Оптимизация разработки программ для встраиваемых систем в симуляторе Vista Virtual Prototyping

Анна Сергеева (Санкт-Петербург)

В статье рассмотрены функциональные возможности программного пакета Vista Virtual Prototyping компании Mentor Graphics, которые позволяют оптимизировать процесс разработки встраиваемого программного обеспечения за счёт ранней отладки кода на виртуальных прототипах аппаратных платформ, в том числе на базе распространённых процессоров ARM.

Введение

В настоящее время широко внедряются и используются всевозможные промышленные и бытовые встраиваемые системы, в том числе на базе процессоров ARM. Каждый ответственный производитель всегда стремится обеспечить высокое качество функционирования, столь критичное для подобных устройств. Каждый потребитель, в свою очередь, вправе рассчитывать на высокую надёжность, хорошую производительность и оптимальный режим работы приобретаемого изделия.

Должное внимание в процессе разработки уделяется как обеспечению качества самого оборудования (аппаратуры), так и установленных на нём рабочих программ. В данной статье рассматриваются особенности разработки программного обеспечения для встраиваемых систем с учётом следующих важных моментов.

В первую очередь, разработчики должны понимать, что ресурсы конкретного устройства, на котором будут запущены их программы, не безграничны. Следует также обеспечивать высокую производительность приложений, причём на всём протяжении процесса разработки [1]. Это значит, что необходимо принимать во внимание особенности конкретной аппаратуры, на которой будет развёрнуто и запущено программное обеспечение (ПО), в том числе производительность, архитектуру, количество ядер процессора и объём памяти.

Производительность системы зависит от того, какая встраиваемая ОС установлена, поскольку её особенности оказывают воздействие на работу ПО. При тестировании приложений может оказаться, что под разными ОС программы работают с разной скоростью и различными ошибками. В рамках обеспечения надёжной работы встраиваемой системы следует выполнять полную отладку и тестирование всего разрабатываемого пакета программ.

Потребности отрасли, как с точки зрения оптимизации цикла разработки проекта (сокращение сроков, превентивное выявление возможных проблем и сбоев), так и из соображений экономии (на приобретении физического оборудования на ранних стадиях испытаний работоспособности разрабатываемых программ), привели к необходимости в средствах создания быстрых, точных и дешёвых программных моделей, симулирующих работу аппаратуры до момента её физической реализации в «железе» [2]. Эти модели необходимо предоставлять команде разработчиков встроенного ПО на самых ранних стадиях процесса разработки.

Такие производители, как Keil или IAR, впрочем, как и другие разработчики встраиваемых устройств, предоставляют собственные симуляторы для разрабатываемых ими микроконтроллеров. Однако они симулируют, по сути, только «голое железо», ориентированы только на процессорное ядро и имеют ограниченный набор функций. В отличие от них рассматриваемая в этой статье программа Vista Virtual Prototyping (далее VVP) от Mentor Graphics предоставляет возможности симуляции аппаратуры, шины, производительности, работы операционной системы и программ. За счёт этого VVP является более удобной и информативной, чем большинство симуляторов, представленных на рынке.

Симуляторы аппаратного обеспечения Vista Virtual Prototyping

Система VVP даёт возможность инженерам выполнить проверку, анализ, интегрирование и оптимизацию разрабатываемого программного обеспечения, используя раннюю модель встраиваемой аппаратуры (до её физической реализации). Она отвечает требованиям, предъявляемым к современным средствам разработки и применения виртуальных прототипов, а также симуляции работы оборудования для разработки и отладки ПО для встраиваемых систем. В числе этих требований:

- проверка работы программ с использованием ранних моделей аппаратуры;
- обеспечение высокой скорости исполнения программного кода;
- поддержка исполняемых файлов для виртуальных прототипов, соответствующих промышленному стандарту SystemC TLM 2.0;
- поддержка большинства моделей TLM и платформ на их основе;
- поддержка работы с двухъядерным процессором ARM Cortex-A9;
- расширенные возможности анализа мощности и производительности оборудования;
- визуальное отображение всех ключевых регистров и атрибутов используемой аппаратуры.

