Ускорение отладки устройств на базе Linux при помощи внутрисхемной отладки JTAG

Йоахим Хампп (Германия)

В статье обсуждаются проблемы, возникающие при отладке встраиваемых систем на базе ОС Linux, и преимущества интегрированных отладочных решений на базе JTAG от компании Wind River, позволяющие ускорить разработку и снизить производственные издержки.

Введение

Традиционно отладка систем на базе Embedded Linux требовала совместной работы программных и аппаратных средств – JTAG-инструментария для ввода оборудования в действие и решений на основе отладочных агентов для разработки программ. Эти инструменты часто решали основные проблемы, но не предназначались для интегрированной разработки Linux.

Компания Wind River изменила способ, которым разработчики могут отлаживать Linux, объединив традиционные средства JTAG-отладки аппаратуры с конфигурированием ядра Linux, управлением исправлениями, а также разработкой, отладкой и анализом приложений пользовательского уровня в единой инструментальной среде (IDE), известной как Wind River Workbench. Такая функциональность позволяет разработчикам использовать JTAG-соединение там, где традиционные решения на базе отладочных агентов технически или экономически не оправданы.

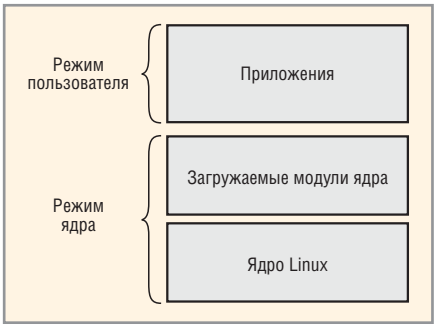

Рис. 1. Компоненты ОС Linux

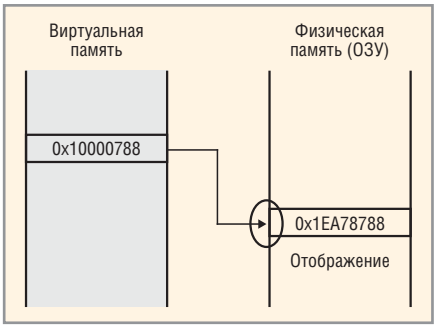

Рис. 2. Управление памятью в Linux

Два наиболее распространённых сценария, где может быть использовано JTAG-соединение, – невозможность подключиться к отладочному агенту из-за отсутствия соединения Ethernet или необходимость отладки в системном режиме при разрешении проблем в пространстве пользователя или ядре Linux. Новое поколение инструментария JTAG от компании Wind River позволяет отлаживать аппаратуру, начальный загрузчик, ядро Linux и приложения пользователя, выявляя аварийные события в системе и другие проблемы, которые возникают между ядром, приложением и целевой системой. Более того, разработчики, использующие операционную систему (ОС) Wind River Linux, могут свободно переключаться между отладкой с помощью агента и отладкой с помощью соединения JTAG. Эти нововведения во встроенной отладке предоставляют эффективные альтернативы для проектов, где отладка на основе агентов является либо недоступной, либо слишком затратной.

Краткий обзор ОС Linux

Первым шагом в устранении неполадок Linux является понимание внутреннего устройства самой ОС Linux и областей, в которых отладчику требуется анализировать взаимодействие компонентов. В общем случае ОС Linux состоит из множества взаимосвязанных модулей и использует сложные механизмы управления памятью. Отладка встраиваемой системы на базе Linux требует мощных инструментальных средств, которые выявляют особенности работы ОС и её взаимодействия с оборудованием.

Основными компонентами ОС Linux являются ядро, модули ядра и прикладное программное обеспечение (ПО). Ядро Linux является стержнем ОС и имеет высшие привилегии. Модули ядра Linux являются элементами ОС, которые динамически загружаются и выгружаются и часто используются для реализации драйверов устройств. После загрузки модуль ядра Linux получает такие же привилегии, как у ядра (см. рис. 1). Ядро Linux разделяет всё программное обеспечение для выполнения в двух возможных режимах – режиме ядра или режиме пользователя. Приложения, выполняющиеся в режиме пользователя, имеют ограниченные привилегии. В частности, они не могут напрямую обращаться к памяти ядра или оборудованию, что позволяет предотвратить возможные сбои основной системы от действий приложения. Таким образом, за все действия системного уровня отвечает ядро. Это обеспечивает приложениям контролируемый доступ к системным ресурсам, включая память и периферийные устройства.

