WInter – среда отладки программного обеспечения мультипроцессорных систем

Расширение рынка процессоров замедляется из-за отсутствия средств быстрого создания системного программного обеспечения и из-за необходимости переучиваться с одной среды разработки программного обеспечения на другую при переходе от одного микроконтроллера к другому. В статье пойдет речь о системе WInter, созданной для решения этих проблем.

Михаил Долинский,
Игорь Ермолаев,
Алексей Толкачев,
Игорь Гончаренко

dolinsky@gsu.unibel.by

Введение

Стремительное развитие микроминиатюризации в цифровой электронике привело к тому, что рынок микропроцессоров и микроконтроллеров (МП/МК), от 4-битных до 256-битных, стремительно расширяется. Технологии программируемых логических интегральных схем (ПЛИС), обеспечивающие быстрый цикл итераций проектирования, значительную емкость (до 10 и более миллионов вентилей) и производительность (до сотен МГц) сделали реальной разработку процессора или даже мультипроцессорной системы под конкретную прикладную проблему. Появились фирмы, которые специализируются на разработке HDL-описаний (VHDL/Verilog) моделей процессоров как общего, так и специального назначения.

Однако расширение рынка процессоров замедляется из-за отсутствия средств быстрого создания системного программного обеспечения: среды отладки прикладного программного обеспечения; симулятора системы команд; ассемблера; линкера; С-компилятора; средств совместной симуляции и отладки программного обеспечения, аппаратного обеспечения и внешней среды функционирования встроенной системы; средств симуляции и отладки программного обеспечения мультипроцессорных систем. Неудобства вызывает и необходимость переучиваться с одной среды разработки программного обеспечения на другую при переходе от одного МП/МК к другому. Для решения этих проблем необходима разработка универсальных кросс-средств, которые способны облегчить труд программистов, работающих с разнообразным спектром микропроцессоров.

Одна из таких систем была разработана в Гомельском государственном университете им. Ф. Скорины в 1987 году. Первые две редакции системы были выпущены в 1989 (для СМ ЭВМ) и 1993 (для IBM PC под MS-DOS). В 2000 году выпущена третья редакция системы (WInter — NewIT.gsu.unibel/WInter), которой и посвящена данная статья.

Сравнительный анализ средств разработки программного обеспечения микропроцессорных систем

Сравнение возможностей разработки программного обеспечения проводилось на момент выпуска первой версии WInter (май 2000 года) с наиболее распространенными на территории СНГ средствами отладки однопроцессорных систем по пяти составным критериям: интерфейс, редактор, окна процессора, команды выполнения и возможности отладки. Каждый составной критерий включал в себя набор свойств. Если у анализируемой системы данное свойство присутствовало, то ей начислялся балл. В приведенной ниже таблице указано, сколько баллов набрала каждая из систем отладки.

Результаты сравнения приведены в таблицах 1–6. В таблицах 2–6 приведена расшифровка каждого из составных критериев.

Уникальная черта системы WInter — *универсальность*. Интегрированная среда разработки программного обеспечения встроенных систем WInter способна взаимодействовать с моделью микропроцессора: получать информацию о микропроцессоре, настраиваться на него и предоставлять пользователю следующие возможности:

- создание мультипроцесссорных систем;

Таблица 1. Интегральная оценка средств отладки

Составной критерий	ProView32 3.3	MPLab 5.0	AVR Studio 3.0	Vision 2.04b	CCStudio 1.0	IAR EW 2.31A	WInter
Год выпуска	1996	2000	2000	2000	1999	1999	2000
Интерфейс	2	5	2	6	6	5	9
Редактор	3	0	2	4	3	3	6
Окна процессора	8	6	9	8	7	8	16
Команды выполнения	3	4	6	5	4	6	7
Возможности отладки	5	5	3	4	2	4	10
Итого	21	20	22	27	22	26	48

Таблица 2. Сравнительный анализ интерфейса пользователя

Критерий	ProView32 3.3	MPLab 5.0	AVR Studio 3.0	Vision 2.04b	CCStudio 1.0	IAR EW 2.31A	WInter
Запоминание текущего проекта	•	•		•		•	•
Возможность загрузить любое из ранее сохраненных состояний		•			•	•	•
Сохранение и восстановление расположения окон, панелей инструментов и т. п.		•		•	•	•	•
Окна всегда находятся на том месте, куда их переместили		•	•	•	•	•	•
Механизм избавления от перекрытий окон				•	•		•
Автоматическая установка размеров окна в зависимости от его содержимого				•	•		•
Возможность убирать из окна приложения любой элемент интерфейса	•	•	•	•	•	•	•
Полноэкранный режим (видно только одно окно редактора)							•
Выбор языка интерфейса							•
Итого	*2*	*5*	*2*	*6*	*6*	*5*	*9*