Система VVP содержит компоненты Sourcery CodeBench (среда отладки разрабатываемого программного кода) и Sourcery Analyzer (анализатор программ, разрабатываемых для встраиваемых систем) [3]. Она использует первичную, абстрактную функциональную модель физического оборудования и предназначена для работы программистов до этапа реализации оборудования в «железе», т.е. предоставляет разработчикам встраиваемых программных продуктов возможность работы до приобретения аппаратного обеспечения. В дальнейшем система VVP позволяет выполнять отладку и тестирование программного обеспе-

чения и на этапах работы с реальным оборудованием, т.е. конечным продуктом. Принцип такого подхода иллюстрирует рисунок 1.

Система VVP даёт возможность запуска программного обеспечения на моделях встраиваемых систем, так же, как и стандартные пакеты поддержки плат (board support packages, BSPs), но вместе с тем предоставляет разработчику больше удобств [4]. Так, в ней реализованы возможности виртуального оборудования и управления симуляцией, что позволяет совместно отлаживать аппаратное и программное обеспечение, а также оптимизировать разрабатываемые программы для достижения необходимых характеристик потребления мощности и производительности встраиваемого устройства.

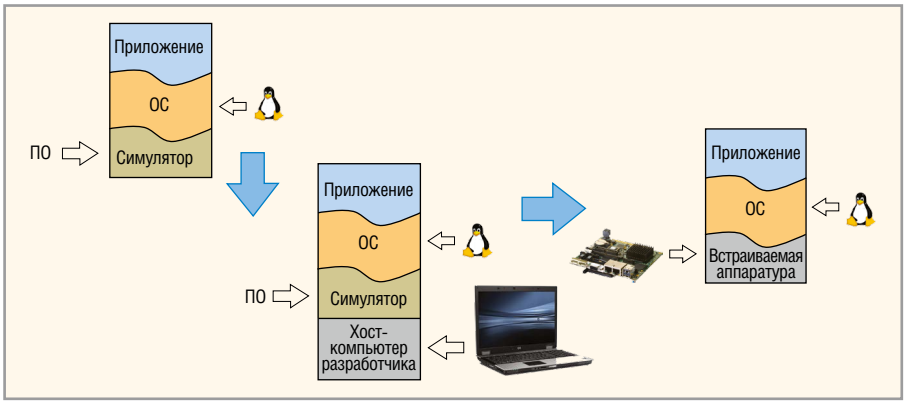

Рис. 1. Применение симуляторов при отладке ПО на виртуальной и реальной аппаратуре