Обычно в системе Linux приложения выполняются в режиме пользователя, а драйверы устройств – в режиме ядра. Такой подход позволяет защитить ядро (Linux kernel) от потенциальных отказов системы и задержек в обслуживании, но имеет свои недостатки, в частности, снижение быстродействия, поскольку модули ядра Linux, как правило, имеют более высокую производительность, чем приложения пользователя.

В отличие от используемых во многих встраиваемых устройствах более простых 8-, 16- и 32-разрядных ОС с «плоской» или «статической» моделью памяти, в ОС Linux реализована передовая технология виртуальной адресации (см. рис. 2). Виртуальная адресация позволяет модулям ядра Linux и загруженным приложениям выполняться в системе как бы в непрерывном пространстве адресов, хотя реально программные модули могут быть разнесены в памяти по различным физическим областям. Отображение и выделение памяти для загружаемых модулей ядра и пользовательских при-

ложений осуществляет ядро Linux. Статическое отображение, используемое для ядра и загружаемых модулей, подразумевает, что виртуальный и физический адреса отличаются на заданную постоянную величину (например, смещение). При динамическом отображении, которое используется для пользовательских приложений, могут применяться более сложные механизмы отображения данных в физической памяти.

Ядро Linux также обладает способностью динамически выделять память под выполняемые процессы, которые могут включать в себя модули ядра Linux и пользовательские приложения. Эти процессы могут быть загружены в память для выполнения конкретной задачи, а затем удалены, когда в них нет необходимости, обеспечивая более эффективное распределение памяти в системе. Следовательно, при передаче памяти от приложения к ядру или модулям ядра Linux следует принимать во внимание возможности ядра Linux по статической и динамической трансляции адреса, а также способность ядра динамически размещать процессы в памяти.

Сложности отладки встраиваемых систем Linux

Масштабы использования операционной системы Linux во встраиваемых устройствах продолжают быстро расти. Согласно маркетинговому исследованию [1], 23% опрошенных компаний планируют использовать Linux в своих новых проектах. Отладка проектов с встраиваемой ОС Linux может быть чрезвычайно сложной задачей, поскольку разработка затрагивает и начальный загрузчик, и ядро Linux, и модули ядра, и приложения. Проблемы, в которых должен разобраться программист, включают файл конфигурации целевой системы для загрузчика, адресацию памяти в режиме ядра и в режиме пользователя, перемещение глобальных идентификаторов ядра, диагностику дефектов, охватывающих пространства ядра и пользователя. Стандартные решения на базе отладочных агентов (GDB, KGDB) не позволяют эффективно решить ни одну из этих задач.

Начальный загрузчик

Операционная система Linux для запуска использует начальный загрузчик. Он представляет собой код, который постоянно находится во флэш-памяти (или любой другой энергонезависимой памяти) и выполняется непосредственно после включения питания или аппаратного сброса. При запуске загрузчик копирует себя в ОЗУ и выполняет все необходимые предварительные операции – низкоуровневую инициализацию процессора, памяти и периферийных устройств – и после этого запускает ОС Linux.

Проблемы отладки начального загрузчика

Отладка начального загрузчика может быть достаточно сложной задачей, поскольку он является аппаратно зависимым, и разработчики должны обеспечить его перемещение из флэш-памяти в ОЗУ после запуска. Современные системы на кристалле (SoC) содержат сотни конфигурационных регистров, которые необходимо правильно инициализировать. Это требует изучения тысяч страниц технической документации в поисках конкретных параметров. Установка неправильных значений регистров может привести к последующим проблемам при запуске

ядра Linux или отладке приложений. Ручное редактирование содержимого регистров является весьма утомительным процессом.

Другая распространённая проблема, возникающая при разработке начального загрузчика, связана с вопросами отладки, которые появляются при загрузке образа Linux в ОЗУ и запуске ОС. После загрузки ядра Linux в физическую память ядро создаёт свою собственную карту виртуальной памяти, которая часто затрудняет определение, какие адреса использует ядро. Как следствие, на момент начала выполнения ядра Linux в ОЗУ могут быть проблемы при настройке таблицы идентификаторов ОС Linux. К сожалению, отладочные решения на базе агентов не поддерживают отладку начального загрузчика, поскольку ОС в процессе загрузки не работает. Поэтому разработчики используют средства JTAG для отладки этого критического процесса.