Таблица 3. Сравнительный анализ возможностей редактора

Критерий	ProView32 3.3	MPLab 5.0	AVR Studio 3.0	Vision 2.04b	CCStudio 1.0	IAR EW 2.31A	WInter
Подсветка синтаксиса языка	•		•	•	•	•	•
Возможность изменять цвета	•			•	•	•	•
Для каждого языка своя схема подсветки				•			•
Автоотступ при редактировании	•		•	•	•	•	•
Табуляция по столбцам							•
Шаблоны (code insight)							•
Итого	*3*	*0*	*2*	*4*	*3*	*3*	*6*

Таблица 4. Сравнительный анализ поддерживаемых окон процессора

Критерий	ProView32 3.3	MPLab 5.0	AVR Studio 3.0	Vision 2.04b	CCStudio 1.0	IAR EW 2.31A	WInter
Окно с дампом памяти	•	•	•	•	•	•	•
Окно с регистрами	•	•	•	•	•	•	•
Окно с флагами			•	•			•
Окно с битами							•
Окно с переменными	•	•	•	•	•	•	•
Окно со стеком	•	•		•	•	•	•
Окно с дизассемблером	•	•	•	•	•	•	•
Подсветка другим цветом изменившихся значений		•	•	•	•	•	•
Отображение адреса текущей ячейки в дампе памяти	•		•				•
Изменение количества колонок в дампе	•		•	•	•		•
Выбор отображаемых регистров, флагов, битов						•	•
Изменение порядка отображения регистров, флагов, битов							•
Изменение отображаемых имен регистров, флагов, битов			•				•
Настройка переменных в памяти	•					•	•
Отсутствие мерцания содержимого окон в режиме анимации							•
Отображение значений ресурсов процессора на фоне произвольного изображения							•
Итого	*8*	*6*	*9*	*8*	*7*	*8*	*16*

Таблица 5. Сравнительный анализ команд выполнения

Критерий	ProView32 3.3	MPLab 5.0	AVR Studio 3.0	Vision 2.04b	CCStudio 1.0	IAR EW 2.31A	WInter
Выполнить оператор				•		•	•
Выполнить инструкцию	•	•	•	•	•	•	•
Выполнить шаг	•	•	•	•	•	•	•
Выполнить до выхода из подпрограммы			•	•	•	•	•
Выполнить до курсора		•	•	•	•	•	•
Анимация по инструкциям	•	•	•			•	•
Анимация по шагам			•				•
Итого	*3*	*4*	*6*	*5*	*4*	*6*	*7*

Таблица 6. Сравнительный анализ возможностей отладки

Критерий	ProView32 3.3	MPLab 5.0	AVR Studio 3.0	Vision 2.04b	CCStudio 1.0	IAR EW 2.31A	WInter
Простые точки останова	•	•	•	•	•	•	•
Условные точки останова	•	•		•	•	•	•
Точки обновления	•	•					•
Точки установки значения							•
Точки проверки значения							•
Показ значение переменной в окне подсказки				•		•	•
Поддержка стандартной периферии	•	•	•	•		•	•
Поддержка произвольной периферии							•
Использование реальных микроконтроллеров	•	•	•				•
Автоматическое тестирование							•
Итого	*5*	*5*	*3*	*4*	*2*	*4*	*10*

- создание систем со сложной периферией и внешней средой;
- работа с объектами микропроцессора (дампом памяти, регистрами, флагами, битами, стеком и дизассемблером);
- загрузка и выполнение программ;
- средства для редактирования и трансляции исходных текстов;
- работа с точками останова (простыми, условными, по доступу);
- работа с точками действия (обновить содержимое окон, установить значение переменной, проверить значение переменной);
- пошаговое выполнение и анимация;
- работа с переменными программы;
- средства для верификации и документирования.

