Обзор средств разработки программного обеспечения мультипроцессорных комплексов

Михаил Долинский,
Игорь Ермолаев

Введение

В публикациях [1–7] довольно детально представлена разработанная в СНИЛ «Новые информационные технологии» (Гомель, Белоруссия) система сквозной совместной разработки программного и аппаратного обеспечения IEESD являющаяся продуктом интеграции нескольких компонентов, основные из которых: среда редактирования, симуляции и отладки программного обеспечения мультипроцессорных систем WInter [2] и среда редактирования, симуляции и отладки аппаратного обеспечения HLCCAD [3].

И до начала, и в ходе разработки IEESD, мы пытались отслеживать всевозможные аналогичные продукты, анализируя, куда и как развивать дальше наш собственный продукт. Частично результаты этого мониторинга представлены в наших «Тенденциях…» [8]. Данная статья представляет систематизированный обзор программных продуктов, предназначенных для решения сходных задач. Отметим сразу, что продукты, декларирующие поддержку симуляции и отладки программного обеспечения мультипроцессорных систем, не имеют доступных с сайта производителя оценочных или демонстрационных версий, что не дает возможности провести их полноценное сравнение. Поэтому по каждому из таких продуктов представлены лишь систематизированные комплексы свойств, наличие которых удалось обнаружить при переработке рекламной и технической информации, представленной на сайтах соответствующих фирм. Поскольку, как правило, такие продукты состоят из двух независимых компонентов «симуляция/отладка» и «редактирование/ведение проектов», то и обзоры выполнены для этих компонентов по отдельности. Кроме того, поскольку на практике при создании мультипроцессорных систем сегодня практически все отечественные разработчики и значительная часть зарубежных пользуются средствами разработки программного обеспечения для однопроцессорных систем, то сравнение с ними приведено в третьем пункте. Заметим, что один вариант такого сравнения (по состоянию на май 2000 года) нами уже приводился [2]. Нынешнее сравнение проведено в мае 2003 года. В разделе 4 приводятся выводы, полученные в результате сравнения.

1. Средства симуляции и отладки мультипроцессорных систем

1.1. CrossView Pro фирмы TASKING

Основными характеристиками CrossView Pro, заявленными на сайте производителя (**www.tasking.com**), являются:

- возможность открывать больше чем одно окно с данными, дампом памяти и регистрами;
- смешанный режим отображения исходного текста и дизассемблированного;
- текущая исполняемая строка С-программы выделяется в смешанном режиме;
- значения переменных и функций во всплывающих подсказках;
- точки останова на исполнение, на доступ к данным, смешанные и при использовании эмуляторов;
- автоматическое пошаговое исполнение до невыполнения условия;
- множество сред исполнения: симулятор, монитор памяти, BDM/JTAG/OCDS, внутрисхемный эмулятор;
- имитация файловой системы;
- прямой доступ к памяти и регистрам;
- группирование регистров;
- пошаговое исполнение программ на языках С и ассемблера;
- анализ скорости исполнения программ:
  - программируемый анализ сигнальных данных;
  - процент выполненного кода и замер времени, потраченного на каждый логический блок;
  - подсчет циклов.
- запись и воспроизведение отладочных сессий;
- вычисление выражений C/C++ с использованием мощного макроязыка;
- расширенная и простая в использовании отладка RTOS.

Окно с исходным текстом контролирует выполнение программы. Оно позволяет видеть текст программы, устанавливать и сбрасывать точки останова, просматривать значения переменных, искать строки, функции, адреса, вызовы функций и вычислять выражения.

Окно с исходным текстом позволяет рассматривать программу как С-текст и как дизассемблиро-

ванный текст. Из этого окна можно переключиться прямо в редактор EDE, в ту же строку исходного текста, что и в отладчике. Это помогает сразу перейти к исправлению той строки исходного текста, в которой была обнаружена ошибка. Позиционируя мышь над переменной или функцией в исходном окне, можно увидеть их значения во всплывающем окне.

Окна с данными предназначены для слежения за значениями переменных (локальных или глобальных) и изменения их значений. Внутри этого окна можно изменить систему счисления отображаемых значений и порядок отображаемых переменных.

