УДК 004.274

А.В. Палагин, М.В. Семотюк, Е.Н. Чичирин, Е.П. Сосненко

Моделирующая среда для создания и отладки систем цифровой обработки

Рассмотрена архитектура операционной среды для отладки программного кода и верификации цифровых проектов. Изучены вопросы выбора графической модели и мониторинга функционирования проектируемой системы.

The architecture of the operating environment is considered for debugging the program code and verification of digital projects. The questions of the selection of a graphic model and monitoring of the functioning of the projected system are investigated.

Розглянуто архітектуру операційного середовища для налаштування програмного коду і верифікації цифрових проектів. Досліджено питання вибору графічної моделі та моніторингу функціонування системи, що проектується.

Введение. В системе современных предприятий подразделения, занимающиеся процессами создания и выпуска управляющих подсистем низкого уровня, сталкиваются с задачами, характерными для всех отраслей, связанных с *hi-tech* технологиями. К ним относятся сокращение цикла разработки и отладки новых средств цифровой обработки в комплексе с сопутствующими программами и приложениями. Уже на стадии разработки и отладки программной модели новых сигнальных процессоров или микроконтроллеров (МПК), обычно предшествующей появлению отлаженных микросхем, необходимо учитывать эффективность их использования для выполнения современных управляющих и телекоммуникационных алгоритмов, алгоритмов аудиовизуальной и интеллектуальной обработки информации. Для этого системы автоматизированного проектирования (САПР) должны быть дополнены системами разработки и отладки библиотек и других первоочередных программных модулей и приложений для новых изделий, а также средствами их верификации совместно с аппаратными образцами МПК в условиях, максимально приближенных к рабочим [1].

Обеспечить такие условия призваны соответствующие интерфейсы в составе интегрированной среды разработки и отладки программного обеспечения МПК, дополненные эффективными средствами его программной симуляции (эмуляции).

Постановка задачи

Моделирование ситуаций, связанных с поиском неисправностей в опытных образцах цифровых устройств и ошибок в бета-версиях программного обеспечения, подтверждает необходимость наличия в составе интегрированной моделирующей операционной среды (ИМОС) средств низкоуровневой и высокоуровневой отладки, совмещенных с программной моделью МПК и редактором программного кода. При этом низкоуровневый отладочный интерфейс должен обеспечивать мониторинг обмена данными между ИМОС и МПК в ручном и автоматическом режиме. Его назначение – проведение профилактических работ и первичное взаимодействие с новыми неработоспособными образцами. Основная задача интерфейса высокого уровня – синхронизация и визуальное представление входного и выходного потока данных МПК и(или) его программной модели, а также результатов их промежуточной обработки в виде, наиболее адекватно воспринимаемом человеком. Полученные решения должны соответствовать принципам реализации человеко-машинного интерфейса – *Human-Machine Interface* (*HMI*) и обеспечивать эффективность комплексной отладки приложений для новых производительных МПК.

Ввод в состав ИМОС программного симулятора обусловлен необходимостью разработки и отладки базового программного обеспечения МПК до получения его работоспособ-

ных образцов. При этом подсистема эмуляции системы команд МПК должна иметь возможность настраиваться (реконфигурироваться) в зависимости от состава его программно-доступных регистров (АЛУ, адресно-индексных регистров, портов ввода–вывода и др.) и регистров его программно-аппаратного окружения [2].

С учетом сказанного получим перечень задач, подлежащих решению при создании ИМОС:

- разработка редактора, кросс-ассемблера, линковщика и загрузчика программного кода и данных для проектируемого МПК;
- разработка программной модели–симулятора с возможностью его реконфигурации с учетом архитектуры МПК (размера внутренней памяти, состава программно-доступных регистров) и его программно-аппаратного окружения;
- обеспечение возможности модификации (реконфигурации) моделей–симуляторов в заданном классе (семействе) МПК;
- выбор графической модели и разработка алгоритмов мониторинга состояний исходной системы и ее программной модели (виртуальной машины);
- выбор интерфейса и разработка драйверов для скоростного обмена данными ИМОС с МПК (с учетом возможного наличия в нем критичных и некритичных для такого обмена неисправностей);
- разработка библиотеки подпрограмм и функций, тестов МПК и дополнительных программных средств, включая элементы *HMI* – интерфейса;
- разработка дизассемблера – для отладки приложений в отсутствие исходных кодов.