WInter обладает современным интерфейсом, высокой скоростью работы и низкими требованиями к системным ресурсам. Кроме того, есть возможность интеграции с системой высокоуровневого проектирования цифровых устройств HLCCAD, что позволяет вести ускоренную совместную разработку программного и аппаратного обеспечения встроенных цифровых систем.

Теоретические основы симуляции программного обеспечения мультипроцессорных систем

Открытые моделируемые компоненты

Универсальный интерфейс моделируемого компонента

Универсальный интерфейс моделируемого компонента построен с помощью технологии COM (Common Object Model). Данная технология позволяет разрабатывать и использовать модели процессоров и устройств, написанные на разных языках программирования.

Существенным недостатком этой технологии является невозможность работать со свойствами объекта напрямую. Для получения или установки значения свойства необходимо вызывать соответствующий метод. Это замедляет работу с объектом и увеличивает количество методов в интерфейсе.

Для решения этой проблемы разработано расширение технологии COM. Объект представляется интерфейсом и структурой со значениями свойств объекта. Первым полем этой структуры является указатель на интерфейс с объектом, чьи данные хранятся в этой структуре, дальше идут свойства объекта, например:

```
struct CDeviceData  {
   IDevice *D; // Указатель на интерфейс с устройством
   char        *Type;      // Тип устройства
   char        *Name;      // Имя устройства
   CResourceData  *Resources;  // Список ресурсов
         …
};
```

Вместо указателя на интерфейс теперь надо хранить указатель на структуру. В результате такого расширения COM со свойствами объекта можно работать напрямую, а для вызова методов объекта следует использовать указатель на интерфейс с объектом, хранящийся в структуре.

Согласной этой технологии все интерфейсы наследуются от IUnknownData, который в свою очередь наследуется от IUnknown. В интерфейс IUnknownData добавлен метод QueryInterfaceData. Этот метод позволяет по идентификатору интерфейса получить указатель на структуру с его свойствами. Схематическое представление объекта приведено на рис. 1.

Универсальный интерфейс моделируемого компонента состоит из двух базовых классов:

- CDevicesData — список моделей устройств;
- CDeviceData — модель устройства.

Для работы с той или иной моделью устройства необходимо загрузить соответствующую DLL-библиотеку и получить указатель на объект CDevicesData, вызвав функцию CreateDevices. Затем следует получить указатель на объект CDeviceData, вызвав метод CreateDevice объекта CDevicesData. В этот метод передаются имя устройства и тип устройства. Список поддерживаемых типов устройств можно получить, используя свойства Types и TypesCount объекта CDevicesData.

Объект CDeviceData содержит следующие свойства:

- **D** — указатель на интерфейс с устройством;
- **Type** — тип устройства;

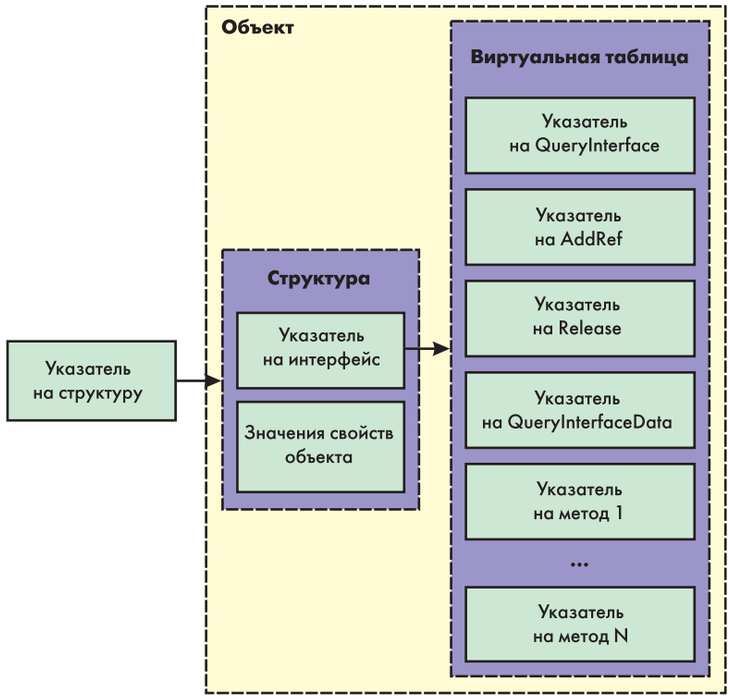

Рис. 1. Структура объекта модели

- **Name** — имя устройства;
- **Resources** — список всех ресурсов устройства;
- **Memories** — список памятей устройства;
- **Registers** — список регистров устройства;
- **Flags** — список флагов устройства;
- **Bits** — список битов устройства;
- **Parameters** — список параметров устройства;
- **Frequency** — тактовая частота устройства;
- **Tackts** — количество выполненных тактов.