Окно с регистрами отображает и предоставляет средства редактирования регистров процессора. Окно полностью настраиваемое и обновляется каждый раз, когда выполнение программы останавливается. Регистры, в которых изменились значения, подсвечиваются. Также есть возможность группировать регистры, например, регистры общего назначения, регистры с плавающей точкой, UART и т. д.

Окно с дампом памяти, отображающее значения в шестнадцатеричном виде и виде текста, позволяет просматривать и модифицировать значение любой ячейки памяти. Изменившиеся значения с момента предпоследней остановки выполнения программы выделяются другим цветом.

Имитация файловой системы позволяет разрабатываемой программе использовать стандартные функции ввода-вывода, такие, как fopen() и fprintf(). Файлы читаются средствами операционной системы, на которой запущен отладчик, пишутся так же средствами ОС или выводятся в специальное окно отладчика.

Имитация ввода-вывода позволяет наблюдать принимаемые и передаваемые значения до обмена с аппаратным обеспечением. С помощью двух специальных функций ввод-вывод можно автоматически переназначить в отладчик, а также можно отладить код, зависящий от обмена с аппаратным обеспечением, даже до получения необходимого «железа».

1.2. MULTI Source-Level Debugger фирмы Green Hills

По информации, полученной с сайта производителя (**www.ghs.com**), MULTI Source-Level Debugger (в дальнейшем SLD) является мощным оконным отладчиком, который позволяет осуществлять:

- загрузку программы;
- выполнение программы;
- контроль исполнения.

SLD обеспечивает отладку программ, написанных на C, C++, Embedded C++, Аде 95, Фортране и ассемблере. Основные функции SLD интуитивны и просты в использовании; программисты тратят меньше времени на изучение средств и больше времени на проектирование программы. SLD включает передовые средства для просмотра данных, проверку ошибок во время исполнения и профилировщик.

Главное окно отладчика состоит из трех частей. Верхняя часть содержит навигационные кнопки. Ниже находится окно с исходным текстом. Нижняя часть является окном команд, которое используется для ввода текстовых команд и просмотра результатов их выполнения.

SLD обеспечивает полный набор отладочных средств, разработанных для облегчения процесса нахождения и исправления ошибок:

- установка точек останова — одиночный щелчок мышью на зеленой точке, расположенной перед каждой строкой исходного или дизассемблированного текста, позволяет установить точку останова. Специальный значок заменит зеленую точку, что бы показать, что точка останова успешно установлена. Повторный щелчок по значку сбросит точку останова;
- условные и командные точки останова позволяют определить условие останова и действие при выполнении строки исходного текста;
- выполнение с заходом в подпрограммы, выполнение без захода и выполнение до выхода из подпрограммы;
- просмотр переменных осуществляется с помощью щелчка мышью по имени переменной в окне с исходным текстом, значение переменной отображается в окне команд;
- окно с переменной отображается по двойному щелчку мышью по имени переменной, в окне отображается имя, тип и текущее значение переменной. Переменная может быть любого типа: целое, структура, массив или объект. Окно с переменной может быть «заморожено» для сравнения с последующими значениями переменной;
- окно с локальными переменными позволяет просматривать значения переменных, объявленных в текущей функции, при перемещении курсора от функции к функции содержимое окна меняется;
- окно со стеком отображает последовательность вызовов функций;
- двойной щелчок мышью по имени функции приводит к отображению в окне с исходным текстом тела этой функции.

SLD обеспечивает просмотр дампа памяти в шестнадцатеричном, десятичном, текстовом, двоичном видах.

SLD тесно взаимодействует с другими компонентами комплекса. Щелкая на кнопке «Редактировать» можно запустить редактор для изменения текущего файла с позиционированием курсора в текущую строку. После окончания редактирования единственным щелчком мыши можно перекомпилировать программу и перезапустить отладчик.

SLD поддерживает смешанную многоязыковую отладку для C, C++, Embedded C++, Ады 95, Фортрана и ассемблера. При перемещении среди модулей с различными языками SLD распознает язык текущего файла и настраивает вычисление выражений и отображение данных соответствующим образом.

