чение, но все больше и больше встраиваемых систем разрабатываются на *FPGA* платформе.*FPGA* может быть использована при ускоренном параллельном выполнении различных задач.

Основными недостатками стандартного программного обеспечения на основе RTOS является то, что они страдают от нагрузки на вычислительные мощности и часто требуют большой объем памяти.

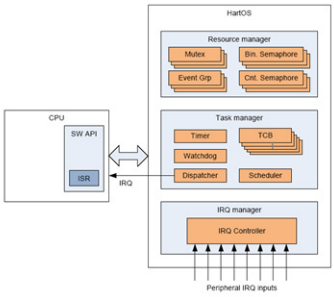
Требования высокой производительности (вычислительной мощности)RTOS обусловлены в основном блоком управления прерываниями, который снижает производительность с ростом количества задач и высоких частотах, но и планирования задач, ресурсов распределение и освобождение, обнаружение тупиков и различные другие функции ОС / API, потребуется время выполнения задач, выполняемых на CPU.

Внешняя обработка асинхронных прерываний также является источником индетерминизма, так как это будет вызывать неопределенность уровня приоритета, чтобы обеспечить завершение задачи в течение их времени выполнения.

Повышенные требования предъявляются к памяти программного обеспечения *RTOS*, отчасти это связано с необходимостью создания стека для каждой задачи, но и требования в объеме пространства как ОЗУ и ПЗУ, для хранения структурных данных *RTOS*, так как программный код может занимать достаточно большой объем.Разработанным решением является HartOS, аппаратное обеспечение реализовано в операционной системе реального времени, поддерживающей все основные функции, ожидаемые от полной RTOS без ущерба для гибкости.

Упрощенная блок-схема системы

HartOS состоит из двух главных частей: программное обеспечение *API* и твердое - ядро изделия. Рисунок показывает первоначальный проект блок-схемы системы.



Блок-схема системы

Программное обеспечение *API* будет содержать все функции интерфейса, необходимые для связи с ядром, а также *CPU* определенного функционала для обработки контекстных переключений, инициализирующих стековые фреймы и т.д.

Аппаратное ядро будет содержать всю функциональность *RTOS*. Его структура состоит из трёх главных частей: Диспетчер задач, Менеджер по IRQ/Прерыванию и Менеджер ресурсов.

Диспетчер задач будет решать все функции, необходимые для реализации базируемого ядра чистой задачи. Главные функции этого: Таймер, Сторожевой таймер, Диспетчер и модули Планировщика. Массив *TCB's* (Блок управления задачей) будет содержать необходимые данные, чтобы управлять и восстановить контекст задач в системе. Менеджер по *IRQ* реализует контроллер прерываний и логику, необходимую для взаимодействия через интерфейс с остальной частью ядра. Менеджер ресурсов должен управлять ресурсами ядра.

Особенности общейархитектуры

Как иллюстрировано в рисунке 1 *HartOS* будет состоять из трех главных (ядро) модулей: Диспетчер задач, диспетчер прерываний и менеджер ресурсов. Главная цель такой архитектуры состоит в том, чтобы сделать каждый из этих модулей максимально независимым, таким образом, модуль может быть легко удален, должна также быть, возможность обновить/изменить внутреннюю функциональность модуля, не затрагивая остальную часть системы.

Заключение

Исходя из рассмотренных публикаций, можно сказать, что основные недостатки программного обеспечения на основе *RTOS* могут быть ликвидированы путем реализации всего ядра операционной системы в режиме реального времени на аппаратном уровне. Продуманный дизайн и использование новейших технологий *FPGA*, позволяет реализовать полнофункциональные и гибкие аппаратные*RTOS (HartOS*).

Список литературы

1. J. Lee, I. Mooney, V.J., A. Daleby, K. Ingstrom, T. Klevin, and L. Lindh, “Acomparison of the rtu hardware rtos with a hardware/software rtos,” pp. 683 – 688, jan. 2003.

2. S. Nordstrom, L. Lindh, L. Johansson, and T. Skoglund, “Application specific real-time microkernel in hardware,” p. 4 pp., jun. 2005.

3. S. Nordstrom and L. Asplund, “Configurable hardware/software support for single processor real-time kernels,” pp. 1 –4, nov. 2007.

АНАЛИЗ СПЕЦИАЛИЗИРОВАННЫХ СИСТЕМ ОБРАБОТКИ ГРАФИЧЕСКОЙ ИНФОРМАЦИИ

Печаткин С.В., Грачева Е.В.