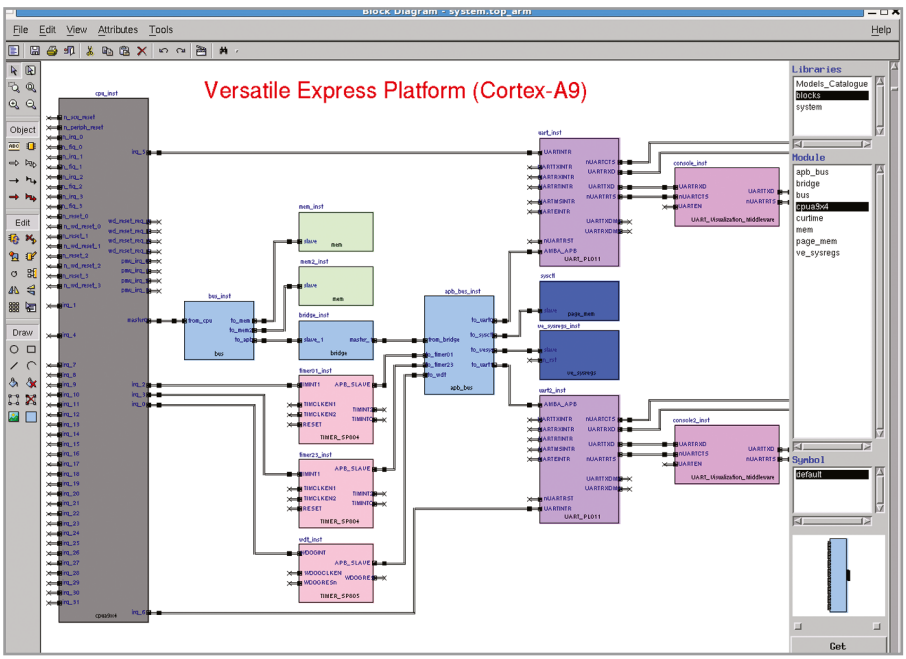

Рис. 2. Редактор блок-схем Vista Block Diagram editor

Построение платформы

Виртуальный прототип строится на базе платформы для моделирования на уровне транзакций (transaction-level modeling, TLM); TLM представляет первичную модель аппаратного обеспечения платформы и является абстрактным приближением к моделируемым функциональным узлам цифровых систем. Платформа TLM состоит из взаимосвязанных моделей уровня транзакций, представленных в формате структурированного кода на языке SystemC. Система VVP поставляется с набором библиотек, среди которых есть и библиотека предустановленных моделей уровня транзакций. Эти модули можно модифицировать или создавать новые. Таким образом, обеспечивается настройка ключевых атрибутов моделей TLM (таких как параметры мощности и время арбитража).

Построение платформы TLM

С помощью редактора блок-схем (Vista Block Diagram editor) легко и удобно создать платформу TLM. Структурированному коду на языке SystemC соответствует набор соединённых друг с другом графических символов, автоматически сгенерированных системой. Так формируется топология полного проекта платформы TLM, а также схемное представление и описание всех объектов разрабатываемого проекта. Интерфейс редактора приведён на рисунке 2.

Каждый раз, когда выполняется операция сохранения проекта, редактор блок-схем Vista Block Diagram editor автоматически генерирует структурированный код на языке SystemC, соответствующий схемному представлению модели. Отметим, что такой подход является более наглядным, интуитивно понятным и простым в использовании по сравнению с кодированием в стандартном текстовом редакторе.

Редактор блок-схем поддерживает большое число полезных операций, среди которых:

- выбор отдельных элементов или групп элементов;
- определение атрибутов и имён, присваиваемых по умолчанию, для графических элементов;
- изменение размеров, перемещение и копирование элементов;
- редактирование графических символов в индивидуальной форме редактирования либо на общей схеме.

Также редактор блок-схем позволяет:

- поворачивать, удалять и дублировать графические символы;
- добавлять и редактировать текст;
- добавлять новые символы, соответствующие модели TLM.

Эти и другие функции редактора Vista Block Diagram editor доступны из главного и контекстного меню и из панелей инструментов, что обеспечивает удобство использования программы для разработчиков платформ TLM.

Построение виртуального прототипа

Конечный виртуальный прототип, созданный на основе платформы TLM, является самостоятельным исполняемым файлом. Он создаётся либо с помощью диалога создания прототипа (Create Virtual Prototype Dialog) в Vista GUI, либо с помощью специальной команды *vista_create_vp*.

Система VVP позволяет определять параметры разрабатываемого проекта в специальном файле *Vista parameters*. Позже, во время работы прототипа, эти значения могут быть модифицированы (при этом повторное создание испол-

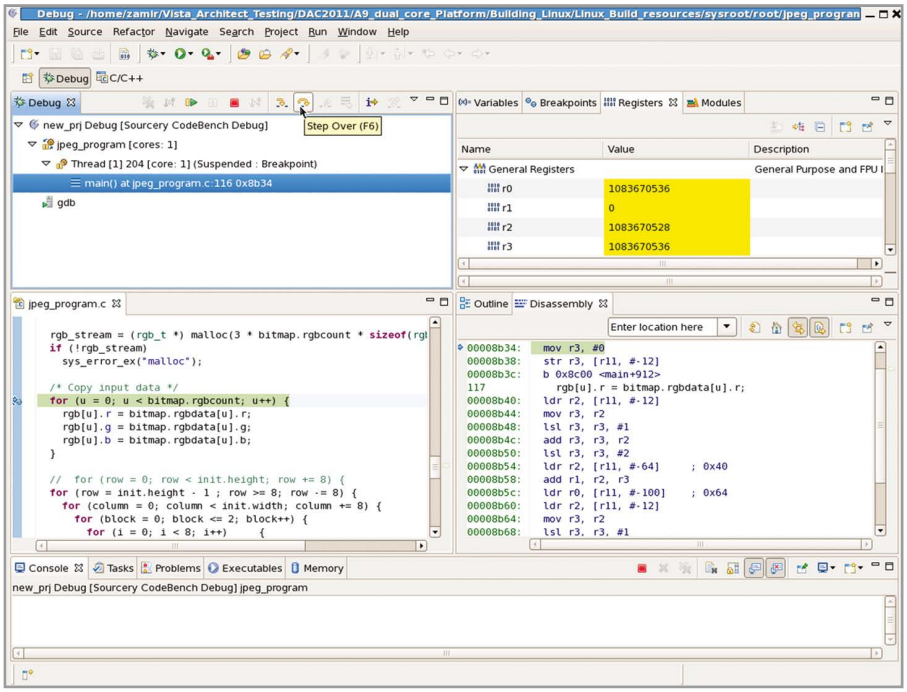

Рис. 3. Интерфейс пользователя Sourcery CodeBench для отладки аппаратного и программного обеспечения