1.3. RealView Debugger фирмы ARM

Как написано на сайте производителя (**www.arm.com**), RealView Debugger является центральным компонентом нового программного комплекса RealView фирмы ARM. Он осуществляет отладку смешанных мультипроцессорных архитектур, основанных на процессоре ARM, системах-на-кристалле (SoC).

Основные особенности отладчика:

- простота использования:
  - графический оконный интерфейс предназначен для одно- и мультипроцессорной отладки;
  - контекстозависимые меню;
  - языковозависимые меню;
  - действия пользователя генерируют текстовые команды, которые могут быть записаны и воспроизведены заново;
  - возможно редактирование исходного текста в окне отладки.
- полная поддержка точек останова:
  - сложные точки останова (например, счетчики, логические конструкции);
  - история точек останова.
- полная поддержка регистров (просмотр в разных форматах).
- просмотр переменных, функций, модулей и файлов.

RealView Debugger подключается к эмулятору ARM Multi-ICE JTAG и эмулятору RealView ICE для отладки реального «железа», и к ARM Development Suite для симуляции выполнения по инструкциям.

RealView Debugger обеспечивает мультипроцессорную отладку процессоров ARM плюс DSP-процессоров, используя единственную отладочную среду. Для каждого процессора можно независимо выполнять пошаговую отладку и устанавливать или сбрасывать точки останова. Мультипроцессорный синхронный старт, остановка и выполнение по шагам доступны для исследования и отладки взаимодействия между приложениями, запущенными на различных процессорах системы.

Возможность останова процессоров поддерживается как в программном, так и аппаратном обеспечении. Это облегчает синхронную остановку процессоров, когда выполнение достигает точки останова в одном из процессоров, что позволяет предотвратить обработку (потерю) данных в процессе исследования состояния процессоров.

RealView Debugger обеспечивает текстовый эквивалент для каждого действия пользователя, произведенного с помощью графического интерфейса (GUI). Такие команды могут вводиться непосредственно в специальном окне, в этом же окне они автоматически отображаются, когда используется GUI. Это возможность чрезвычайно полезна при работе в пакетном режиме.

1.4. SeeCode Debugger фирмы MetaWare

По информации, опубликованной на сайте производителя (**www.metaware.com**), SeeCode Debugger предоставляет набор традиционных отладочных средств, включающих в себя отладку по исходным и дизассемблированным текстам, условные точки останова, просмотр и изменение переменных, регистров и памяти.

Скоординированная мультипроцессорная отладка (CMPD) для SeeCode Debugger позволяет отлаживать мультипроцессорные

приложения, использующие до 256 процессоров, в одной отладочной сессии.

SeeCode Debugger позволяет понять внутреннюю динамику системы, отображая информацию о том, что менялось и как менялось.

Основные достоинства системы:

- поддержка C и C++;
- прозрачная поддержка эмулятора и симулятора;
- история выполнения инструкций;
- смешанная отладка по исходному и дизассемблированному тексту;
- гибкое отображение периферии и регистров;
- настраиваемость на новые процессоры и операционные системы;
- поддержка форматов файлов ELF и DWARF.

Дополнительные возможности:

- изменение размера шрифта для каждого окна;
- больше чем одно окно с регистрами, дампом памяти, дизассемблером, исходным текстом и т. д.;
- раздельное множество окон для каждого отлаживаемого процесса;
- опционально множество окон для каждого отлаживаемого потока;
- расширяемый GUI (путем написания Java-классов);
- возможность «заморозить» любое окно для сравнения значений до и после;
- вычисление значений функций;
- поиск значений в дампе памяти.

1.5. XRAY Debugger фирмы Mentor Graphics

Как заявлено на сайте производителя (**www.mentor.com**), XRAY Debugger предоставляет мощные средства отладки современных 32- и 64-разрядных процессоров. Разработчик управляет процессом отладки с помощью текстовых команд или GUI. Эти операции могут быть сохранены в файле и воспроизведены в дальнейшем. Мощный интерфейс пользователя позволяет осуществлять:

- управление простыми и сложными точками останова;
- пошаговое исполнение;
- изменение кода программы;
- непрерывное наблюдение за переменными.