Пензенский государственный технологический университет, Пенза, Россия, e-mail: los@pgta.ru

Введение

Концепция параллельной обработки давно привлекала внимание специалистов своими потенциальными возможностями повышения производительности и надежности вычислительных систем. В последнее время эстафета параллельных вычислений перешла к массовому рынку, связанному с трёхмерными приложениями. Универсальные устройства с многоядерными процессорами для параллельных векторных вычислений, используемых в 3D-графике, достигают высокой пиковой производительности, которая универсальным процессорам не достижима.

Сравнение CPU и GPU в параллельных расчётах

Для 3D видеоускорителей ещё несколько лет назад появились первые технологии неграфических расчётов общего назначения GPGPU (General-Purpose computation on GPUs). Современные видеочипы содержат сотни математических исполнительных блоков, и эта мощь может использоваться для значительного ускорения множества интенсивных приложений. Современные GPU обладают гибкой архитектурой, что вместе с высокоуровневыми языками программирования и программно-аппаратными архитектурами, раскрывает эти возможности и делает их значительно более доступными.

На создание GPCPU разработчиков побудило появление достаточно быстрых и гибких шейдерных программ, которые способны исполнять современные видеочипы. Разработчики задумали сделать так, чтобы GPU рассчитывали не только изображение в 3D приложениях, но и применялись в других параллельных расчётах. В GPGPU для этого использовались графические API: OpenGL и Direct3D, когда данные к видеочипу передавались в виде текстур, а расчётные программы загружались в виде шейдеров. Недостатками такого метода является:

- сравнительно высокая сложность программирования;

- низкая скорость обмена данными между CPU и GPU и другие ограничения.

Вычисления на GPU развиваются очень быстро. Два основных производителя видеочипов, NVIDIA и AMD, разработали и анонсировали соответствующие платформы под названием CUDA (Compute Unified Device Architecture) и CTM (Close To Metal или AMD Stream Computing), соответственно. В отличие от предыдущих моделей программирования GPU, эти были выполнены с учётом прямого доступа к аппаратным возможностям видеокарт. Платформы не совместимы между собой, CUDA — это расширение языка программирования C, а CTM — виртуальная машина, исполняющая ассемблерный код. Вычисления с GPU-ускорением заключаются в использовании графического процессора (GPU) совместно с CPU для ускорения приложений в области науки, аналитики, проектирования и на предприятиях. Вычисления с GPU-ускорением были изобретены компанией NVIDIA в 2007 году, теперь GPU обеспечивают мощностью энергоэффективные центры обработки данных в правительственных лабораториях, университетах и на предприятиях всех размеров по всему миру. GPU ускоряют приложения на различных платформах, начиная от автомобилей, мобильных телефонов и планшетов и заканчивая беспилотными летательными аппаратами и роботами.

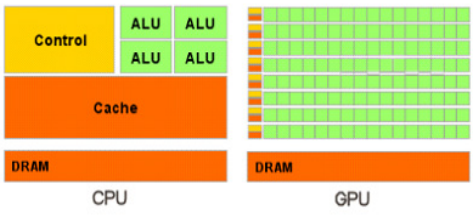
Компания NVIDIA выпустила платформу CUDA – C-подобный язык программирования со своим компилятором и библиотеками для вычислений на GPU. CUDA позволяет раскрыть все возможности и даёт программисту больший контроль над аппаратными возможностями GPU.

CUDA доступна на 32-битных и 64-битных операционных системах Linux, Windows и MacOS X.

Рост тактовых частот универсальных процессоров определен физическими ограничениями и высоким энергопотреблением. Увеличение их производительности чаще всего происходит за счёт использования нескольких ядер в одном чипе. Современные процессоры содержат до четырёх ядер и они предназначены для обычных приложений, используют MIMD – множественный поток команд и данных. Каждое ядро работает отдельно от остальных, исполняя разные инструкции для разных процессов.

В видеочипах NVIDIA основной блок – это мультипроцессор с восемью-десятью ядрами и сотнями ALU в целом, несколькими тысячами регистров и небольшим количеством разделяемой общей памяти. Кроме того, видеокарта содержит быструю глобальную память с доступом к ней всех мультипроцессоров, локальную память в каждом мультипроцессоре, а также специальную память для констант.

Самое главное – эти несколько ядер мультипроцессора в GPU являются SIMD (одиночный поток команд, множество потоков данных) ядрами. И эти ядра исполняют одни и те же инструкции одновременно, такой стиль программирования является обычным для графических алгоритмов и многих научных задач, но требует специфического программирования. Зато такой подход позволяет увеличить количество исполнительных блоков за счёт их упрощения.