Предмет исследования

В настоящей статье исследуется архитектура операционной среды для разработки и отладки программного кода, а также моделирования и верификации проектируемых систем цифровой обработки, рассматриваемая в контексте расширения функциональных возможностей программного симулятора проектируемой системы и алгоритмов мониторинга процессов их совместного функционирования.

Модель проектируемого МПК (*симулятор* или, с учетом динамики моделируемых процессов – *виртуальная машина*) может быть построена на основании анализа технического задания проекта. При этом форма представления задания определяет долю «ручного» труда при переводе этого документа в программную модель на языке моделирующего компьютера или *host*-компьютера. Другой путь – трансляция формализованного представления МПК с высокоуровневого языка проектирования цифровых систем типа *VHDL*. Помимо проблематичности качества трансляторов второй подход не избавляет от затрат на подготовку самого *VHDL*-описания. И, главное, из-за отсутствия альтернативного *VHDL*-описанию «понимания» технического задания при разработке симулятора, снижается достоверность верификации всего проекта путем сравнения функционирования опытного образца МПК и его программной модели [3].

Для ускорения этапов проектирования, изготовления (программирования) и отладки МПК можно использовать методы сетевого планирования, аналогичные конвейерным архитектурам самих МПК. Например, целесообразно сначала передать в ИМОС на верификацию (сравнение с программной моделью) АЛУ проектируемого МПК, а САПР в это время занять под проектирование устройства управления МПК. Для реализации данного подхода необходимы два условия:

- на любом этапе верификации МПК должен поддерживать обмен данными с ИМОС, установленной на *host*-компьютере;
- в ИМОС должны быть предусмотрены средства работы с МПК в режиме *Master – Slave* (ведущий–ведомый), когда тест МПК выполняется его симулятором, а в МПК загружаются данные и считываются реакции только для очередного верифицируемого блока.

Принципиально в данном случае то, что нелинейный и достаточно сложный в общем случае тест пишется на одном из входных языков МПК независимо от способа реализации обмена ИМОС с МПК. Существенна только минимизация затрат на поддержку такого интерфейса со стороны МПК. Это может быть вариант канала прямого доступа, технологический либо другой (в том числе удаляемый на последнем этапе проектирования) интерфейс. Из-

вестны впечатляющие примеры близкого подхода при производстве тех же МПК (разбраковка и переоценка, а не полная отбраковка кристаллов с урезанными возможностями).

Критическим параметром программных моделей всегда была производительность. Рассмотрим один из эффективных способов ее повышения.

Последовательность шагов симулятора при эмуляции очередной команды МПК практически может быть представлена в виде дерева D, т.е. ориентированного графа без циклов. Некоторые технологические циклы, осуществляемые в компиляторах с целью оптимизации размера кода, всегда могут быть преобразованы к линейному виду. Каждая ветвь V дерева D, анализирующая поля битов только текущей команды, может быть представлена последовательностью из двух шагов:

1. Выделить в поле $B1$ команды при помощи маски M битовое поле $B2$.

2. Сравнить поле $B2$ с некоторой константой C.

Если $B2 = C$, выполнить операцию $Z1 = F1(X1, Y1)$ и перейти к ветви $V1$, иначе выполнить операцию $Z2 = F2(X2, Y2)$ и перейти к ветви $V2$. Здесь переменные M, C, $Z1$, $Z2$, $X1$, $X2$, $Y1$, $Y2$ обозначают адреса ячеек памяти, содержащих значения (возможно нулевые) соответственно множеств масок, констант и рабочих ячеек; V, $V1$,$V2$ – номера ветвей дерева; $F1$, $F2$ – арифметико-логические операции. Одна или обе операции на втором шаге могут как отсутствовать (с образованием цепочки проверки условий), так и представлять некоторое множество операций.