При достижении указанной точки останова или выполнении одного шага программы могут быть осуществлены заранее определенные действия. Например, точка останова может быть задана с условием «when(x==5)». При достижении этой точки процессор будет остановлен только в том случае, когда выполнится условие. Аналогично, каждой выполняемой инструкции может соответствовать команда, которая записывает в файл исполняемую строку программы или инструкцию для постоянной записи процесса хода выполнения программы.

Центральным окном XRAY Debugger является Code Window, которое представляет приложение в виде исходного текста на языке высокого уровня (ЯВУ) или в виде дизассемблированного текста. В режиме отображения исходного текста ключевые слова ЯВУ, комментарии, строки и константы выделятся разными цветами для улучшения читабельности. В режиме отображения дизассемблированного текста исходный текст на ЯВУ отображается вместе с дизассемблированным.

Code Window также служит редактором, который с легкостью может быть интегрирован с коммерческой системой контроля версий.

Для отладки мультипроцессорных приложений XRAY Debugger позволяет открыть несколько окон Code Window одновременно.

Code Window дополняется полным набором вспомогательных окон, которые специально разработаны для облегчения выполнения сложных отладочных задач.

Многие SoC используют расширенные наборы регистров, отображаемые на память, для поддержки памяти, расположенной на чипе, и других периферийных устройств. Работая с Mentor Graphics Embedded Software Knowledge Center, можно настроить окно с регистрами таким образом, чтобы эти наборы регистров казались частью ядра.

Inspector Window — мощный инструмент исследования данных. Переменные любого типа и в любых количествах могут быть просмотрены и изменены в этом окне. Переменные и структуры данных могут даже приводиться к другому типу с использованием того же синтаксиса, что и в исходном тексте.

Traceback Window отображает вложенность вызовов подпрограмм. Каждая подпрограмма пронумерована начиная с самого нижнего уровня (0). Это окно позволяет просмотреть параметры и тип возвращаемого результата для каждой подпрограммы, а также просмотреть значения и типы локальных переменных.

XRAY Debugger предоставляет макроязык, похожий на C. Макросы ведут себя как C функции. Они могут:

- вызывать и быть вызванными из других макросов;
- передавать и принимать параметры;
- содержать локальные переменные;
- возвращать значение.

Макросы также могут содержать команды XRAY Debugger и использовать специальные переменные (системные или определенные пользователем). Они даже могут быть использованы для расширения команд отладчика и присоединены к кнопкам, определенным пользователем в CodeWindow.

XRAY Debugger поддерживает три типа командных файлов: include, log и journal. Include-файлы в основном являются скриптами, содержащими команды XRAY Debugger. Include-файлы позволяют задавать:

- новые переменные;
- настройки окон с регистрами и дампом памяти;
- настройки пошагового выполнения и точек останова.

Include-файлы особенно полезны для быстрой настройки отладочного окружения. Например, для выбора чипа, загрузки приложения и добавления основных переменных в Inspector Window.

В log-файлы записываются все команды отладчика, которые были введены в течение отладочной сессии, позволяя автоматически создавать include-файлы.

В journal-файлы записываются выполненные команды и результаты их выполнения.

Комбинация командных файлов и макросов предоставляет мощное решение для автоматической отладки и регрессионного тестирования. Include-файлы могут быть использованы для автоматизации отладочной сессии и сравнения результатов с journal-файлами.

Так как эти командные файлы могут быть использованы при любых типах выполнения, то разработчики могут использовать их регрессионные тесты на протяжении всего жизненного цикла программы, начиная с раннего проектирования и заканчивая поддержкой выпущенного продукта.

В XRAY Debugger точки останова могут быть простыми или сложными. Все точки останова могут быть временно запрещены в Code Window или Breakpoint Manager, а позже вновь разрешены.

Сложные точки останова включают «pass-count». Выполнение программы будет остановлено только в том случае, если количество выполнений строки с точкой останова достигло указанного значения. К точке останова может быть присоединен макрос для выполнения любого количества команд при срабатывании точки останова. Значение, возвращаемое макросом, используется для определения необходимости остановки выполнения программы.

XRAY Debugger предоставляет следующие режимы отладки:

- XRAY Simulator,
- Seamless Co-verification,
- On-Chip Debugging,
- Software Monitor.

XRAY Simulator позволяет моделировать исполнение программы по инструкциям и может быть применен до получения или до завершения проектирования аппаратной части.