Отличия в конфигурации CPU и GPU

Перечислим основные различия между архитектурами CPU и GPU. Ядра CPU созданы для исполнения одного потока последовательных инструкций с максимальной производительностью, а GPU проектируются для быстрого исполнения большого числа параллельно выполняемых потоков инструкций.

Универсальные процессоры оптимизированы для достижения высокой производительности единственного потока команд, обрабатывающего и целые числа и числа с плавающей точкой. При этом доступ к памяти случайный.

Разработчики CPU стараются добиться выполнения как можно большего числа инструкций параллельно, для увеличения производительности. Для этого, начиная с процессоров Intel Pentium, появилось суперскалярное выполнение, обеспечивающее выполнение двух инструкций за такт, а Pentium Pro отличился внеочередным выполнением инструкций. Но у параллельного выполнения последовательного потока инструкций есть определённые базовые ограничения и увеличением количества исполнительных блоков кратного увеличения скорости не добиться.

Видеочип принимает на входе группу полигонов, проводит все необходимые операции, и на выходе выдаёт пиксели. Обработка полигонов и пикселей независима, их можно обрабатывать параллельно, отдельно друг от друга. Поэтому, из-за изначально параллельной организации работы в GPU используется большое количество исполнительных блоков, которые легко загрузить, в отличие от последовательного потока инструкций для CPU. Кроме того, современные GPU также могут исполнять больше одной инструкции за такт (dual issue). Так, архитектура Tesla в некоторых условиях запускает на исполнение операции MAD+MUL или MAD+SFU одновременно.

GPU отличается от CPU ещё и по принципам доступа к памяти. В GPU он связанный и легко предсказуемый – если из памяти читается тексель текстуры, то через некоторое время придёт время и для соседних текселей. Да и при записи то же – пиксель записывается во фреймбуфер, и через несколько тактов будет записываться расположенный рядом с ним. Поэтому организация памяти отличается от той, что используется в CPU.

Работа с памятью у GPU и CPU несколько отличается. Так, не все центральные процессоры имеют встроенные контроллеры памяти, а у всех GPU обычно есть по несколько контроллеров.

Универсальные центральные процессоры используют кэш-память для увеличения производительности за счёт снижения задержек доступа к памяти, а GPU используют кэш или общую память для увеличения полосы пропускания. CPU снижают задержки доступа к памяти при помощи кэш-памяти большого размера, а также предсказания ветвлений кода. Эти аппаратные части занимают большую часть площади чипа и потребляют много энергии. Видеочипы обходят проблему задержек доступа к памяти при помощи одновременного исполнения тысяч потоков – в то время, когда один из потоков ожидает данных из памяти, видеочип может выполнять вычисления другого потока без ожидания и задержек. Есть множество различий и в поддержке многопоточности. CPU исполняет 1-2 потока вычислений на одно процессорное ядро, а видеочипы могут поддерживать до 1024 потоков на каждый мультипроцессор, которых в чипе несколько штук. И если переключение с одного потока на другой для CPU стоит сотни тактов, то GPU переключает несколько потоков за один такт. Кроме того, центральные процессоры используют SIMD (одна инструкция выполняется над многочисленными данными) блоки

для векторных вычислений, а видеочипы применяют SIMT (одна инструкция и несколько потоков) для скалярной обработки потоков. SIMT не требует, чтобы разработчик преобразовывал данные в векторы, и допускает произвольные ветвления в потоках.

Можно сказать, что в отличие от современных универсальных CPU, видеочипы предназначены для параллельных вычислений с большим количеством арифметических операций. И значительно большее число транзисторов GPU работает по прямому назначению – обработке массивов данных, а не управляет исполнением (flow control) немногочисленных последовательных вычислительных потоков. Это схема того, сколько места в CPU и GPU занимает разнообразная логика.

Выводы

В итоге, основой для эффективного использования мощи GPU в научных и иных неграфических расчётах является распараллеливание алгоритмов на сотни исполнительных блоков, имеющихся в видеочипах. А использование нескольких GPU даёт ещё больше вычислительных мощностей для решения подобных задач.