Используя эти свойства, программный комплекс отображает состояние устройства и получает информацию, необходимую для моделирования. Кроме обычных свойств, модель устройства имеет и ряд обработчиков событий. Большинство обработчиков вызываются моделью при необходимости выполнения действий, которые сама модель полностью выполнить не может:

- OnResourcesChanged вызывается моделью после изменения списка ресурсов;
- OnSendText вызывается моделью при необходимости вывести текстовое сообщение;
- OnBeforeRead вызывается моделью перед чтением ресурса;
- OnAfterWrite вызывается моделью после записи ресурса.

Методы OnBeforeRead и OnAfterWrite вызываются не для всех ресурсов, а только для тех, которые могут использоваться в операциях обмена между моделями устройств. Обычно это контакты (пины) устройства и внешняя память.

Единственный обработчик, который вызывается подсистемой моделирования, а не моделью, является OnExecute. Реакцией на вызов этого обработчика, должно быть выполнение моделью текущей активности. Для ядра процессора — это выполнение инструкции, для периферийного устройства — это анализ значений на контактах и выполнение соответствующего алгоритма. Обработчик должен вернуть количество тактов, затраченных на выполнение этой активности.

Интерфейс объекта CDeviceData содержит всего один метод — PowerOn. Этот метод вызывается, когда нужно произвести выполнение процедуры начальной инициализации, перед моделированием.

Интерфейс модели процессора

Интерфейс модели процессора базируется на универсальном интерфейсе моделируемого компонента. Таким образом, модели процессоров могут участвовать в процессе моделирования наряду с другими моделями устройств. Свойства модели процессора хранятся в структуре CProcessorData производной от CDeviceData, а методы представлены интерфейсом IProcessor производным от IDevice.

Для получения указателя на объект, представляющий модель процессора, необходимо сначала получить указатель на объект, представляющий модель соответствующего устройства. Так как CDeviceData является базовым классом для CProcessorData, то для получения указателя на CProcessorData необходимо воспользоваться методом QueryInterfaceData:

```
CDeviceData *pDD;
...
CProcessorData *pPD;
if (pDD->D->QueryInterfaceData(IID_IProcessor,
    reinterpret_cast<void **>(&pPD)) == S_OK) {
    // Взаимодействие с моделью процессора
    ...
}
```

Объект CProcessorData, вдобавок к унаследованным от CDeviceData, содержит следующие свойства:

- PiCPU — указатель на интерфейс с процессором;
- Instructions — количество выполненных инструкций.

Кроме свойств объекта в структуре хранится указатель на обработчик события OnBeforeExecute. Этот обработчик устанавливается подсистемой моделирования и вызывается моделью процессора перед выполнением каждой инструкции. В обработчик передается:

- указатель на кодовую память, из которой будет браться код очередной инструкции;
- количество тактов, затраченных на выполнение предыдущей инструкции;
- тип предыдущей инструкции.

С использованием этого обработчика осуществляется поддержка точек действия, пошаговое исполнение и мультипроцессорное моделирование.

Интерфейс объекта CProcessorData, вдобавок к унаследованным от CDeviceData, содержит следующие методы:

- **Trace** — выполнять инструкции, с вызовом OnBeforeExeceute перед выполнением каждой инструкцией;
- **Run** — выполнять инструкции, с вызовом OnBeforeExecute перед инструкциями, для которых указаны точки действия;
- **Stop** — остановить выполнение инструкций;
- **SetBreakpoint** — установить признак наличия точки действия для инструкции по указанному адресу;
- **ReSetBreakpoint** — сбросить признак наличия точки действия для инструкции по указанному адресу.

Механизм симуляции мультипроцессорных систем

Алгоритм моделирования мультипроцессорной системы можно представить в виде блок-схемы, приведенной на рис. 2. В прямоугольных блоках под чертой русскими буквами указаны формулы, объяснение которых будет приведено ниже, английскими буквами указаны методы, обработчики и свойства класса CDeviceData, являющегося базовым классом универсальной моделируемого компонента.