XRAY Debugger в комбинации с Seamless Co-verification позволяет проверить аппаратно-программный интерфейс до производства чипа, уменьшая и исключая возникновение ошибок.

XRAY On-Chip Debugging, используя решения сторонних производителей, включающие в себя Agilent Emulation Probe, Embedded Performance MAJIC+, Macraigor Systems Raven probe и ARM Multi-ICE, предоставляет простое в использовании решение сложных проблем интеграции программного и аппаратного обеспечения.

XRAY Software Monitor в паре с RTOS-совместимым программным монитором предоставляет уникальные отладочные возможности в виде:

- точек останова специфичных для задач;
- отладки в режиме выполнения, когда некоторые задачи могут быть остановлены, в то время как остальные продолжают выполняться;
- исчерпывающего отображения и контроля системных потоков и ресурсов.

2. Средства ведения проектов программного обеспечения мультипроцессорных комплексов

2.1. Embedded Development Environment фирмы TASKING

Из описания, взятого с сайта производителя (www.tasking.com), EDE — не просто редактор. Это полноценная среда управления проектами, которая предоставляет прямой доступ к инструментальным средствам и операциям, позволяющий повысить эффективность работы программиста. К этим основным возможностям EDE можно отнести:

- «кнопки» для выполнения многочисленных задач с использованием различных программных средств;
- плотную интеграцию инструментальных средств, позволяющую ускорить процесс редактирования-компилирования-отладки, что ведет к повышению продуктивности путем автоматизации часто повторяющихся действий;
- настройку параметров с помощью GUI;
- автоматизированный процесс «сборки» приложения;
- встроенную систему помощи.

EDE позволяет указать файлы, входящие в проект, и настроить, переключаясь между закладками, инструментальные средства для компиляции. С помощью менеджера проекта можно легко создавать и редактировать проект. В специальном файле сохраняются следующие параметры проекта:

- список исходных фалов;
- настройки инструментальных средств:
  - компилятора;
  - ассемблера;
  - компоновщика;
  - отладчика.
- пути к инструментальным средствам;
- настройки, описывающие процесс «сборки».

Менеджер проекта отслеживает зависимость исходных файлов проекта и определяет последовательность операций, «необходимых» для сборки приложения. EDE анализирует сообщения об ошибках, сгенерированные компилятором и ассемблером, и показывает их местоположение в исходном тексте, что позволяет ускорить процесс исправления ошибок.

EDE поднимает простоту использования на новые высоты, позволяя полностью настроить среду разработки для своих нужд. В процессе настройки можно создавать свои меню, добавлять необходимые возможности, задавать свои комбинации «горячих клавиш» и настраивать панель инструментов.

В процессе разработки необходимость координировать несколько различных проектов приводит к ощутимым затратам времени. EDE Project Spaces позволяет минимизировать эти затраты путем предоставления доступа к любому файлу проекта, в любое время, где бы он ни находился.

EDE Code Sense позволяет при наборе текста выбирать из списка поля структур и видеть параметры функций. Позиционируя указатель мыши над именем функции, можно увидеть ее прототип. Это делает почти невозможным неправильное использование параметров функций и полей структур, что приводит к сокращению времени, потраченного на отладку и изучение документации.

Используя EDE Tag Browser, можно получить графическое изображение всех важных перекрестных ссылок для приложения. Это позволяет легко перемещаться к доступным переменным и функциям. При использовании быстрого доступа к функциям и классам, а также к их параметрам и свойствам максимизируется эффективность кодирования и увеличивается продуктивность работы.

EDE CodeFolio повышает эффективность и согласованность процесса кодирования, предоставляя возможность использования шаблонов и макросов. Когда используется шаблон, EDE CodeFolio раскрывает макросы и запрашивает необходимую информацию для генерации текста. Макросы могут выполняться в указанное время, перемещать курсор в заданную позицию, выполнять математические действия и т. д.

