МНОГОЯДЕРНОСТЬ КАК ВЫЗОВ ВРЕМЕНИ

Смена концепции развития микропроцессорных систем, связанная с переходом от наращивания их частоты тактирования к увеличению количества параллельно работающих ядер в одном чипе, сулит новые возможности разработчикам встраиваемых систем

Слюсар В.И., д. т. н., профессор, зам. начальника научно-исследовательского управления ЦНИИ вооружения и военной техники ВС Украины, swadim@voliacable.com

Появление первых индустриальных процессорных модулей с двухъядерными решениями позволило отработать четыре основных технологии повышения быстродействия вычислительных средств [1], без использования хотя бы одной из которых эффективность многоядерной архитектуры будет сведена на нет. Сюда относятся:

- симметричная мультиобработка – symmetric multiprocessing (SMP);
- асимметричная мультиобработка – asymmetric multiprocessing (ASMP);
- технология виртуализации – virtualisation technology (VT);
- произвольная комбинация перечисленных вариантов.

Симметричная мультиобработка (SMP)

SMP – наиболее простая из выше перечисленных технологий. В случае ее применения единственная установленная операционная система равномерно использует все ядра, из-за чего запускаемые приложения могут выполняться параллельно на всем их множестве. Эта версия мультипроцессинга облегчает подлинную мультипоточность и мультизадачное управление. Наличие общей для всех ядер операционной системы позволяет централизованно распределять ресурсы между запущенными приложениями, гарантируя лучшее использование для них имеющихся аппаратных средств ЭВМ.

Служебный инструментарий операционной системы может скоординированно собирать статистику прикладных взаимодействий для всего мультиядерного чипа, облегчая таким образом отладку и оптимизацию встраиваемых приложений на этапе их разработки или модификации, в том числе под более сложные технологии мультипроцессинга.

В общем случае данную технологию можно рекомендовать как начальную при создании новых проектов, использующих распараллеливание программных потоков. Задача синхронизации приложений в рассматриваемом случае решается довольно просто, опираясь на стандартные функции операционной системы вместо использования сложных механизмов межпроцессорной связи.

Принцип SMP может быть востребован в многочисленных встраиваемых приложениях. Однако, в случаях, где требуется детерминированное исполнение программ в реальном масштабе времени, в сочетании с комплексом средств визуализации мультимедийных данных, возможности чисто симметричной мультиобработки в двухядерном процессоре становятся весьма ограниченными.

Причина – банальная борьба за общие ресурсы, спрос на которые может превысить возможности по предоставлению доступа, что повлечет за собой простой системы в реальном времени. Для снижения остроты данной проблемы компанией QNX Software Systems была предложена модификация SMP, получившая наименование исключительной многопроцессорности (Bound Multiprocessing, BMP) [2]. Как и в случае традиционной симметричной технологии, в этом режиме одна операционная среда полностью контролирует все системные ресурсы, динамически распределяя их между приложениями.

Однако, при этом во время инициализации приложений пользователь может указать, чтобы одно или несколько из них выполнялось только на заданном ядре, независимо от того, простаивают ли другие ядра. При этом разработчик берет на себя задачу распределения ядер между приложениями, исключая конфликты переполнения общей для процессора кэш-памяти.

Асимметричная мультиобработка

Данная технология является кардинальным решением проблемы борьбы за общие ресурсы. Следует отметить, что асимметричный мультипроцессинг (ASMP) полезен для приложений, которые не могут эффективно выполняться в рамках единой операционной системы.

В простейшем случае на каждом ядре процессора при ASMP функционирует отдельно установленная системная оболочка той же версии, что и на остальных ядрах. Однако, в более сложном варианте для каждого или части ядер используется своя уникальная операционная система, в том числе реального и «нереального» времени.

Существенно, что асимметричная процедура – это единственная технология, в которой различные операционные системы могут быть использованы параллельно на физическом уровне. В случае асимметричной обработки мультиядра, по сути, представляют собой жестко разделенные ресурсы аппаратных средств ЭВМ.