няемого файла не требуется). Пользователи могут осуществлять настройку среды исполнения, которая используется во время симуляции виртуального прототипа: устанавливать значения по умолчанию для переменных окружения, копировать дополнительные папки и файлы. Виртуальные прототипы Vista можно запускать на рабочих станциях, работающих под ОС Linux или Windows.

Отметим также, что создатель платформы может контролировать использование виртуального прототипа определёнными пользователями, добавив к виртуальному прототипу дополнительное условие в лицензии.

Итак, перечислим коротко ключевые возможности системы Vista, предоставляемые при создании платформы:

- вместе с Vista поставляется предустановленная библиотека моделей TLM;
- атрибуты моделей, такие как характеристики синхронизации и мощности, доступны для пользовательской настройки;
- реализована возможность создавать и добавлять собственные модели TLM к имеющейся библиотеке;
- интерфейс интуитивно понятен для редактирования и визуального отображения создаваемой платформы;
- исполняемый файл виртуального прототипа создаётся автоматически;
- существует возможность запуска виртуальных прототипов под разными ОС (Linux и Windows).

Управление симуляцией и взаимосвязанная отладка программ и аппаратуры

Управление симуляцией

Запустить виртуальный прототип Vista можно либо из командной строки, либо из интегрированной среды разработки Sourcery CodeBench IDE. В режиме командной строки в качестве аргументов строки существует возможность передачи переменных среды запуска прототипа.

При работе в интегрированной среде разработки Sourcery CodeBench IDE можно получать визуальное представление аппаратуры, осуществлять взаимосвязанную отладку программной и аппаратной частей, выполнять взаимодействие с файловой системой. Эти функциональные возможности доступны при подключении к среде Sourcery CodeBench, которая взаимодействует с API виртуального прототипа. Интерфейс отладчика приведён на рисунке 3.

Разработка программного обеспечения

Пакет VVP обеспечивает разработку программного обеспечения, интеграцию и проверку функционирования на целевой аппаратуре, поскольку виртуальный прототип обладает функциональными возможностями целевых прототипов и пакетов поддержки плат (board support packages, BSPs).

Отлаживаемое программное обеспечение запускается на виртуальном прототипе точно так же, как оно позже будет запущено на аппаратуре. Обеспечена поддержка пользовательского интерфейса, стеков приложений, промежуточной прошивки и драйверов, запускаемых поверх ОС Linux, Android и Nucleus, а также в режиме без ОС – на «голом железе» (Bare-Metal). Система предоставляет все средства, необходимые для построения kernel-ядер Linux, а также для быстрой загрузки ОС.

Виртуальные прототипы Vista могут подключаться к физическим устройствам, таким как терминалы и дисплеи рабочих хост-компьютеров, обеспечивая управление прототипами. С помощью соединений, устанавливаемых через Ethernet и USB, виртуальный прототип можно запустить на хост-машинах, используя их в качестве реальной среды функционирования. В режиме полухостинга пользователь может выводить на экран сообщения о прохождении операций, отслеживать текущее время симуляции и устанавливать уровень детализации сообщений об ошибках.

Режимы симуляции аппаратуры и привязка ко времени исполнения

Система VVP может запускать симуляцию аппаратуры в двух режимах: в функциональном режиме и в режиме представления.

В функциональном режиме производится интеграция, верификация и отладка программного обеспечения. Режим представления позволяет проводить анализ и оптимизацию программного обеспечения для достижения лучшей производительности и снижения энергопотребления. Для работы в этих режимах модель виртуального прототипа строится на двух уровнях детализации синхронизации: синхронизация, не привязанная ко времени исполнения (loosely timed, LT), и синхронизация, учитывающая время исполнения (approximately timed, AT).

