интенсивным развитием программных средств для многоядерных процессоров. Но, несмотря на активность разработчиков, до оптимального управления подобными ресурсами еще далеко. Необходима поддержка ОСРВ виртуальных машин, сгенерированных на других программных платформах – системах типа Windows XP Embedded, Linux и др. Важно, чтобы следующие поколения ОСРВ поддерживали VT различных разработчиков с возможностью переноса виртуальных разделов разных ОС (например, разделов ОС QNX в ОСРВ LynxOS-178 и наоборот). Иначе строить гибкие многоядерные системы будет довольно сложно.

Разработчик приложений для многоядерных систем должен позаботиться о том, чтобы сократить длину потоков команд при параллельном программировании. Уменьшение размеров параллельно исполняемых задач снижает время их выполнения, следовательно, быстрее освобождаются ресурсы системы. Нужно обратить внимание на процессы, которые могут выполняться независимо, но в ходе работы требуют результатов других задач. Ведь приложения, которые находятся в состоянии ожидания, могут взаимно заблокировать свои процессы, что приведет к системному сбою. Поэтому для надежного выполнения распределенных вычислений обращение к таким взаимозависимым задачам следует четко регламентировать [8].

АППАРАТНЫЕ НУЖДЫ МНОГОЯДЕРНЫХ АРХИТЕКТУР

Следующая проблема – реализация встроенных интерфейсов процессорных модулей. На рынке уже появились первые двухъядерные модули в стандарте Compact PCI. Однако в полной мере потенциал многоядерных архитектур раскроется с переходом на последовательно-параллельные шины: спецификации Compact PCI Express, ATCA* и др. Благодаря большой пропускной способности интерфейсов и использованию нескольких параллельных каналов система сможет быстрее обрабатывать порты ввода-вывода. Другая цель перехода – управление раздельным доступом ядер к каналам ввода-вывода с помощью динамического или детерминированного закрепления за каждым ядром фиксированного набора портов.

* Advanced Telecommunications Computing Architecture, ATCA. Тип компьютерных систем с последовательно-параллельным интерфейсом, базируется на многоканальном применении интерфейсов PCI Express [10].

Уровнять шансы многоядерных процессоров с Compact PCI и Compact PCI Express можно усовершенствованием спецификаций PICMG 2.16 и 2.20. Суть необходимой доработки – предоставление каждому из ядер независимого доступа к интерфейсам ввода-вывода. В идеале, часть интерфейса следует реализовать непосредственно на ядре. Однако это требует радикальных изменений существующей архитектуры. Другой способ – усовершенствование внутрипроцессорной электроники с добавлением функций разбора приходящих на интерфейс потоков и их коммутации на нужное ядро. Однако нужно помнить, что введение дополнительной маршрутизации приводит к задержкам обработки данных.

Кроме разработки новых чипсетов (с шлюзованием встроенных в микропроцессор каналов PCI Express на линии Ethernet) необходимо расширить номенклатуру адаптеров ввода-вывода в самих процессорах. Так, в сигнальные процессоры фирмы Texas Instruments TMS320C6455, помимо адаптеров Ethernet, встроены каналы высокоскоростного интерфейса RapidIO (до 12 Гигабит/с). Вообще, у современных многоядерных процессоров (Intel, AMD, Texas Instruments и др.) общие недостатки: узкий набор встроенных контроллеров, отсутствие трансиверов RocketIO, новейших интерфейсов с LVDS (Low Voltage Difference Signal) и т.д.

Примером удачной технической политики является архитектура ПЛИС Virtex 5 фирмы XILINX. В ее библиотеке стандартных элементов содержатся контроллеры практически всех известных интерфейсов ввода-вывода. При этом диапазон поддерживаемых интерфейсов постоянно обновляется и совершенствуется – как аппаратно, так и за счет модулей программной реконфигурации.

Применение технологии ПЛИС для исполнения встроенных в процессор адаптеров интерфейсов представляется весьма перспективным. При проектировании системы разработчик мог бы произвольным образом конфигурировать узлы ввода-вывода ядер процессора, загрузив соответствующие модули в области кристаллов, где реализована технология ПЛИС. Потенциально такие гибридные процессоры обладают беспрецедентной гибкостью по настройке интерфейсов. У них есть возможность напрямую, в обход коммутирующих чипсетов, стыковаться через шинные буферы с разъемами устройств ввода-вывода. Основание для такой интеграции – общность методов приемопередачи сигналов для многих интерфейсов. В частности, принцип LVDS используется в PCI Express, RapidIO, трансиверах RocketIO и др. Первым шагом может стать размещение кристаллов ПЛИС и многоядерного чипа в одном корпусе – подобно четырехъядерному процессору Intel, состоящему из пары двухъядерных кристаллов.