При этом распределение имеющихся аппаратных ресурсов обычно обеспечивается в процессе запуска ЭВМ и является статическим, неизменным во времени. На физическом уровне этот процесс затрагивает также средства хранения данных, использование периферии и обработку прерываний. В дополнение к сказанному, система динамически назначает ресурсы межпроцессорной коммуникации, что делает координацию между ядрами сложным процессом.

Такой менеджмент ресурсов целесообразно в дальнейшем расширить в направлении предоставления разработчику возможности жесткого закрепления конкретных ядер за интересующими его приложениями, используя нумерацию ядер подобно регистрам классического одноядерного процессора. Это позволит избежать временных затрат на динамичное перераспределение периферийных ресурсов, предоставит возможность пользователю жестко закреплять встроенные в процессор котроллеры ввода-вывода за соответствующими аппаратными модулями.

Следует отметить, что в асимметричной системе мультиобработки любая задача всегда обрабатывается внутри одного и того же ядра от начала до ее завершения, даже если остальные ядра в этот момент времени свободны. Хотя отдельные ядра могут использоваться в крайне напряженном режиме или иметь недостаточную нагрузку, главное преимущество такого подхода состоит в том, что каждое из приложений функционирует надежно и независимо.

Применительно к двух- и четырехъядерной конфигурациям процессоров эта форма мультиобработки гарантирует устойчивое функционирование аппаратных средств ЭВМ в тех индустриальных применениях, где прежде не представлялось возможным осуществлять многозадачность в одной системе.

В качестве примера можно указать радиолокационную станцию с цифровой обработкой сигналов. В данном случае на одном из ядер процессора обеспечивается запуск операционной системы реального времени для реализации ввода первичных радиолокационных данных, а на другом ядре (ядрах) производится обслуживание последующих процессов визуализации траекторных параметров сопровождения трасс обнаруженных целей и передачи данных потребителям.

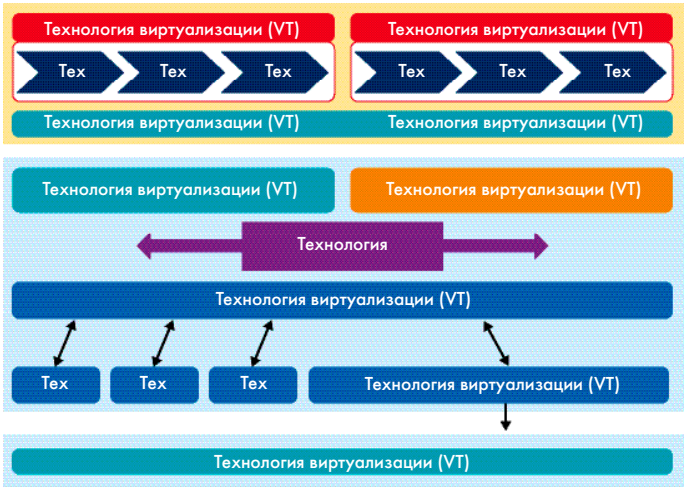
Хотя такое распределение задач зачастую может не быть оптимальным по быстродействию, это неважно, ибо критическим фактором являются, в данном случае, надежность и эффективность выполнения асимметричных процедур, комбинирующих обработку данных по принципу «2 в 1».

С приходом на смену двухъядерным процессорам мультиядерных решений станет возможной реализация многозадачности по принципам «4 в 1» или «8 в 1». И споры, являются ли ядра полностью используемыми, станут излишними. С увеличением количества ядер до 4 и более предпочтительной будет реализация в одном процессоре комбинации асимметричных и симметричных процедур мультипроцессинга, что обеспечит эффективную по быстродействию альтернативу доступным сегодня для дуальных ядер версиям системного менеджмента. Но неизбежно будет востребована возможность реализации механизма произвольного объединения ядер в кластеры и предоставления функций межкластерного распараллеливания задач. К примеру, в 8-ядерном процессоре можно реализовать симметричные мультипроцессинги на двухъядерном и трехъядерном кластерах, а оставшиеся три ядра использовать для ассимметричного распараллеливания приложений.