При работе в функциональном режиме виртуальные прототипы задействуют уровень синхронизации, не привязанной ко времени исполнения. Здесь каждая транзакция представляет собой завершённую передачу данных по аппаратной шине, вне зависимости от того, как именно возникла передача данных, и от того, сколько времени она заняла. В этом режиме скорость

симуляции измеряется в сотнях MIPS, что очень близко к скорости обработки в реальном времени.

При работе в режиме представления виртуальные прототипы задействуют уровень синхронизации, учитывающий время исполнения транзакций. Здесь каждая транзакция представляет фазу передачи данных с помощью конкретного протокола шины, применяемой в модели (например, фазы адресации и данных для записи или чтения по шине AHB). Однако следует учитывать, что такая повышенная точность в режиме представления существенно замедляет симуляцию виртуального прототипа по сравнению с функциональным режимом. Система VVP позволяет переключаться между двумя доступными режимами синхронизации непосредственно во время работы прототипа.

Функциональный режим является вполне достаточным для верификации и отладки большей части программного обеспечения верхнего уровня, такого как операционная система и уровни приложений. Для анализа мощности и производительности аппаратного обеспечения под программным управлением и для низкоуровневого программного кода (такого как различные драйверы и программы реального времени) требуется работа виртуального прототипа в режиме представления. Под программами реального времени понимаются те, для которых время исполнения программного кода зависит от задержек аппаратного обеспечения или от того, как данные разнесены по фазам синхронизации.

Визуальное представление аппаратуры

Интегрированная среда разработки Sourcery CodeBench IDE также обеспечивает визуальное представление и управление аппаратными объектами разрабатываемой платформы. Эти объекты включают все периферийные регистры и локальные переменные, которые декларированы при создании компонента модели TLM.

Для распознавания объектов используются их иерархические пути, а связанные с ними величины представлены в отдельных ветках дерева, доступного в окне просмотра регистров Sourcery CodeBench Register. Все значения объектов являются редактируемыми; в процессе отладки можно добавлять новые значения.

Взаимосвязанная отладка программ и аппаратуры

Пользователи могут выполнять взаимосвязанную отладку аппаратной и программной частей, устанавливая точки прерывания в аппаратном обеспечении, чтобы остановить его симуляцию при условии достижения точки останова. Позже симуляция может быть возобновлена, поскольку программный отладчик останавливается после завершения обработки инструкции, которая и привела к останову. Это обеспечивает проверку состояния программы в данной точке и переход к следующим программным инструкциям с одновременным просмотром состояния аппаратных объектов, которые изменяются в зависимости от выполнения той или иной программной инструкции.

Управление симуляцией аппаратуры

Для управления симуляцией аппаратуры имеется набор команд, доступных из консоли Sourcery CodeBench. С помощью этих команд пользователь может:

- выполнить перезапуск ядра ЦП, обрабатывающего программный код, или произвести перезапуск всех узлов платформы, включая ядра процессора;
- установить точку прерывания, которая остановит симуляцию по завершении обработки инструкции, получившей доступ к аппаратному объекту (регистру или переменной). Точка прерывания подобна триггеру, срабатывающему на чтение, запись или на оба типа доступа;
- запросить местонахождение точки прерывания и получить к нему иерархический путь;
- удалить точку прерывания;
- загрузить готовый файл в формате ELF в память текущего ядра ЦП;
- управлять режимом запуска платформы;
- отображать режим функционирования, который установлен для платформы в данный момент;
- отключить обработку DMI (Direct Memory Interface), что (в функциональном режиме) позволит быстрее отслеживать транзакции через шинную архитектуру платформы и при желании отображать их на встроенном осциллографе Vista Waveform;
- отображать, установлена ли платформа в режим DMI;
- отображать текущее время симуляции кода SystemC;
- отображать информацию о процессе симуляции;
- прерывать процесс симуляции с дополнительно заданным кодом выхода.

Дополнительно система VVP позволяет управлять трассировкой ядер ЦП с помощью предустановленных вызовов (callbacks) по значащим событиям (например, когда ядро перешло в режим ожидания). Это полезно для дополнительной отладки и анализа, задаваемого пользователем. Также для управления симуляцией обеспечивается доступ по API с поддержкой полухостинга и других полезных задач.