Для повышения производительности симулятора МПК используется то обстоятельство, что переменные M, C, $F1$, $F2$, $V1$, $V2$ для каждой ветви имеют свое постоянное значение. Это позволяет удалить в каждой такой команде шаг проверки значений полей и все операции в тупиковой ветви дерева. Для этого (перед выполнением программы) ее необходимо перекомпилировать в промежуточный код. К сожалению, команды с анализом полей, которые могут изменяться в результате каких-либо операций, не могут быть ускорены таким образом.

Для повышения производительности симуляции арифметических операций *host*-машина должна работать с разрядностью, не меньшей разрядности эмулируемого процессора с последующим отсечением незначащих двоичных разрядов (старших для целых чисел и младших для дробных чисел, меньших единицы).

При выборе графической модели представления данных для получения высоких эргономических характеристик необходимо обращать особое внимание на правильное структурирование выводимой информации как по типам данных (соответственно типам мониторов), так и по их взаимному расположению на экране конкретного монитора. Итоговое решение приведено в разделе *Архитектура и реализация*, а ниже даны примеры возможных стратегий в вопросах работы с медиаданными:

- разработка алгоритмов автоматического обновления текстовой и графической информации с отслеживанием ее изменения в памяти и регистрах МПК и его симулятора (оптимальный вариант, не требующий написания кода для МПК);
- выделение общей памяти для обмена данными между ИМОС и МПК (требует написания программного кода для каждого приложения МПК);
- разработка для МПК полноценных драйверов ввода–вывода медиаданных. Лучший и самый сложный вариант – реализация, совместимая с *API OpenMAX*, представляющего на высоком уровне абстракции доступ к функциональности, связанной с аудиовизуальными средствами [4].

Замеры распределения затрат процессорного времени *host*-компьютера при выполнении типовой задачи моделирования показали, что до 90 процентов времени уходит на динамическое обновление данных графических мониторов. При реализации графической модели ИМОС был разработан алгоритм, позволяющий примерно в пять раз повысить скорость обновления данных. Алгоритм содержит четыре условных оператора:

- создание списка адресов записи ячеек памяти, в которые была включена запись при выполнении симулятором очередной команды;

• обновление (по окончании выполнения команды) измененных данных во всех мониторах, непрерывные области сканирования которых содержат адреса из списка адресов записи;

• обновление всех данных монитора при изменении его базовых или индексных адресов сканирования или при его включении;

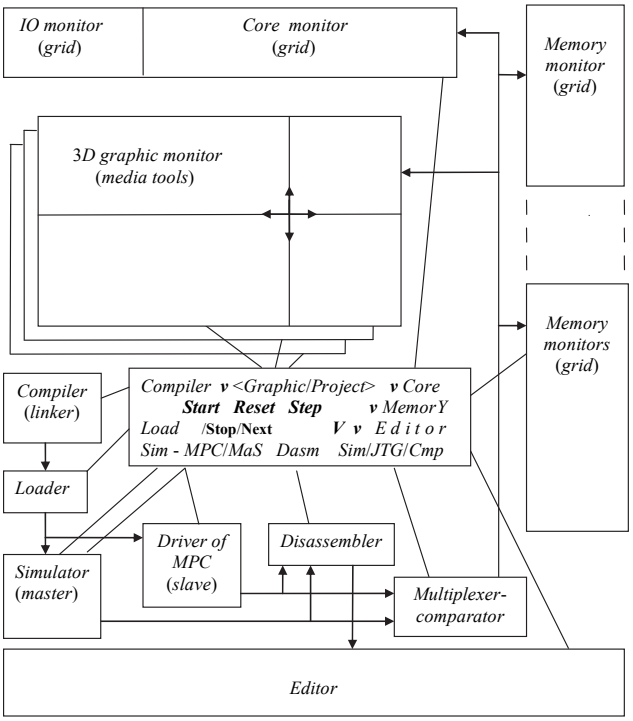
• обновление измененных данных во всех мониторах, непрерывные области сканирования которых содержат адреса из списка адресов записи, полученных при интерактивном вводе данных в один из мониторов.