Упрощение отладки начального загрузчика

Решения на базе отладочных агентов (таких как GDB и KGDB) не будут работать для отладки начального загрузчика. Однако отладочные решения на базе JTAG обеспечивают мощные возможности, помогающие разработчиками быстро и эффективно выявлять проблемы и тестировать начальный загрузчик. При помощи средств отладки JTAG легко изучать и устанавливать значения регистров; способность задавать аппаратные точки останова позволяет пошагово выполнять код во флэш-памяти, чтобы быстро выявить источники ошибок. Ещё более упрощают процесс отладки интегрированные среды (IDE), которые поддерживают режимы дизассемблирования и смешанного отображения кода, перемежая исходный текст кодом ассемблера. Эффективное управление идентификаторами позволяет отлаживать код с учётом его перемещения из флэш-памяти в ОЗУ.

Дополнительным преимуществом некоторых JTAG-решений является способность загружать ядро Linux без помощи начального загрузчика. Это достигается благодаря использованию строки загрузочных параметров (boot line), и является особенно полезным в тех проектах, где разработку системы начинают до того, как будет готов и доступен загрузчик. Решения на базе JTAG, поддерживающие передачу строки загрузочных параметров, позволяют распараллелить разработку загрузчика и стабилизацию ОС, сокращая сроки разработки проекта.

Ядро Linux и модули ядра

Ядро Linux и модули ядра являются основными компонентами ОС Linux. Ядро стартует непосредственно после того, как начальный загрузчик инициализирует систему. Модули подгружаются позже, по необходимости.

На стадии стабилизации ОС разработчики занимаются оптимизацией и адаптацией ядра ОС Linux, а также разработкой модулей ядра. При этом взаимодействию оборудования и программного обеспечения уделяется особое внимание. Для отладки ядра требуется подробная картина состояния регистров, кэш-памяти и много другой низкоуровневой информации. Отладка загружаемых модулей также требует доступа к коду инициализации, т.к. она сопряжена с динамическим распределением памяти в ОЗУ.

Чтобы обеспечить работу агента KGDB, необходимо стабильное ядро Linux и драйверы интерфейсов. Отладочный агент не позволяет наблюдать работу оборудования и не предоставляет полной диагностической информации, требуемой для понимания взаимодействия ядра с аппаратурой. Для использования агентов необходимо инструментирование кода ядра, а это может вызывать побочные эффекты во встраиваемых устройствах.

Другие соображения, касающиеся использования агентов для отладки ядра Linux и модулей ядра, относятся к ограниченной функциональности точек останова. Например, KGDB не остановит центральный процессор (особенно многоядерный), чтобы дать разработчику возможность изучить состояние системы; KGDB не поможет диагностировать критический сбой системы, т.к. агенту для работы требуется, чтобы ядро было функционально. Кроме того, агенту KGDB требуется порт связи, такой как Ethernet, чтобы подключить хост к целевой системе. Таким образом, чтобы использовать KGDB для отладки в режиме ядра, необходимо иметь работающий стек протоколов TCP/IP, стабильное ядро Linux и функционирующие драйверы устройств. Это создаёт потенциальную проблему, поскольку на этапе стабилизации ОС такие средства связи могут быть недоступны или сами могут нуждаться в отладке.

Использование JTAG для отладки

Для проверки и стабилизации ядра Linux на целевой системе необходимы сложные отладочные решения, работающие с ядром Linux и модулями ядра. Отладочные решения на базе JTAG чрезвычайно полезны в этом рабочем процессе. Функции инструментария JTAG-отладки включают способность отслеживать локальные и глобальные переменные и регистры, а также просматривать данные и содержимое кэш-памяти команд. Некоторые коммерческие отладчики JTAG обеспечивают прозрачное отображение виртуальных адресов в физические, позволяя разработчику отслеживать содержимое ОЗУ по правильным адресам в памяти. Эти решения допускают отладку инициализации модулей ядра и могут многократно загружать и выгружать их без необходимости отключения и повторного подключения к целевой системе.

Важной функцией отладочных решений JTAG является их способность обеспечить полный останов операционной системы и приложений. Известная как отладка в режиме системы, эта функция чрезвычайно полезна для отладки ядра Linux и его модулей. В таком режиме отладки разработчик может останавливать всю систему, включая процессор, ОС, все потоки и обработку прерываний. Остановив систему таким способом, можно получить подробную визуализацию состояния оборудования и ПО, а также перезапустить систему и пошагово пройти код после перезагрузки.

Главным отличием отладочных решений на базе JTAG является то, что они будут работать там, где не может работать KGDB, – особенно в условиях критической ошибки ядра Linux и аварии целевой системы. Это делает их незаменимыми в задачах стабилизации драйверов устройств и ОС.