В системе VVP можно манипулировать файлами посредством запросов встроенной операционной системы, загруженной на виртуальный прототип. Это позволяет разрабатывать, выполнять сборку и отлаживать программные пакеты на хост-машине, затем получать к ним доступ с консоли целевой ОС без необходимости повторной симуляции и перезапуска ОС на виртуальном прототипе. Например, файлы могут копироваться с файловой системы хост-компьютера в целевую локальную папку на прототипе (или наоборот) с использованием команды *cp*, вызываемой из встроенной ОС Linux.

Ещё раз коротко перечислим ключевые возможности системы, предоставляемые для совместной отладки аппаратуры и программ, а также для управления симуляцией:

- управление симуляцией с использованием интегрированной среды разработки Sourcery CodeBench IDE;
- выбор и переключение между двумя доступными режимами синхронизации (функциональным и режимом представления) может осуществляться непосредственно во время работы прототипа;
- визуальное отображение и управление виртуальной аппаратурой из среды Sourcery CodeBench IDE;
- остановка симуляции в точках прерывания, устанавливаемых в аппаратном или программном обеспечении;
- манипуляции с файлами для встроенных операционных систем;
- управление симуляцией из командной строки с использованием набора специальных команд.

Рис. 4. Просмотр состояния ЦП в окне Sourcery Analyzer

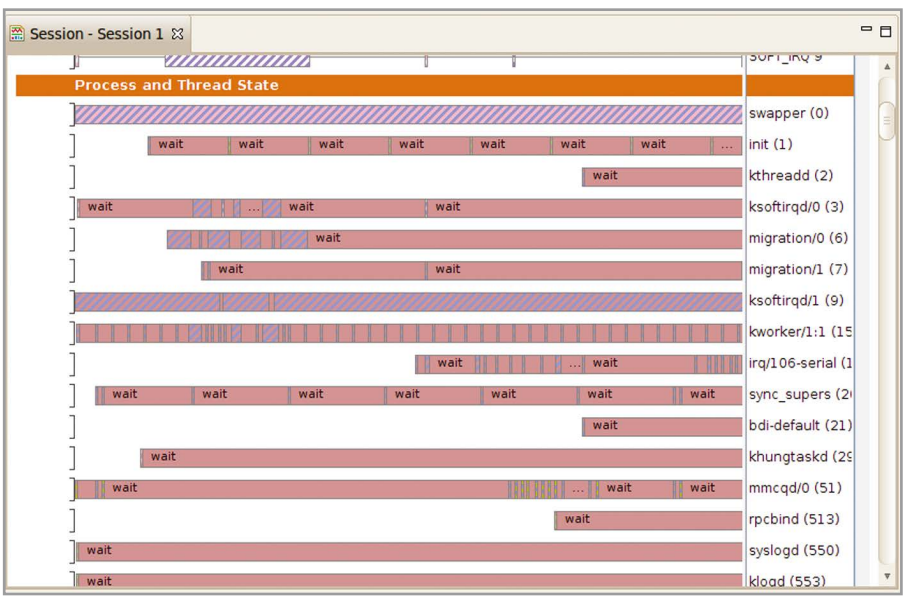

Рис. 5. Просмотр состояния потоков и процессов в окне Sourcery Analyzer

Возможности анализа виртуальной аппаратуры и разрабатываемых программ

Программный пакет VVP предоставляет богатый набор функциональных возможностей для анализа встроенного программного обеспечения и аппаратуры с программным управлением. Результаты анализа могут быть представлены для просмотра в виде графиков и таблиц, а также в формате сводных отчётов.

Анализ программного обеспечения

Анализ программного обеспечения в системе VVP осуществляется с помощью программы Mentor Embedded Sourcery Analyzer. Она работает с виртуальными прототипами Vista, интегрирует данные от одноядерных и многоядерных ЦП, поддерживает операционные системы типа Linux и RTOS, а также работает в режиме без операционной системы – на «голом железе» (Bare-Metal).

Программа Sourcery Analyzer включает библиотеку популярных и интуитивно понятных инструментов системного анализа и визуализации. Эти инструменты содержат большинство возможностей, востребованных программистами для оценки влияния функционирования ЦП/ядра и программ на работоспособность, производительность и мощность конечного продукта. Они позволяют отслеживать статистику и состояние ЦП, активность файловой системы во времени, функциональные вызовы и статистику по ним, запасы задержки, таймеры ожидания блокировки и удержания, а также состояния потоков и процессов.