Наибольшую пользу усовершенствование I/O-интерфейсов принесет производителям многоядерных процессоров для встраиваемых систем. Например, для реализации алгоритмов цифрового формирования лучей в базовых станциях

сотовой связи, радиолокации, спутниковой навигации и др. Эти приложения требуют постоянной однозначной привязки устройств ввода данных к выходам антенных элементов решетки [9]. Для таких задач каждому ядру нужны свои, независимые каналы ввода-вывода, например на базе нескольких линий PCI Express. Тогда многоядерный процессор сможет производить ввод-вывод данных параллельно с обслуживанием модулей цифровой обработки сигналов. Кроме того, не нужно будет реконфигурировать систему при включении, поскольку конкретным устройствам ввода-вывода назначены определенные ядра процессора.

Теоретически компьютер с двумя или более ядрами обладает большей производительностью, чем одноядерный процессор. Конечно, при правильном распределении программных и аппаратных ресурсов системы. Уже первые испытания (например, модулей фирмы Kontron) показали, что производительность новых двухъядерных процессоров близка к теоретическим границам. Если сравнить модули на двухъядерном процессоре Intel Core Duo (2,16 ГГц) и процессоре Intel Pentium M 756 (2,1 ГГц), то для идентичных частот и стандартных ПК-приложений производительность по операциям с плавающей запятой двухъядерного процессора выше на 96,5%, а по операциям целочисленной арифметики – на 89,3% [11]. Что же касается ускорения операций ввода-вывода, то над этой проблемой еще предстоит потрудиться всем разработчикам. Роста скорости можно добиться не только за счет нескольких ядер, но и путем увеличения параллельных каналов передачи, возможного исключения коммутирующих чипсетов и маршрутизаторов потоков данных из интерфейсных цепей.

ЛИТЕРАТУРА

1. **Hauser N.**, **Hahner I.** and **Ahne P.** Which kind of multi-core processing for which application? – ECE, October 2006, pp. 27–29. http://www. eceoct06p27.pdf.
2. **Николаев А.** Поддержка многоядерных процессоров во встраиваемых системах. –Открытые системы, 2006, № 07. – http://www.osp.ru/text/302/3290779/.
3. US Patent No. 7103631. Int. Cl.: G06F 15/16; G06F 9/44; G06F 9/46. Symmetric multi-processor system. Peter H. van der Veen. – QNX Software Systems (Ottawa, CA). – Filing Date: 1999-08-25. – Publication Date: 2006-09-05.
4. US Patent No. 7124224. Int. Cl.:G06F 12/00, G06F 11/00, G06F 12/14. Method and apparatus for shared resource management in a multiprocessing system. Steven Tu; Hang Nguyen. – Intel Corporation (Santa Clara (CA), US). – Filing Date: 2000-12-22. - Publication Date: 2006-10-17.
5. **Пантюхин В.** Микропроцессорное многопоточие. – Открытые системы, 2006, №6. – http://www.osp.ru/text/302/2700454/.
6. ARINC and POSIX for Safety-Critical Applications. PDF brochure. – http://www.lynuxworks.com/rtos/0620-00-los178_arinc653_hb_alt.pdf.
7. **Золотарев С.В.** Современные требования к операционным системам реального времени для авиации. – Доклад на семинаре "Решения Kontron для встраиваемых систем высокой надежности и жестких условий эксплуатации. Практический опыт внедрения в ВПК и энергетике Украины и России". 28 февраля 2006 г. – Киев: Институт поддержки эксплуатации АЭС. – http://www.ivl.kiev.ua/articles/0003/.
8. **Таненбаум Э.**, **Хердер Дж.**, **Бос Х.** Надежные и защищенные операционные системы. – Открытые системы, 2006, №6. – http://www.osp.ru/text/302/2700569/.
9. **Слюсар В.** Схемотехника цифровых антенных решеток. Грани возможного. – ЭЛЕКТРОНИКА: НТБ, 2004, № 8. – http://www.electronics.ru/pdf/8_2004/07.pdf.
10. **Слюсар В.** Новые стандарты промышленных компьютерных систем. – ЭЛЕКТРОНИКА: НТБ, 2005, № 6.
11. Doppelte Power. Dual-Core schafft neue Perspektiven. – Elektronik JOURNAL, 2006, № 7. – http://emea.kontron.com/downloads/pressreleases/0706_5658.pdf