Архитектура и реализация

Рассматривается интегрированная программная среда, разработанная в рамках проекта по созданию перспективных средств управления и цифровой обработки. ИМОС поддерживает ввод, редактирование и отладку программного кода разрабатываемого МПК на языке его ассемблера, обеспечивает различные режимы графической визуализации процессов преобразования данных при выполнении приложений в реальном устройстве и(или) на его программной модели. Кроме графического, поддерживается также аудио ввод–вывод с использованием средств ОС персонального компьютера.

Программная среда, реализованная на персональном компьютере, обеспечивает управление МПК, используя скоростной последовательный интерфейс. Основные подсистемы ИМОС и их связи с соответствующими элементами управления, продублированными, помимо главного меню, на компактной панели управления (ПУ), показаны на рисунке. Такое же место ПУ занимает на экране компьютера – главном элементе пользовательского *HMI*-интерфейса операционной среды (перемещение ПУ в текущую активную зону экрана, например, рядом с редактором программного кода существенно сокращает путь многократного перемещения «мышки» [5]). Отличительная особенность данного ПУ – ортогональная система элементов управления, т.е. допустима любая комбинация их состояний и все они целесообразны.

Графическая оболочка *HMI*-интерфейса поддерживает комфортную работу с редактором программного кода и рядом интерактивных мониторов, допускающих ввод данных в выполняемую программу.



Структура моделирующей среды для создания и отладки систем цифровой обработки

Editor – текстовый редактор с возможностью настройки шрифтов, автосегментацией и независимым раскрытием сегментов программного кода, поддержкой отображения адресов и номеров командных строк, а также автофокусировкой на выполняемую команду и многоцветной подсветкой синтаксических групп элементов кода и данных.

3D Graphic monitor – мультиформатная графическая среда с возможностью вывода до 16 диаграмм или графиков (в четырех зависимо перестраиваемых окнах), отображающих динамическое состояние непрерывных областей памяти, а также развертку на графике последовательности этих состояний во времени (параметры *SHistory* и *LHistory*). Поддерживается индивидуальное изменение визуальных параметров графиков с возможностью ручной, автоматической и следящей настройки базовых и индексных адресов сканируемых областей памяти, а также ввод графиков при помощи «мыши».

Memory monitors – мультиформатные таблицы, отображающие динамическое состояние

непрерывных областей памяти (до восьми). Поддерживается индивидуальное изменение формата, ручная и автоматическая настройка базовых и индексных адресов таблиц, шрифтов, формата представления данных и их адресов (целые и дробные десятичные, шестнадцатеричные, символы).

Core monitor – мультитабличная структура, отображающая архитектуру МПК и состояние ее программно-доступных регистров АЛУ, адресов, индексов, состояний и др.

IO monitor – аналогичная структура для портов ввода–вывода.

Элементы управления на ПУ позволяют выполнять основные действия с перечисленными программными блоками:

- включить – разрешить динамическое обновление отображаемой монитором информации или прокрутить содержимое редактора до выполняемой на текущем шаге команды;
- выключить – запретить динамическое обновление (с целью повышения производительности процесса эмуляции);
- развернуть – занять область графического монитора и редактора;
- свернуть – занять верхнюю или нижнюю (для редактора) половину экрана.

Элементы управления на ПУ представлены своими именами – надписями с изменяемым, в зависимости от состояния элемента, цветом. Для включения и выключения служат символы *v* и *V* – для общего выключения обновления.

Угловые скобки < и > справа и слева от элемента управления графическим монитором обеспечивают горизонтальную прокрутку его содержимого: <*File*|*Project*|*sBase*|*Tools*|*Graphic*> и, соответственно, последовательный вывод на экран данного монитора: дерева файловой системы компьютера, архива файлов и параметры текущего проекта, базу программных модулей, набор системных и дополнительных инструментов.