Отладка Linux-приложений

Приложения Linux представляют собой пользовательские программы, выполняющиеся под управлением ядра Linux. Они получают доступ к ресурсам, выполняя системные вызовы. Ядро Linux обрабатывает системные вызовы и решает, как предоставить приложению доступ к памяти и периферийным устройствам (см. рис. 3).

Для отладки пользовательских приложений разработчикам необходим прямой доступ к потокам, включая способность останавливать и возобновлять любой поток и просматривать его переменные и стек. Поскольку каждое приложение может состоять из множества процессов и потоков, может понадобиться останов всех потоков, связанных с отлаживаемым потоком приложения, включая потоки, которые могут влиять на данное приложение. Также может возникнуть необходимость в контроле регистров периферии вместе с различными процессами и центральными процессорами. Агент GDB не способен остановить всю систему или несколько потоков одновременно.

Разработчикам, отлаживающим пользовательские приложения, может потребоваться пошаговое выполнение системных вызовов, с динамическим переключением из режима пользователя в режим ядра и обратно. Здесь важно правильно отслеживать адресацию ОЗУ в моменты переходов, учитывая применяемый в Linux механизм виртуальной памяти. Полагаться на фиксированные физические адреса нельзя, – необходимо правильно интерпретировать выделение и отображение памяти. Как следствие, отладка системных вызовов при помощи агентов потребует одновременного использования GDB и KGDB. Использование нескольких агентов одновременно может значительно усложнить процесс отладки.

В некоторых встраиваемых устройствах не предусмотрено ни порта Ethernet, ни последовательного интерфейса. В таких случаях использование отладочных агентов невозможно. Даже если устройство имеет порт связи, отладочному агенту требуется его готовность. Если порт не готов, или требует отладки, или недостаточно памяти для работы IP-стека, отладочные агенты работать не будут.

Отладочные решения на основе JTAG предоставляют детальную видимость происходящего в приложениях, которые работают в режиме пользователя Linux. Для приложений, которые выполняют системные вызовы, способность JTAG-решений вести отладку и в режиме ядра, и в режиме пользователя даёт чёткую картину происходящего в обоих режимах. Всё это происходит без какого-либо инструментирования ядра Linux. «Двухрежимное» отладочное решение позволяет отслеживать все потоки приложения, их контексты, входы в ядро Linux, а также все используемые параметры и переменные. Для устройств, не оснащённых портом связи (например, мобильных и медицинские приборов, автомобильных систем и т.п.), отладочные решения на базе JTAG являются превосходной альтернативой отладке с использованием агентов.

Поддержка отладки в системном режиме упрощает отладку многопоточных приложений, т.к. позволяет полностью остановить процессор и исследовать состояние ОС и всех потоков на момент останова. Как уже упоминалось выше, множество проблем отладки связано с взаимодействием нескольких потоков. Неспособность

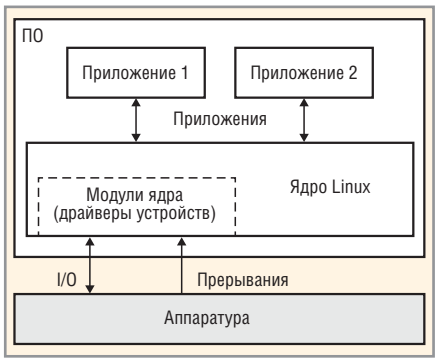

Рис. 3. Взаимодействие ПО с аппаратурой в Linux

отладочного агента остановить все потоки одновременно значительно усложняет локализацию проблем, что, в свою очередь, может существенно увеличить время отладки проекта. Отладка в системном режиме даёт детальное представление о состоянии системы в данный момент времени (состояние каждого потока, переменных и т.п.) и поэтому является предпочтительной.