2.2. MULTI Integrated Development Environment фирмы Green Hills

На сайте производителя (www.ghs.com) указано, что MULTI Integrated Development Environment (в дальнейшем IDE) — полноценная среда разработки для встроенных систем, поддерживающая C, C++, Embedded C++, Аду 95 и Фортран. IDE предоставляет прямой графический интерфейс для всех компиляторов фирмы Green Hills и поддерживает многоязыковую разработку. IDE состоит из следующих компонентов:

- построитель проектов;
- GraphicalBrowser;
- текстовый редактор;
- подсистема контроля версий.

Построитель проектов предоставляет интуитивный графический интерфейс для настройки и создания сложных программных проектов. GUI позволяет выбрать целевой процессор, стратегию оптимизации кода, уровень детализации отладочной информации, настроить языковозависимые параметры и указать режим контроля ошибок во время исполнения. Проект отображается в виде «дерева» и состоит из файлов с исходными текстами, объектных файлов, библиотек и подпроектов. Отдельно поддерживаемые библиотеки и подпроекты могут разрабатываться как независимые модули и включаться в различные проекты. Из построителя проектов пользователь может запустить компилятор, отладчик, редактор и т. д.

Мастер проектов задает пользователю серию вопросов и создает пустой проект или шаблон как первое приближение разрабатываемого приложения.

Graphical Browser позволяет пользователю видеть как всю структуру программы, так и фокусироваться на конкретном объекте или функции. Многие элементы графического представления интерактивны, например, щелкнув на имени класса, можно увидеть окно со списком свойств класса.

IDE включает полнофункциональный настраиваемый пользователем текстовый редактор, поддерживающий подсветку синтаксиса и автоотступ. Подсветка синтаксиса позволяет программисту быстро распознавать ключевые слова и конструкции языка, включающие комментарии, строки и константы. Автоотступ автоматизирует форматирование текста программы в процессе ввода. Объединение этих особенностей редактора усиливает читабельность текста программы и облегчает написание кода.

Текстовый редактор поддерживает все стандартные функции, он полностью настраиваемый: пользователь может изменить комбинации клавиш, расположение и поведение «полос прокрутки», поведение мыши и т. д.

Подсистема контроля версий полностью интегрирована в IDE, это позволяет сделать процесс управления версиями проекта эффективным и ненавязчивым. Все наиболее часто используемые действия автоматизированы. Подсистема осуществляет контроль версий исходных файлов приложения, сохраняет историю изменений и показывает различия в графическом виде.

3. Средства разработки программного обеспечения однопроцессорных систем

В сравнении участвовали следующие продукты (демоверсии которых нам удалось получить в мае 2003 года):

- AVR Studio фирмы Atmel;
- IAR Embedded Workbench фирмы IAR Systems;
- mVision2 фирмы Keil Software.

Главным недостатком этих средств является их однопроцессорная ориентация. Конечно, с помощью этих продуктов можно разрабатывать ПО для каждого процессора из мультипроцессорной системы по отдельности. Но, во-первых, придется постоянно переключаться между проектами или загружать несколько копий интегрированных сред (по одной на каждый проект). А во-вторых, не возможно будет провести комплексное моделирование и отладку всей мультипроцессорной системы целиком. Придется моделировать каждый процессор по отдельности и придумывать разные ухищрения для симуляции внешних воздействий других процессоров.

В таблице 1 приведены результаты сравнения средств, ориентированных на один процессор. В сравнении также участвовала система WInter, разработанная в СНИЛ «Новые информационные технологии» Гомельского государственного университета. Данный продукт может использоваться для проектирования ПО как для однопроцессорных, так и для мультипроцессорных систем.

Сравнение проводилось по пяти составным критериям. Расшифровки составных критериев приведены в таблицах 2–6. Для получения значения каждого из составных критериев использовалась следующая методика. Подбирался набор признаков, характеризующих составной критерий. Затем проверяли каждый из вышеперечисленных продуктов. Если продукт удовлетворял признаку, то ему добавлялся балл к соответствующему состав-

ному критерию. В результате сумма всех баллов и являлась численной оценкой составного критерия.