Технология виртуализации (VT)

В нынешней «двухъядерной реальности» интеграция симметричного и асимметричного мультипроцессинга может осуществляться в рамках VT. Суть ее сводится к запуску и функционированию в рамках одной операционной системы множества виртуальных машин, на каждой из которых может быть реализована своя операционная среда.

Виртуальные разделы (partition) должны жестко изолироваться друг от



Принцип реализации симметричного мультипроцессинга (SMP) на примере двухъядерного процессора

друга с точки зрения используемого процессорного времени и оперативной памяти. При этом на уровне ядер программные процессы выполняются по технологии симметричного распределения задач и ресурсов, а на уровне пользователя реализуется принцип виртуальной асимметричности.

Такое решение упрощает управление аппаратными средствами ЭВМ, освобождая прикладных разработчиков от необходимости обработки межпроцессорных коммуникаций. Учитывая тот факт, что в рамках VT процессы, которые могут находиться в противоречии друг с другом на единственных основных системах, оказываются разграниченными и полностью изолированными от остальных, стабильность полных систем улучшается.

Виртуальное разделение ресурсов с помощью VMM может быть адаптивно переназначено, согласуясь с текущими потребностями по ходу выполнения задач всей совокупностью виртуальных машин. Высокая степень абстракции и замкнутости виртуальных оболочек процессов позволяет приложениям «кочевать» от одного сервера к другому, что было невероятно до недавнего времени, и предоставляет новые возможности для удаленного администрирования и реконфигурации системных архитектур.

Примечательно, что новые процессорные технологии Intel способствуют становлению VT в качестве стандарта для всех аппаратных платформ и операционных систем. При этом заметно снижается острота проблемы достижения аппаратной независимости программного обеспечения.

Наконец, виртуализация уже сегодня предлагает новый вариант программной интеграции в единой системе автономных приложений, требовавших прежде аппаратного разнесения на несколько ЭВМ (брандмауэры, серверы данных и т. п.). Это способствует достижению заметной экономии в стоимости оборудования.

Технология виртуализации найдет применение и в рамках асимметричной процедуры на отдельно взятых ядрах. Такой комбинированный метод мультипроцессинга, объединяющий VT и ASMP, при его реализации на основе ОС реального времени позволяет извлечь дополнительный выигрыш от многоядерных решений, который может быть существенным в случае сложных систем управления. По этой причине данный подход используется во многих индустриальных приложениях, работающих в режиме реального времени.

Однако, пока что целесообразность запуска ОС реального времени (ОСРВ) в режиме отдельной виртуальной машины представляется весьма сомнительной из-за недостаточного быстродействия известных операционных систем. Действительно, довольно нелепой выглядела бы попытка установить ОСРВ как виртуальный раздел из-под Windows XP Embedded или Linux.

Указанный режим матрешечной виртуализации ОСРВ станет оправданным только в случае использования монитора виртуальных машин на основе такой же ОСРВ. К сожалению, следует отметить, что на данный момент на рынке программного обеспечения выбор полноценных ОС реального времени, реализующих функции поддержки виртуальных машин, весьма ограничен.

Архитектурный стандарт

Наиболее полно критериям и требованиям, предъявляемым к встраиваемым системам, отвечает архитектура ОСРВ LynxOS-178 фирмы LynuxWorks (*www.lynuxworks.com*). Главное ее преимущество – соответствие требованиям стандарта ARINC 653 («Avionics ApplicationSoftware Standard Interface», *www.arinc.com*) разработанного компанией ARINC (Aeronautical Radio, Inc.) в 1997 году для регламентации универсального программного интерфейса APEX (Application/Executive) между операционной системой бортового авиационного компьютера и прикладным программным обеспечением [3].