Группа не видимых на экране, но не менее значимых для функционирования ИОС программ включат в себя совмещенный с линковщиком программных модулей (файлов) компилятор *Compiler* (*linker*) и загрузчик *Loader*. Для проектов цифровых систем с системой машинных кодов, отличных от существующих, компилятор пишется отдельно. Это наш случай, но здесь не обсуждается. Загрузчик допускает загрузку объектных файлов как в программу симулятор, так и через *Driver* в проектируемый МПК(*MPC*) даже при неполной работоспособности последнего. При отключенном компиляторе в МПК и/или модель загружается прежний исполняемый файл (контроль загрузки). При отключенном для ускорения отладки загрузчике используются ранее загруженные файлы.

Simulator (*master*) – программный симулятор МПК, поддерживаются режимы: симуляция *Sim* – работа МПК–*MPC*, совместная работа *Sim* – *MPC*, работа в режиме *Master* – *Slave Sim* – *MaS*.

Multiplexer – comparator выбирает в качестве источника данных мониторов либо аппаратно реализованный МПК (индицируется надпись *JTG*), либо его программную модель – симулятор (*Sim*). При задании режима (*Cmp*) производится сравнение обоих потоков данных, на мониторы поступают данные из аппаратной реализации, в таблицах монитора памяти несовпадающие данные выделяются шрифтом.

Disassembler – дизассемблер, поддерживает трансляцию машинного кода в код ассемблера и загрузку его в файл листинга.

Start/*Stop*/*Next* – пуск процесса выполнения программы (с возможностью пошаговой подсветки выполняемой команды в редакторе программ и индикации результатов ее выполнения в табличном и графическом виде) или его остановка с возможностью продолжения.

Reset – кнопка сброса счетчика команд и ИМОС в состояние, соответствующее первоначальному пуску программы.

Step – кнопка пошагового режима выполнения программы.

Все элементы управления ПУ, кроме группы *Start–Reset–Step*, могут быть установлены в выключенное состояние, которому соответствует слабая яркость надписей (на рисунке не показано).

Окончание на стр. 70

В ИМОС реализован ряд современных технологий *HMI*-интерфейса:

- автосохранение в отдельном файле проекта всех текущих установок, списка файлов проекта, а также настроек экрана, редактора и мониторов при выходе из среды или запуске исполняемого файла проекта [6];
- автозапуск среды при открытии ассоциированного с ней файла проекта;
- возможность задания цветовой гаммы окон и панелей.

Заключение. Разработанная интегрированная программная среда в рамках проекта по созданию перспективных средств управления и цифровой обработки поддерживает ввод, редактирование и отладку программного кода разрабатываемого МПК на языке его ассемблера, обеспечивает различные режимы графической визуализации процессов преобразования данных при выполнении приложений в реальном устройстве и(или) на его программной модели.

1. *Корнеев В.В.*, *Кисилев А.В.* Современные микропроцессоры – СПб.: БХВ-Петербург, 2003. – 448 с.
2. *Наливкин А.В*. Микропроцессоры и ЭВМ в измерительной технике: Электр. учебник. – СПбГУ ИТМО. – *http*://*de.ifmo.ru*. 20.09.2012.
3. *Синицин С.В*., *Налютин Н.Ю*. Верификация программного обеспечения: Курс лекций. – М.: МИФИ, 2006. – 157 с.
4. *OpenMAX* – The Standart for Media Library Portability. – *http://www.khronos.org*. 21.09.2012.
5. *Acoustic* commander – интегрированная операционная среда для измерения и расчета акустических параметров / А.В. Палагин, М.В. Семотюк, Е.Н. Чичирин и др. // Комп`ютерні засоби, мережі та системи. – 2009. – № 4. – С. 3–10.
6. *Бильфелд Н.В*., *Затонский А.В*. Основы разработки интерфейсов. Реализация в системе *Borland Delphi*: Учеб. пособие. – Перм. гос. техн. у-нт, Березниковский филиал. – 2010. – 96 с.

Поступила 01.10.2012
Тел. для справок: +38 044 526-3348, 526-0656, 526-6439 (Киев)
E-mail: *yaviz@ukr.net*