Таблица 1. Сравнение средств разработки ПО, ориентированных на один процессор

Составной критерий	WInter	AVR Studio	IAR Embedded Workbench + C-SPY	mVision2
Интерфейс	7	3	3	4
Редактор	7	3	4	4
Анализ процессора	17	7	8	10
Команды выполнения	8	4	5	5
Отладчик	8	1	3	3
Показатель качества	47	18	23	26

Таблица 2. Расшифровка составного критерия «интерфейс»

Признак	Winter	AVR Studio	IAR	mVision2
Запоминание текущего проекта	•		•	•
Возможность загрузить любое из ранее сохраненных состояний	•	•	•	•
Сохранение и восстановление расположения окон, панелей инструментов и т. п.	•	•	•	•
Механизм избавления от перекрытий окон	•	•		•
Автоматическая установка размеров окна в зависимости от его содержимого	•			
Полноэкранный режим (видно только одно окно редактора)	•			
Выбор языка интерфейса	•			
Итого	7	3	3	4

Таблица 3. Расшифровка составного критерия «редактор»

Признак	Winter	AVR Studio	IAR	mVision2
Подсветка синтаксиса языка	•	•	•	•
Возможность настроить цветовую схему	•	•	•	•
Для каждого языка своя схема подсветки	•			•
Автоотступ при редактировании	•	•	•	•
Табуляция по столбцам	•			
Шаблоны (code insight)	•			
Макросы	•		•	
Итого	7	3	4	4

Таблица 4. Расшифровка составного критерия «анализ процессора»

Признак	Winter	AVR Studio	IAR	mVision2
Окно с дампом памяти	•	•	•	•
Окно с регистрами	•	•	•	•
Окно с флагами	•	•		•
Окно с битами	•			
Окно со стеком	•	•	•	•
Окно с дизассемблером	•		•	•
Окно с переменными и выражениями	•	•	•	•
Окно с контактами	•			
Окно со статистикой	•		•	
Подсветка изменившихся значений	•		•	•
Отображение адреса ячейки в дампе памяти, на которую указывает мышь	•			
Изменение количества колонок в дампе	•	•		•
Выбор отображаемых регистров, флагов, битов	•			
Изменение порядка отображения регистров, флагов, битов	•			
Показ значения переменной, регистра, флага, бита в окне подсказки	•	•	•	•
Показ значения выражения в окне подсказки	•			•
Итого	17	7	8	10

Таблица 5. Расшифровка составного критерия «команды выполнения»

Признак	Winter	AVR Studio	IAR	mVision2
Выполнить инструкцию	•			
Выполнить шаг	•	•	•	•
Выполнить без захода в подпрограмму	•	•	•	•
Выполнить до выхода из подпрограммы	•	•	•	•
Выполнить до курсора	•		•	•
Анимация по инструкциям	•	•	•	•
Анимация по шагам	•			
Установить указатель	•			
Итого	8	4	5	5

Таблица 6. Расшифровка составного критерия «отладчик»

Признак	Winter	AVR Studio	IAR	mVision2
Простые точки останова	•	•	•	•
Условные точки останова	•		•	•
«Теневые» команды	•		•	•
Моделирование терминала	•			
Моделирование внешней периферии	•			
Возможность разработки и добавления собственных моделей периферийных устройств	•			
Возможность разработки и добавления собственных моделей процессоров	•			
Автоматическое тестирование	•			
Итого	8	1	3	3

4. Анализ результатов сравнения

На основе проведенного обзора средств моделирования мультипроцессорных систем и разработки программного обеспечения мультипроцессорных комплексов, а также средств, ориентированных на один процессор, можно сделать следующие выводы:

- большинство средств не предоставляет возможности моделировать мультипроцессорные системы совместно с аппаратным окружением и внешней средой;
- в некоторых системах существует ограничение на количество одновременно моделируемых процессоров (например, SeeCode Debugger фирмы MetaWare);
- в большинстве систем редактор и отладчик — это два отдельных приложения, что приводит к необходимости постоянно переключаться между ними;
- в большинстве систем на каждый моделируемый процессор открывается отдельное окно (с исходным текстом, окнами с данными и т. д.), при моделировании процессоров числом больше 10 это приведет к существенному загромождению экрана и может сильно усложнить процесс отладки;
- почти все средства ориентированы на конкретный процессор (или семейство) и не могут быть настроены для использования с другими;
- отсутствует возможность интеграции произвольных внешних трансляторов и компоновщиков;
- отсутствует возможность моделировать сложные внешние воздействия и использовать подключаемые библиотеки с моделями периферийных устройств;
- работать с реальными (не моделируемыми) периферийными устройствами можно только с использованием внутрисхемного эмулятора (нельзя использовать модель процессора и реальное «железо»);
- отсутствует возможность генерировать модели процессоров на основе описания ресурсов и алгоритмов поведения;
- предлагается минимальный набор средств отладки программного обеспечения;
- почти во всех приложениях отсутствует возможность проводить автоматическое тестирование и верификацию разрабатываемого программного обеспечения.