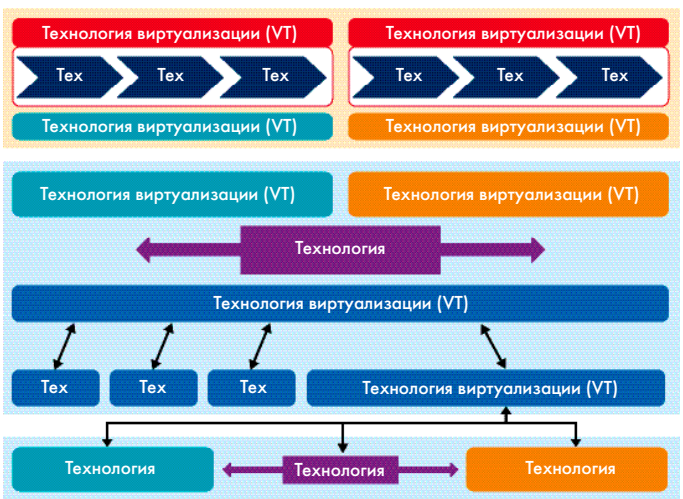
Принятая в 2003 году действующая редакция ARINC 653 предполагает в качестве неотъемлемого атрибута для истинной ОСРВ наличие архитектуры изолированных (partitioning) виртуальных машин [3]. Однако, по сравнению с описанной выше технологией виртуализации, в ОСРВ LynxOS-178 дополнительно предусматривается, что каждое из приложений, а не вся операционная среда с массой программ, должно выполняться независимо.

При этом виртуальные разделы (partition) представляют собой совокупность некоторого приложения и так называемой POS (Partition Operating System) – операционной системы раздела. Рассмотрим особенности реализации технологии виртуализации на базе ОСРВ LynxOS-178 с несколькими изолированными разделами, каждый из которых является самостоятельным приложением. Все данные и программный код в отдельном разделе линкуются вместе и выполняются в пользовательском режиме.

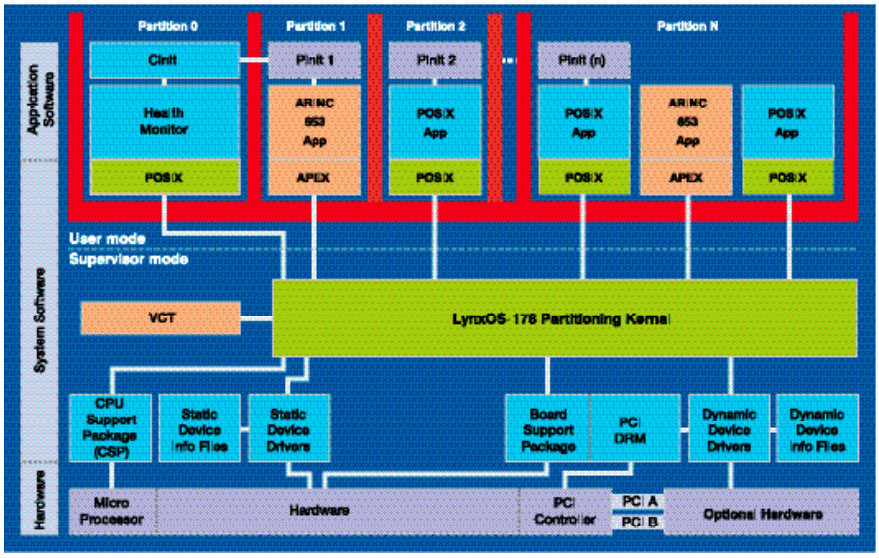
Ядро (Kernel) LynxOS-178 представляет собой модульную операционную систему (MOS, Module Operating System), компоненты которой совместно с BSP (Board Support Package) функционируют в супервизорном (Supervisor) режиме. Дополнительно может назначаться один специальный раздел с некоторыми специфичными возможностями, реализующими функции ввода-вывода, коммутации режимов и т. п.

Стандарт ARINC 653 задает требования к интерфейсу для обмена информацией между виртуальными разделами. Этот шлюзовой интерфейс должен гарантировать изоляцию всех задействованных в разделах программно-аппаратных элементов: отведенных сегментов оперативной памяти, времени использования процессора, исполняемых программных кодов, векторов прерываний.