Моделирование происходит по следующей схеме. Производится сброс по включению питания всех моделей устройств (процессоров и внешней периферии). Сброс осуществляется вызовом метода PowerOn. Определяется начальное модельное время каждого устройства по формуле:

$$MBi = TCi / TЧi$$

где MBi — модельное время i-го устройства, TCi — количество тактов, затраченное i-ым устройством на процедуру сброса (для получения этой величины используется свойство Tackts). TЧi — тактовая частота устройства (эта величина устанавливается в среде моделирования и хранится в переменной Frequency). После этапа подготовки осуществляется само моделирование, состоящие из пяти этапов.

Первый этап заключается в проверке условий останова процесса моделирования и переходе в конец моделирования при выполнении условий. Процесс моделирования может быть остановлен как пользователем, так и при достижении одной из точек останова.

На втором этапе определяется устройство с минимальным временем моделирования, то есть устройство, работа которого меньше всего моделировалась. Пусть это будет k-ое устройство.

На третьем этапе осуществляется обработка модельных соединений k-го устройства. Происходит передача данных от подсоединенных устройств в ресурсы k-го устройства и, при необходимости, устройство оповещается об этих изменениях.

Четвертый этап заключается в выполнении очередной активности k-го устройства. Для ядра процессора это выполнение инструкции, для периферии — реакция на изменения ресурсов.

На пятом этапе осуществляется изменение модельного времени k-го устройства по формуле:

$$MBk = MBk + TAkj / TЧk$$

где MBk — модельное время k-го устройства, TAkj — количество тактов, затраченное k-ым устройством на выполнение j-ой активности (это значение передается в обработчик OnBeforeExecute в качестве параметра tkt). TЧk — тактовая частота k-го устройства (хранится в переменной Frequency).

Для предотвращения переполнения при длительном моделировании модельные времена всех устройств (MBj) периодически уменьшаются на минимальное модельное время, при условии, что это время больше либо равно десяти в десятой степени (10^{10}).

$$MBk = \min MBj$$
$$MBj = MBj - MBk$$

Работа алгоритма проиллюстрирована на рис. 3. Ti — это время окончания очередной активности (TCi / TЧi или MBk + TAkj / TЧk). Последовательность выполнения активностей следующая:

$$A_{10}, A_{20}, A_{30}, A_{11}, A_{21}, A_{31}, A_{22}, A_{32}$$

Активность Ai0 — это сброс i-го устройства.

При моделировании мультипроцессорной системы необходимо решить две задачи:

- хранение списка устройств;
- нахождение устройства с минимальным временем моделирования.

Для решения этих задач предлагается использовать так называемую «частично упорядоченную кучу». Эта структура данных представляет собой массив элементов, но данные организованы в виде двоичного дерева. Достигается это следующим образом: каждому элементу массива с индексом i сопоставляется левый и правый узлы с индексами 2·i и 2·i+1 соответственно. Данные хранятся в массиве таким образом, что модельное время элемента с индексом i меньше либо равно модельному времени элементов с индексами 2·i и 2·i+1 (то есть левого и правого узлов).

В результате заполнения массива по описанной выше схеме первым элементом массива является указатель на устройство с минимальным временем моделирования.

После выполнения текущей активности этого устройства, его модельное время изменяется и выполняется так называемая процедура «утряски». В результате первым элементом массива опять оказывается указатель на устройство с минимальным временем моделирования. Причем структура двоичного дерева не меняется, то есть модельное время элемента с индексом i меньше либо равно модельному времени элементов с индексами 2·i и 2·i+1.

Благодаря специальной структуре этого массива и эффективной реализации алгоритма, процедура «утряски» выполняется очень быстро и имеет сложность $\log_2(n)$, где n — количество моделируемых устройств. Кроме того, хранение указателей на устройства в виде массива экономит память и минимизирует время доступа к элементам.

Интеграция с системой симуляции аппаратного обеспечения

Для повышения адекватности модели мультипроцессорной системы можно использовать потактовую симуляцию работы процессоров и внешней периферии. Достичь этого позволяет интеграция средств разработки программного обеспечения и моделирования мультипроцессорных систем с системой симуляции аппаратного обеспечения HLCCAD. Процесс интеграции сводится к решению следующей задачи.

Существуют два интерфейса:

- универсальный интерфейс моделируемого компонента (в дальнейшем будем называть интерфейс A);
- интерфейс модели устройства, используемой в системе симуляции аппаратного обеспечения (в дальнейшем будем называть интерфейс B).