Отсутствие адекватных средств совместного моделирования программного и аппаратного обеспечения мультипроцессорных систем и внешней среды, а также ориентированность на конкретное семейство процессоров существенно увеличивает срок и стоимость разработки программного обеспечения встроенных мультипроцессорных систем. Отсутствие средств автоматического тестирования и верификации делает невозможным сквозной контроль качества разработки программного обеспечения вычислительных систем.

Поэтому актуальной является выполненная в СНИЛ «Новые информационные технологии» разработка WINTER/IEESD, включающая:

- единую универсальную интегрированную среду, настраиваемую на используемые процессоры;
- открытые моделируемые компоненты с универсальным базовым интерфейсом, выступающие как модели процессоров и периферийных устройств;
- язык описания процессора (ядра и внутренней периферии);
- средства генерации моделей процессоров по описанию ядра и внутренней периферии;
- средства получения моделей процессоров по описанию ресурсов и алгоритмов поведения на языках программирования высокого уровня;
- производительную симуляцию мультипроцессорных систем;
- средства моделирования сложных внешних воздействий;
- средства автоматизации тестирования и верификации.

Заключение

В связи с тенденциями микроминиатюризации и удешевления электронных изделий, мультимикропроцессорные системы становятся вполне эффективными решениями многих прикладных проблем не только в военной отрасли, но и в автомобилестроении, сетевой обработке, телекоммуникации, потребительских мультимедийных электронных устройствах и т. д. В тоже время дальнейший рост применения мультипроцессорных систем в значительной степени сдерживается недоступностью широкому кругу разработчиков средств сквозной совместной разработки программного и аппаратного обеспечения и внешней среды, настраиваемых на требуемую систему микропроцессоров.

Литература

1. Долинский М. Концептуальные основы и компонентный состав IEESD-2000 — интегрированной среды сквозной совместной разработки аппаратного и программного обеспечения встроенных цифровых систем // Компоненты и Технологии. 2002. № 8.
2. Долинский М., Ермолаев И., Толкачев А., Гончаренко И. WInter — среда отладки программного обеспечения мультипроцессорных систем // Компоненты и Технологии. 2003. № 2.
3. Долинский М., Литвинов В., Галатин А., Ермолаев И. HLCCAD — среда редактирования, симуляции и отладки аппаратного обеспечения // Компоненты и Технологии. 2003. № 1.
3. Долинский М., Литвинов В., Толкачев А., Корнейчук А. Система высокоуровневого проектирования аппаратного обеспечения HLCCAD: тестирование // Компоненты и Технологии. 2003. № 3.
4. Долинский М., Литвинов В., Галатин А., Шалаханова Н. Система высокоуровневого проектирования аппаратного обеспечения HLCCAD: открытый универсальный интерфейс моделируемых компонентов // Компоненты и Технологии. 2003. № 4.
5. Долинский М., Литвинов В., Ермолаев И, Федорцов А. Система высокоуровневого проектирования аппаратного обеспечения HLCCAD: технология разработки потактовой модели микроконтроллера с использованием языка программирования высокого уровня на примере Intel 8051/Object Pascal // Компоненты и Технологии. 2003. № 5.
6. Долинский М., Ермолаев И., Федорцов А., Литвинов В., Толкачев А., Гончаренко И., Коршунов И. Технология разработки моделей микроконтроллеров с использованием декларативно-алгоритмических языков описания ядра и периферии // Компоненты и Технологии. 2003. № 7. 2003.
7. http://NewIT.gsu.unibel.by
8. Долинский М. Тенденции и перспективы развития EDA-индустрии по материалам портала DACafe.com // Компоненты и Технологии. 2002. № 8–9. 2003. № 1–6.