С этой целью каждому виртуальному разделу отводится непрерывное физическое адресное пространство ОЗУ, границы которого не могут меняться в процессе работы приложений. Выделение и освобождение памяти из жестко отведенного адресного пространства осуществляется MOS. ОСРВ должна гарантировать полную изоляцию всей информации в адресном пространстве внутри назначенного каждому разделу сегмента. Это требование распространяется на исполняемый программный код, стеки, массивы констант, статические и динамические массивы данных.



Принцип реализации асимметричного мультипроцессинга (ASMP)



Реализация архитектуры системы на базе ОСРВ LynxOX-178 с несколькими изолированными разделами, каждый из которых – самостоятельное приложение

Ключевой основой ARINC 653 является предоставление возможности подконтрольного влияния процессов в одном виртуальном разделе на выполнение приложений в другом по заранее установленным правилам (например, путем установки флагов или условий). Процессорное время при этом должно выделяться каждому разделу строго циклически, а интервал времени захвата ядер процессора для каждого раздела регламентируется в специальной конфигурационной таблице.

Взаимодействие между приложениями разных разделов может осуществляться на основе обмена сообщениями по командам передачи/приема. При этом формируется канал связи, который должен быть заранее «прорисован» в конфигурационной таблице системы. Возможен и вариант обмена данными через буфер. В этом случае для каждого виртуального раздела назначается свой буфер под запись данных, из которого приложения всех остальных разделов способны лишь считывать информацию. Такой подход обеспечивает возможность массового обмена данными между изолированными разделами.

Внутри виртуальных разделов LynxOS-178 процессы (нити) конкурируют между собой за использование процессорного времени на основе механизма диспетчеризации с вытеснением по приоритетам. При необходимости в режиме Supervisor из MOS доступна функция назначения отдельным областям памяти атрибутов «только для выполнения». Это исключит разрушение области кода приложения в пользовательском режиме.

Асинхронизм прерываний, как известно, вляется главной проблемой любой ОСРВ. Особого внимания эти события заслуживают в свете необходимости изоляции виртуальных разделов. Важно, чтобы прерывания в одном из них не влияли на время выполнения процессов и обращение к памяти в остальных.

Наиболее распространенные генераторы прерываний – таймеры. В LynxOS-178 они управляют диспетчеризацией событий как внутри POS, так и в самой MOS. Чтобы исключить взаимное влияние прерываний между разделами, реализуется ряд требований. Прежде всего, прерывания таймера могут возникать только в супервизорном режиме, тогда как в пользовательском прямой доступ к таймерам невозможен. При активности какого-либо раздела ему должны переадресовываться все касающиеся этого раздела внешние временные события. В течение интервала пассивности раздела все относящиеся к нему прерывания откладываются и сохраняются для последующей передачи при очередной активизации.

Важнейшим источником прерываний могут быть устройства ввода-вывода. По требованиям ARINC 653, для реализации синхронной передачи данных следует применять метод регулярного опроса устройств. Если же операции ввода-вывода требуют обработки прерываний, то соответствующий поток прерывающих событий должен обрабатываться на уровне MOS и лишь потом переадресовываться в POS активного раздела.

К сожалению, при такой технологии ввода-вывода могут возникать временные задержки на интервал цикличности активности виртуальных разделов. Чтобы избежать при этом потерь информации разработчикам следует заранее принимать меры по недопущению подобных коллизий.

Независимо от выбранного метода, разработчик многоядерной системы должен позаботиться о существенном уменьшении размеров задач или длины нитей при выполнении параллельного программирования. Это позволит максимизировать детализацию структуры программного кода и тем самым гарантировать оптимальное использование доступных ресурсов.

Особого внимания с его стороны заслуживают процессы, которые могут выполняться независимо, но чтобы получить правильный результат, требуют информации от других задач. Это касается приложений, которые «ждут» друг друга и поэтому могут взаимно заблокировать свои процессы, вплоть до зависания системы.