Необходимо предоставить возможность средствам разработки программного обеспечения и моделирования мультипроцессорных систем взаимодействовать через интерфейс A с моделью процессора (поддерживающей интерфейс B), используемой в системе симуляции аппаратного обеспечения.

Для решения этой задачи используется так называемый «переходник». «Переходник» — это объект, который поддерживает интерфейс A и хранит у себя указатель на объект, представляющий собой модель процессора, используемую в системе симуляции аппаратного обеспечения (в дальнейшем будем называть указатель на процессор).

«Переходник» с помощью указателя на процессор получает информацию о ресурсах процессора и формирует списки ресурсов в соответствии с интерфейсом A. Кроме того, «переходник» сопоставляет вызовы методов и обработчиков событий интерфейса A, мето-

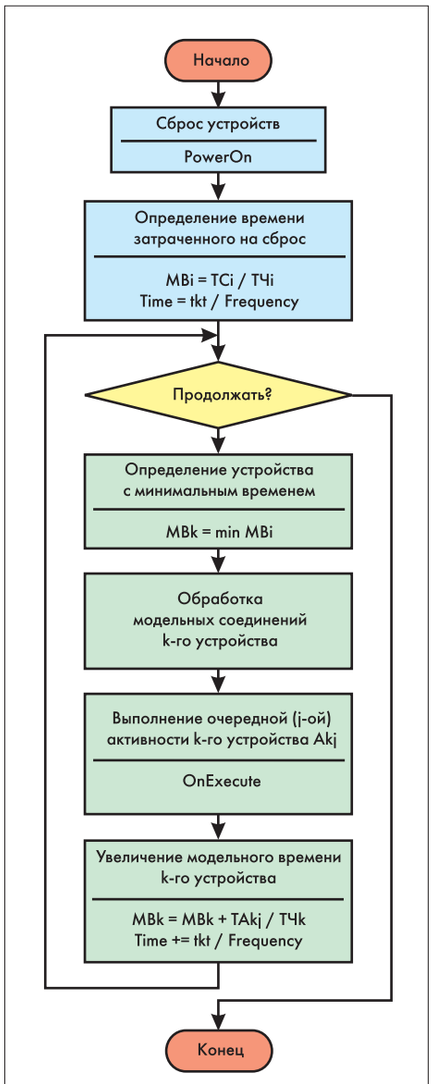

Рис. 2. Алгоритм моделирования мультипроцессорной системы

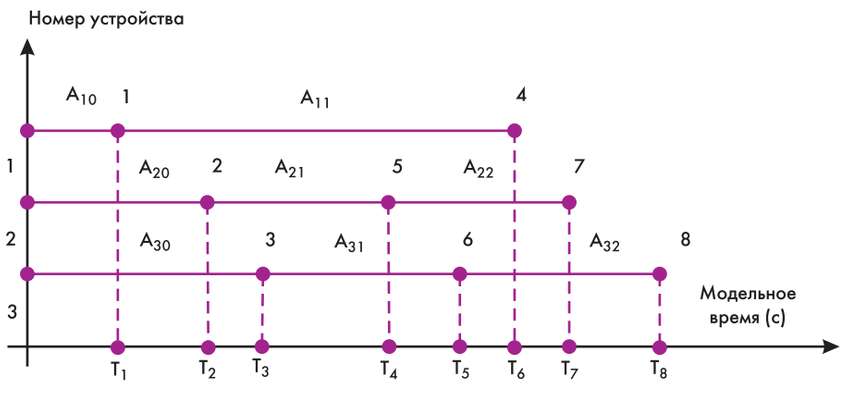

Рис. 3. График выполнения активностей устройствами

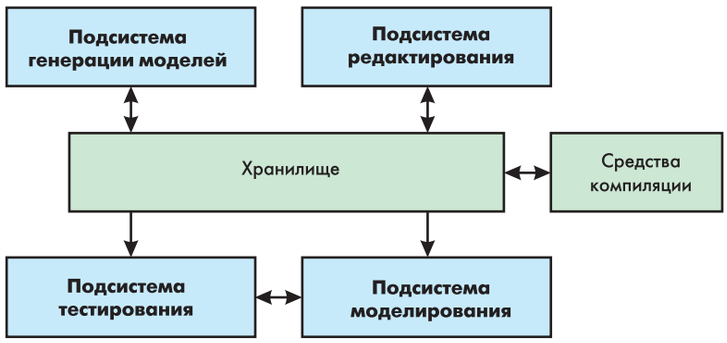

Рис. 4. Структурная схема комплекса WInter

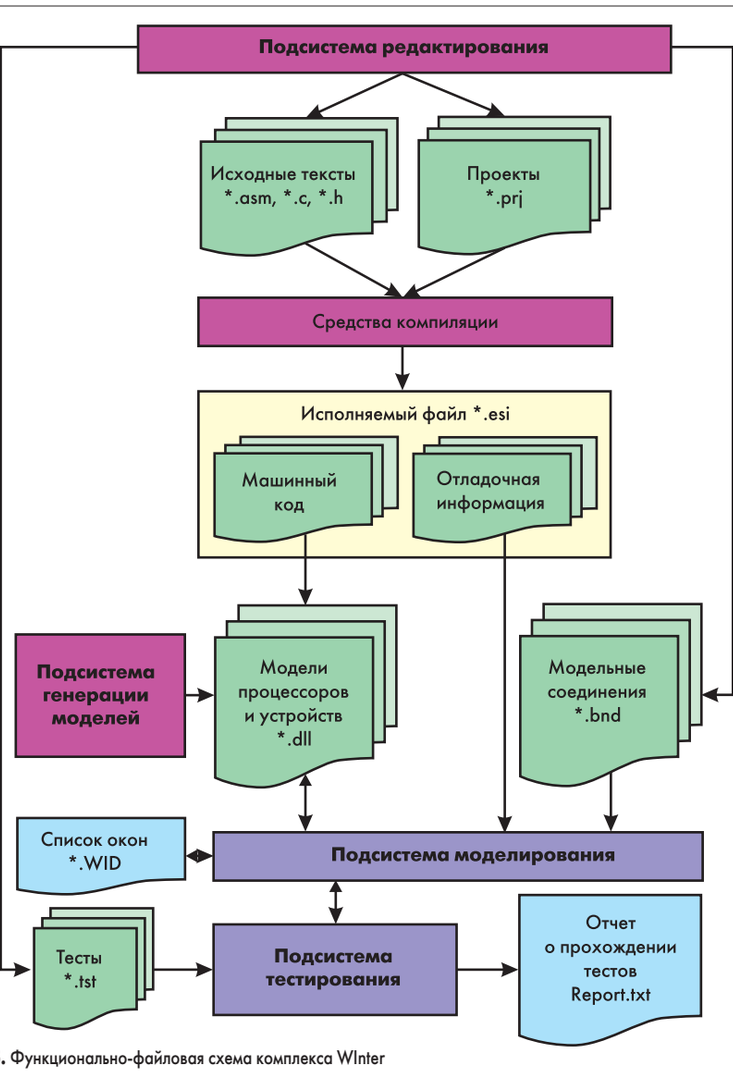

Рис. 5. Функционально-файловая схема комплекса WInter

дам и обработчикам событий интерфейса B. А также сопоставляет вызов обработчиков событий интерфейса B, обработчикам интерфейса A.

Таким образом, подсистема моделирования работает с «переходником» как с обычной моделью процессора, предоставляя пользователю весь набор средств проектирования, отладки и верификации. При этом производится потактовая симуляция работы процессора и внешней периферии, что существенно повышает адекватность моделируемой системы.

Аппаратные модели

Для повышения адекватности и скорости моделирования обеспечивается использование реальных процессоров и периферийных устройств. Достичь этого позволяют аппаратно-программные разработки UniICE и UniICS.

Для интеграции UniICE и UniICS в процесс моделирования разработана специальная методика, базирующаяся на универсальном интерфейсе моделируемой компоненты.

Использование реального процессора при моделировании, осуществляется с помощью так называемой «промежуточной, настраиваемой модели процессора». Данная модель поддерживает универсальный интерфейс моделируемого компонента и является «посредником» между подсистемой моделирования и UniICE. UniICE предоставляет доступ к ресурсам реального процессора, подключенного с помощью специального прибора к порту компьютера LPT. Также UniICE позволяет управлять выполнением программы на реальном процессоре.