Решение на базе отладчика JTAG является неразрушающим соединением с целевым оборудованием. JTAG-отладчик можно подключить к уже работающей системе, в которой только что произошел сбой, не меняя при этом состояния регистров процессора, и синхронизировать контексты для ядра и приложений, чтобы выполнить отладку. Одним из примеров является система, вошедшая в состояние взаимной блокировки потоков (deadlock). При помощи JTAG-отладчика разработчик может подсоединиться к целевой системе, не изменяя её состояния, и получить доступ к объектам ядра Linux, контекстам приложений и про-

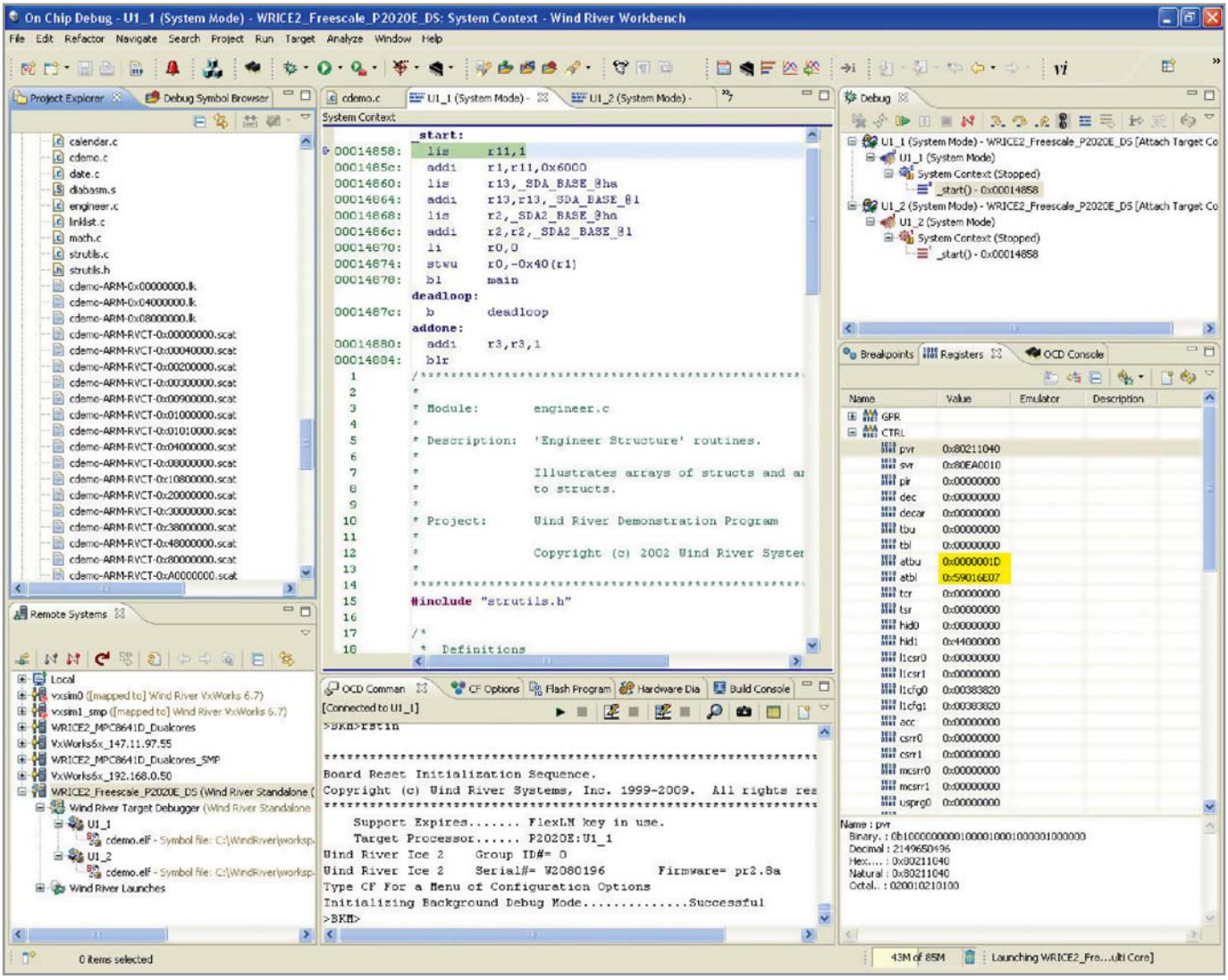

Рис. 4. Отладочное решение Wind River на базе JTAG

чей информации, которая позволит выявить, какие конкретно потоки, системные вызовы и параметры привели к возникновению ситуации отказа. Такое «сквозное» решение особенно полезно в тех случаях, где нельзя использовать отладку на основе агентов. Отладка с помощью JTAG существенно экономит время разработчиков за счёт того, что используемый инструментарий прост и понятен.