Против таких артефактов программного кода не в силах пока бороться ни одна из операционных систем. Поэтому коммуникационные обращения к данному виду взаимозависимых задач должны осуществляться на основе четко регламентированных правил, гарантирующих «целостность» распределенных действий.

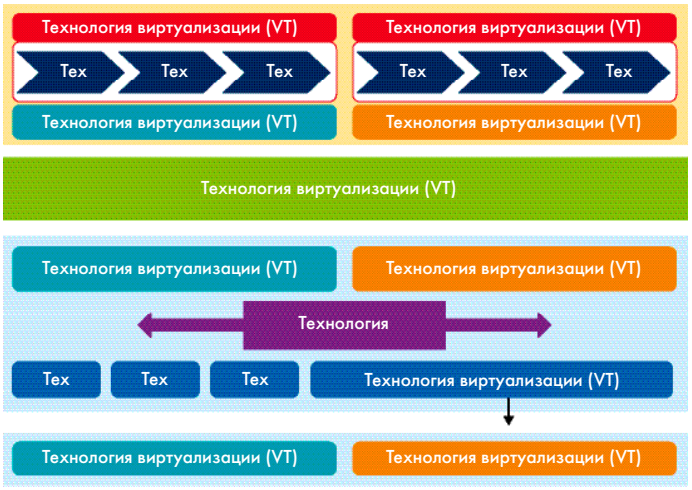
Альтернативные методы

Помимо архитектуры, соответствующей ARINC 653, существуют и другие подходы к реализации изолированных разделов. И внутри семейства ОСРВ, удовлетворяющих формальным признакам ARINC 653, тоже наблюдается отсутствие однообразия механизмов виртуализации.

Например, в отличие от LynxOS-178 некоторые программные продукты, согласно [3], реализуют не только одноуровневую диспетчеризацию, управляя виртуальными разделами, но и двухуровневую, с изоляцией как разделов, так и процессов внутри них. Все это разнообразие является следствием неизбежных издержек интенсивного развития данной отрасли программного обеспечения.

Однако при всех огромных достижениях, которыми программные средства встретили появление многоядерных процессоров, до оптимального управления подобными ресурсами еще далеко. Как показал опыт, поддержка многоядерности и двухъядерности оказались далеко не тождественными понятиями, поскольку многие формально многопоточные программы, эффективно использовать больше двух ядер CPU еще не способны.

Насущной потребностью, по мнению автора, является реализация поддержки в ОСРВ виртуальных машин, сгенерированных в других программных платформах, в том числе, в опе-



Технология виртуализации (VT)

рационных системах типа Windows XP Embedded, Linux и др.

Важно, чтобы дальнейшее развитие ОСРВ на поле многоядерности шло по пути если не интеграции конкурирующих ныне продуктов, то хотя бы создания шлюзов для поддержки на одной ОСРВ виртуальных технологий различных разработчиков с возможностью переноса виртуальных разделов, например, операционной системы QNX в ОСРВ LynxOS-178 и наоборот. Без этого невозможно будет строительство гибких многоядерных систем в конкурентоспособное время.

Другим проблемным вопросом является качество аппаратной реализации процессорных модулей на основе многоядерных технологий. В настоящее время на рынке встраиваемых систем появились первые двухъядерные модули в стандарте Compact PCI.

Однако, очевидно, что максимальная реализация потенциальных возможностей многоядерных архитектур возможна только с переходом на спецификацию Compact PCI Express или к аналогичным интерфейсам, использующим последовательно-параллельные шины. Речь идет об упоминавшейся уже задаче обеспечения управляемого раздельного доступа ядер к канала ввода-вывода.

Уровнять шансы Compact PCI и Compact PCI Express в некотором смысле могло бы усовершенствование спецификаций типа PICMG 2.16. Суть необходимой доработки состоит в предоставлении каждому из ядер независимого шлюза на часть или же все каналы Ethernet.