Будущее множества вычислений явно за параллельными алгоритмами, почти все новые решения и инициативы направлены в эту сторону. Но, конечно, GPU не заменят CPU. В их нынешнем виде они и не предназначены для этого. Сейчас что видеочипы движутся постепенно в сторону CPU, становясь всё более универсальными, так и CPU становятся всё более «параллельными», обзаводясь большим количеством ядер, технологиями многопоточности, не говоря про появление блоков SIMD и проектов гетерогенных процессоров. Скорее всего, GPU и CPU в будущем просто сольются. Известно, что многие компании, в том числе Intel и AMD работают над подобными проектами.

Список литературы

1. http://www.ixbt.com/video3/cuda-1.shtml;
2. http://www.nvidia.ru/object/gpu-computing-ru.html

БИОЛОГИЧЕСКИ АДАПТИВНЫЕ ЭЛЕКТРОННЫЕ ИМПЛАНТЫ

Рыжков С.С.

Пензенский государственный технологический университет
Пенза, Россия, e-mail: los@pgta.ru

Введение

В современной медицинской диагностике одним из передовых направлений является разработка имплантируемых датчиков, совместимых с живым организмом, которые будут практически не ощутимы пациентом. Некоторые исследователи в качестве решения предлагают устройства, способные разлагаться в теле. Однако в этом случае датчики будут иметь весьма ограниченные сроки службы, что не всегда соответствует исходным задачам.

Биологически адаптивные электронные импланты

Специалисты из Техасского университета в Далласе и Токийского университета создали биологически адаптивное, гибкое устройство, которое становится мягким при имплантации внутрь человеческого тела, что позволяет использовать его для диагностики и лечения различных мелких тканей, включая нервы и кровеносные сосуды. Таким образом, хирург может без труда имплантировать устройство, которое буквально оборачивает объёмные объекты внутри тела, не мешая их естественному функционированию.

Ранее врачи уже пытались устанавливать электронику в живом организме, но одной из проблем была её жёсткость, что совершенно не совместимо с биологической тканью.

Добиться необходимой гибкости и мягкости удалось благодаря так называемым «запоминающим полимерам» (memory polymers). Кроме этого для построения микросхем использовалась разработанная ранее гибкая фольга.

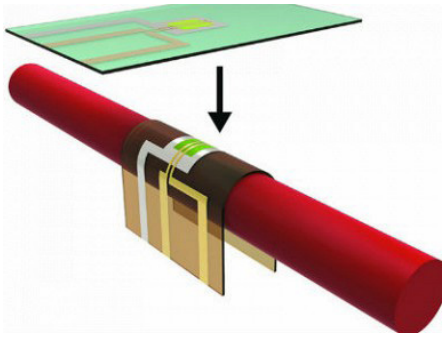


Рис. 1. Полимеры с памятью формы

Испытания нового элемента, проведенные на грызунах, показали, что его можно вживлять практически в любое место живой ткани. В дальнейшем ученые планируют на базе нового транзистора создать ряд датчиков и целых систем мониторинга, которые можно было бы вживлять в тело человека, не причиняя ему никакого дискомфорта в дальнейшем.

Ещё одним важным достижением стал сам метод создания транзистора на органической основе, который представляет собой адаптированный вариант технологии производства кремниевой электроники. Последнее сказывается на стоимости готового устройства в сторону уменьшения. Исследование является одним из первых демонстраций транзисторов, которые могут изменять форму и поддерживать свои электронные свойства после того как они имплантируются в тело.

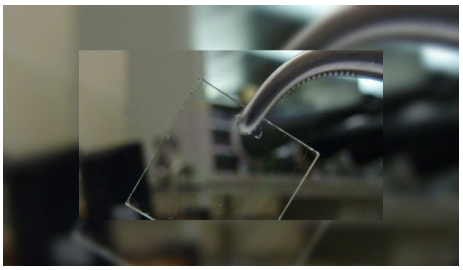


Рис. 2. Прозрачный органический полупроводник на стеклянной подложке

Эти разработки в будущем поможет врачам узнать больше о том, что происходит внутри тела, и стимулировать тело для лечения.

Ученые и врачи уже давно устанавливают электронику в организм на некоторое время, но одна из проблем заключается в том, что жесткость общих электронных компонентов не совместима с биологическими тканями человека. Необходимо устройство, которое будет жестким при комнатной температуре, что бы хирург мог имплантировать устройство, но мягким и достаточно гибким, чтобы обернуться вокруг 3-мерных объектов, чтобы организм продолжат вести себя именно так, как это было бы без устройства. Поставив электронику с изменяющейся формой из смягчающихся полимеров можно решить эту задачу.

Движущей силой этой технологии являются полимеры с памятью формы. Полимеры подстраиваются под среду организма, и становятся менее жесткими после имплантирования в организм.