Рис. 5. IDE на основе Eclipse для отладки приложений Linux

Решения Wind River для JTAG-отладки систем на базе Linux

Комплекс JTAG для внутрисхемной отладки от компании Wind River не заменяет, а дополняет существующие решения на базе отладочных агентов (см. рис. 4). Решения, не поддерживающие JTAG, например, GDB и KGDB, могут оказаться неподходящими для ранних стадий разработки, таких как отладка загрузчика, стабилизация ОС и драйверов устройств. Это может оказать негативное влияние на процесс разработки устройств, изначально ориентированный на использование отладочных агентов.

Большинство устройств оснащены портами JTAG для отладки и тестирования. Однако во многих из них (особенно в миниатюрных, недорогих и с ограниченной памятью) не реализован порт связи, необходимый для отладки при помощи агента. В ряде других случаев использование отладочных агентов недоступно из-за жёстких требований к пространству памяти, соображений безопасности или ввиду отсутствия стека TCP/IP. В таких средах JTAG-решения просто незаменимы.

Решение JTAG также обеспечивает детальное представление обо всей системе, которое отладочные агенты предоставить не могут. Учитывая двухуровневую структуру Linux (режим ядра и режим пользователя), получение информации о приложении, ОС и оборудовании одновременно с отладкой при помощи агентов может вызвать серьёзные затруднения. JTAG-решения могут дать информацию во всех этих областях одновременно, упрощая и ускоряя процесс отладки.

Основанная на широко распространённой платформе Eclipse (см. рис. 5), среда разработки Wind River Workbench для внутрисхемной отладки (т.н. Workbench OCD) поддерживает отладочные функции JTAG как для дистрибутива Wind River Linux, так и для дистрибутивов, предоставляемых изготовителями полупроводниковых устройств, а так-

же «классической» ОС Linux, доступной на интернет-странице kernel.org. При этом обеспечивается поддержка широчайшего спектра устройств, включая новейшие многоядерные процессоры.

Отладочный инструментарий для полного цикла разработки

Среда Wind River Workbench OCD позволяет разработчикам систем на базе Linux использовать JTAG-соединение в процессе ввода оборудования в действие, отладки ядра Linux, связующего ПО и пользовательских приложений, переключаясь на использование отладочных агентов там, где это необходимо (см. рис. 6), и всё это в пределах одной и той же интегрированной среды. Такие возможности сокращают затраты времени на поиск и исправление ошибок и повышают эффективность взаимодействия различных групп разработчиков.

Отладочное решение Wind River отвечает нуждам постоянно усложняющихся систем. Организация удалённого доступа к целевой системе повышает эффективность работы распределённых групп разработчиков. Удобный конфигуратор подключений позволяет соединяться с несколькими целевыми объектами (вычислительными ядрами, процессорами, процессами) одновременно, обеспечивая, таким образом, отладку всей системы из единого интерфейса. Внутрисхемная отладка Wind River поддерживает несколько типов подключения, включая средства JTAG и агентов, и обеспечивает максимальную гибкость в отладке систем Linux – от начального загрузчика, стабилизации ядра и его модулей до разработки приложения.

Заключение

По мере роста сложности встраиваемых приложений отладка становится всё более трудной для разработчиков. Традиционные решения на основе отладочных агентов теряют эффективность в современных сложных средах. Отладочные решения на базе JTAG, интегрированные в открытую инструментальную среду на основе Eclipse, адаптированную для Linux, способны внести весомый вклад в разработку систем на базе Linux и существенно оптимизировать цикл «редактирование – компиляция – отладка».

Решения Wind River расширяют возможности использования JTAG, предлагая мощную альтернативу там, где отладочные агенты являются недоступными или слишком затратными. Уникальные функции внутрисхемной отладки, интегрированные с решениями на основе отладочных агентов, повышают эффективность даже самых сложных систем, позволяя компаниям сократить сроки разработки и снизить издержки, что в современных конкурентных условиях является одним из основополагающих факторов успеха.

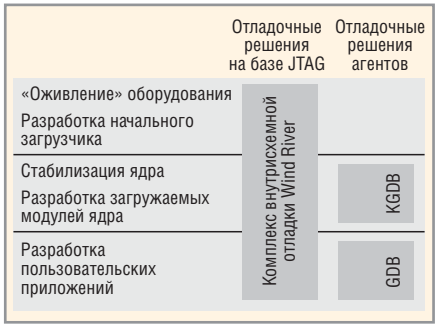

Рис. 6. Покрытие цикла разработки систем Linux отладочными решениями

Литература

1. VDC, Linux in the Embedded Systems Market. The Embedded Software Market Intelligence Program, November 2007.