Однако такая доработка представляется слишком радикальной и нуждается в создании не только специализированных чипсетов, реализующих функции шлюзования встроенных в многоядерный микропроцессор каналов PCI Express на линии Ethernet, но и совершенствовании архитектуры самих процессоров в направлении расширения номенклатуры адаптеров ввода-вывода.

Примером тому могут служить сигнальные процессоры фирмы Texas Instruments TMS320C6455, в которых помимо адаптеров гигабитной Ethernet имеются встроенные каналы высокоскоростного интерфейса RapidIO (до 12 Гигабит/с). Однако недостатком процессоров Intel, AMD, и процессоров Texas Instruments является пока слишком узкий набор встроенных контроллеров, отсутствие трансиверов RocketIO, новейших версий LVDS и др.

В этом смысле примером удачной технической политики является архитектура ПЛИС фирмы XILINX, в топологии которой в большом ассортименте содержатся контроллеры практически всех известных интерфейсов ввода-вывода, и этот перечень постоянно обновляется и совершенствуется.

Возможным вариантом решения проблемы адаптации контроллеров ввода-вывода, заложенных в кристалл многоядерного процессора, под нужды разработчиков встроенных систем явилось бы создание гибрида многоядерного процессора и технологии ПЛИС, по которой бы выполнялись встроенные адаптеры интерфейсов. В этом случае разработчик при проектировании своей системы мог бы произвольным образом сконфигурировать узлы ввода-вывода процессора, загрузив соответствующие конфигурационные модули в ту область кристалла, где реализуется технология ПЛИС.

В итоге, в распоряжении создателей встраиваемых систем появились бы беспрецедентно гибкие по настройке интерфейсных возможностей многоядерные процессоры, которые могли бы напрямую, в обход коммутирующих чипсетов, стыковаться через шинные буферы с разъемами устройств ввода-вывода данных. Данная технология очень важна для повышения быстродействия систем цифрового формирования луча в цифровых антенных решетках [5]. Дело в том, что для таких задач в идеале каждое ядро должно иметь свои независимые каналы ввода-вывода, например, на базе нескольких дифференциальных линий PCI Express. В результате ввод-вывод данных при обслуживании многоядерным процессором модулей цифровой обработки сигналов станет независимым и полностью параллельным во времени. Кроме того, на этой основе удастся исключить необходимость переконфигурации системы при ее включении.

Назначив конкретным устройствам ввода-вывода конкретные ядра процессора, можно не заботиться о том, что после очередного выключения система поведет себя неконтролируемым образом. Это очень важно, например, для встраиваемых систем, реализующих алгоритмы цифрового формирования луча в базовых станциях сотовой связи, радиолокации, спутниковой навигации и др. приложений. В них непременным условием является постоянная однозначная привязка ресурсов ввода данных к выходам конкретных антенных элементов.

Таким образом, теоретически индустриальный компьютер с двумя или больше ядрами будет способен к большей производительности, чем одноядерный процессор, при условии, что распределение программных и аппаратных ресурсов системы будет соответствующим.

Тестирование, к примеру, модулей фирмы Kontron, показало, что двухъядерные процессоры близки к достижению производительности, обещанной в теории. Сравнение быстродействия модуля, содержащего двухъядерный Intel Core Duo (2,16 ГГц), и модуля на основе процессора Intel Pentium M 756 (2,1 ГГц) свидетельствует, что для идентичных частот и обычных офисных приложений производительность по операциям с плавающей запятой возрастает на 96,5%, а по операциям целочисленной арифметики – на 89,3%.

Что же касается ускорения операций ввода-вывода, то над этой проблемой, в свете указанных выше идей, еще предстоит потрудиться всем разработчикам, начиная с той же Intel. В результате, по всей видимости, соответствующий прирост скорости будет более существенным, сообразно не только количеству ядер, но и с учетом увеличения параллельных каналов передачи, а также исключения из линий обмена данными коммутирующих чипсетов и маршрутизаторов потоков.

Литература

1, 2, 3, 4, 5 – документы см. на прилагаемом диске.