С помощью программы Sourcery Analyzer пользователь может изготовить свои инструменты отладки и воздействия на производительность. Написанные на языке Java, они помогают обеспечить анализ и визуализацию, характерную для конкретных приложений, а также оптимизировать производительность всего проекта. Программа предоставляет доступ ко всему API, который использует все встроенные аналитические программы данного инструмента. Встроенный мастер помогает создавать новые аналитические программы.

Наиболее важные аналитические характеристики программы Sourcery Analyzer:

- состояние ЦП/ядра – работа (busy), бездействие (idle), прерывание процесса (см. рис. 4);
- статистика ЦП показывает время, которое каждое из ядер находилось в том или ином состоянии;
- активность файловой системы во времени;
- функциональные вызовы. Когда и какие функции были запущены;
- статистика функциональных вызовов. Какие функциональные вызовы, кем и как часто осуществлялись;
- запас задержки. На каких элементах израсходован запас задержки;
- таймеры ожидания, блокировки и удержания. Сколько времени занимает выполнение операции блокировки, и как долго сохранялось удержание;
- состояние потоков и процессов. Когда программные потоки были запущены, когда они были в режиме сна и когда осуществляли системные вызовы (см. рис. 5);
- размещение памяти приложения. Использование «кучи» (таблицы без индексов);
- уровень (частота) сбоев страницы памяти;
- мощность виртуального прототипа. Мощность, потребляемая виртуальным прототипом (общая и по элементам);
- коэффициент успешных/неуспешных обращений к кэш-памяти виртуального прототипа. Отношение числа успешных и неуспешных обращений к кэш-памяти процессора виртуального прототипа (также раздельно для режимов READ и WRITE).

Эти функциональные возможности в сочетании с атрибутами аппаратного обеспечения, такими как мощность и коэффициент успешных и неуспешных обращений к кэш-памяти, позволяют анализировать влияние работы программного обеспечения на функциональность, производительность и потребляемую мощность разрабатываемого изделия.

Анализ виртуальной аппаратуры

Средства анализа виртуальной аппаратуры в Vista Virtual Prototype позволя-

ют отслеживать следующие ключевые атрибуты разрабатываемого проекта:

- пропускную способность (Throughput). Уровень активности выбранных объектов, измеряемый как число транзакций или число переданных байтов за указанный период времени;
- задержку (Latency). Среднее время задержки за указанный период времени, необходимое для выполнения транзакции указанного типа (например, чтение или прерывание). Время задержки отображается для каждой транзакции выбранного типа;
- мощность (Power). Динамическая мощность, статическая мощность (утечка) и мощность, расходуемая для дерева синхронизации. Отобразится для полной схемы, а также и для выбранных элементов схемы;
- распределение мощности (Power Distribution). Вклад каждого элемента в суммарное энергопотребление;
- атрибуты (Attributes of User Customized Analysis) (аналитические данные, настраиваемые пользователем). Значения атрибутов с изменением времени. Атрибутами могут быть определяемые пользователем переменные, заданные в модели TLM;
- пропускную способность шины/полосу пропускания (Bus Throughput/Bandwidth). Для каждой шины, добавленной в разрабатываемую схему, можно открыть окно аналитических данных о её пропускной способности (Bus Throughput analysis view);
- анализ уровня конкуренции (Contention Level Analysis). Предоставляет взвешенное по времени среднее значение запросов, ожидающих доступа к тому или иному компоненту схемы, подключённому к сокету шины;
- анализ уровня арбитража (Arbitration Level Analysis). Предоставляет среднее время, необходимое для доступа транзакций к шине по заданному сокету;
- коэффициент успешных/неуспешных обращений к кэш-памяти виртуального прототипа (Cache Hit/Miss Ratio Analysis). Отношение числа успешных и неуспешных обращений к кэш-памяти процессора виртуального прототипа (также раздельно для режимов READ и WRITE).

Анализатор VVP предоставляет значительную гибкость в выборе отображаемых данных. Для графиков может варьироваться степень детализации, а также установка начальных и конечных значений. При каждом новом просмотре все данные обновляются. При движении указателем мыши по графику можно увидеть значение данных в каждый момент времени, произвести замер изменения значения между двумя точками на графике, как по вертикали, так и по горизонтали. Можно изменять масштаб графиков, как по горизонтали, так и по вертикали, чтобы задавать требуемую точность измерений.

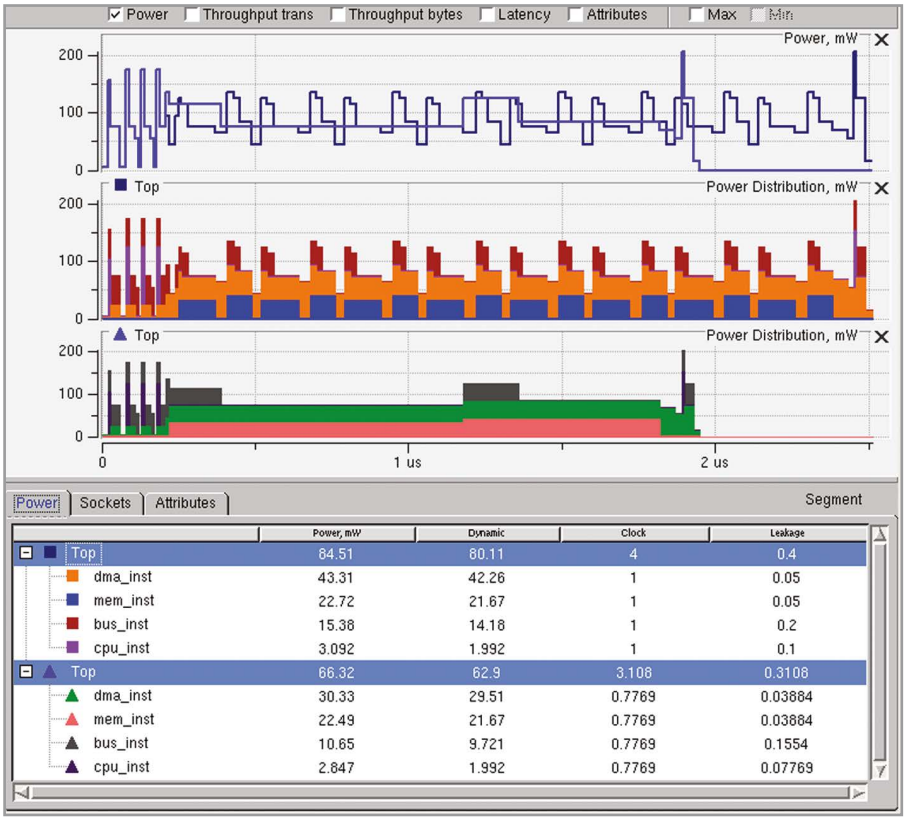

Рис. 6. Сравнение распределения мощности для двух архитектур

Анализатор VVP позволяет выполнять анализ и сравнение результатов нескольких сеансов симуляции, чтобы определить эффективность изменения системных конфигураций, выбора протокола и программных изменений (см. рис. 6).

Заключение

Симуляторы аппаратуры, с одной стороны, являются необходимыми инструментами для создания надёжных встраиваемых систем. На самых ранних этапах проектирования и разработки программ и аппаратуры они позволяют значительно сократить сроки работы над проектом и в то же время повысить качество производимого продукта.

С другой стороны, симуляторы весьма удобны в работе, поскольку не требуют привязки к конкретному оборудованию и позволяют программистам создавать свой код ещё до того, как будет изготовлено реальное оборудование.

Основные функциональные возможности программного пакета Vista Virtual Prototyping:

- поддержка операционных систем Linux, RTOS, а также возможность работы в режиме без операционной системы – на «голом железе» (Bare-Metal);
- использование анализатора Sourcery Analyzer и встроенных агентов – анализаторов;
- отслеживание и отображение ключевых аналитических данных, связанных с состояниями ЦП, функциональными вызовами и обрабатываемыми процессами;
- отслеживание характеристик производительности аппаратного обеспечения, таких как полоса пропускания, задержка и потребляемая мощность.

Литература

1. *Сергеева А.* Тестирование работоспособности промышленного компьютера. Компоненты и технологии. № 2. 2014.
2. *Сергеева А.* Одновременная разработка программ и аппаратуры для встраиваемых систем при помощи симулятора аппаратуры Vista Virtual Prototyping. Компоненты и технологии. № 3. 2014.
3. www.mentor.com/esl.
4. www.mentor.com/embedded-software.