Для работы такой модели необходим файл с описанием процессора. Используя этот файл, модель формирует списки ресурсов процессора и определяет механизм взаимодействия с UniICE. Таким образом, благодаря использованию универсального интерфейса моделируемого компонента, с точки зрения подсистемы моделирования, взаимодействие с реальным процессором ничем не отличается от взаимодействия с моделью процессора.

Использование реального периферийного устройства осуществляется аналогично использованию реального процессора. Для этого разработана «промежуточная, настраиваемая модель периферийного устройства». Данная модель поддерживает универсальный интерфейс моделируемого компонента и позволяет, с помощью UniICS, «связать» контакты реального устройства с контактами моделей устройств.

UniICS представляет собой специальный прибор, подключаемый через LPT-порт компьютера. У этого прибора есть набор контактов, к которым можно подключать реальные устройства. UniICS предоставляет возможность прочитать значения с этих контактов и установить значения на этих контактах.

Для работы такой модели необходимо задать набор контактов, с которыми могут быть «соединены» контакты моделей других устройств. Набор контактов задается путем указания имени контакта и перечислением контактов UniICS, которые будут использоваться при передаче данных.

Программный комплекс симуляции мультипроцессорных систем и отладки программного обеспечения мультипроцессорных комплексов

Комплекс предназначен для разработки программного обеспечения мультипроцессорных систем.

Структурная схема разработанных средств приведена на рис. 4. Комплекс состоит из четырех основных подсистем:

- подсистема моделирования, предназначенная для моделирования, анализа и исследования мультипроцессорных систем;
- подсистема генерации моделей, предназначенная для получения моделей процессоров и периферийных устройств;
- подсистема редактирования, предназначенная для получения исходных текстов, проектов, модельных соединений и тестов;
- подсистема тестирования, предназначенная для интерактивного и пакетного тестирования программного обеспечения и моделей процессоров.

Функционально-файловая схема приведена на рис. 5.

Файлы комплекса группируются по каталогам.

Каталог As содержит ассемблеры для всех поддерживаемых процессоров, линкер и помощь по ассемблеру.

Каталог Asm содержит примеры исходных текстов на языке ассемблер для всех поддерживаемых процессоров.

Каталог Bin содержит исполняемые файлы комплекса, ресурсы, скрипты для подсветки синтаксиса, языковые файлы, настройки средств компиляции для разных платформ, помощь, настроенные рабочие окружения с примерами использования комплекса.

Каталог Bmp содержит графические файлы с рисунками, используемые в некоторых примерах.

Каталог C содержит примеры исходных текстов на языке C.

Каталог Dis содержит dll-библиотеки для дизассемблирования машинного кода.

Каталог Mdl содержит dll-библиотеки с моделями процессоров и устройств.

Каталог translators-support содержит поддержку внешних трансляторов с языка C и ассемблер.

Основная часть комплекса реализована на языке C++ в среде C++Builder 3.0. Общее число исходных файлов — 669, общее число строк исходного текста — более 87000.

Практическая апробация системы WInter

Разработка программного обеспечения измерительного прибора П-216М

С помощью разработанных технологий спроектирована модель измерительного прибора электрических свойств жидкости П-216М (разработчик и производитель — «Антех», г. Гомель) и реализована управляющая им программа. Прибор состоит из следующих компонентов:

- процессор Atmel 89C55WD (совместимый с Intel 8051);
- два матричных дисплея SED1521 с разрешением 61×32 точек (первый отображал левую половину экрана, второй — правую, в итоге получался экран с разрешением 122×32 точек);
- четырехкнопочная клавиатура;
- EEPROM объемом 512 байт;
- аналого-цифровой преобразователь (АЦП) AD7712;
- цифро-аналоговый преобразователь (ЦАП) DAC7612;
- порт RS-232.

Модель процессора Intel 8051 к началу работ была уже готова, модели клавиатуры и порта RS-232 входят в библиотеку стандартных устройств системы WInter. Необходимо было создать модели матричного дисплея, EEPROM, АЦП и ЦАП. Для облегчения процесса создания моделей использовался набор базовых классов для:

- устройства — CDeviceBase;
- окна устройства — CViewBase;
- обмена данных — CSerialChanel;
- обмена данных по протоколу I²C — CI2CChanel.

Через неделю все модели были готовы и оттестированы. Каждая модель была представлена окном, в котором отображалось состояние устройства и с помощью которого разработчик мог влиять на выполнение программы. Таким образом, моделировалась внешняя среда и внешние воздействия.

Для создания модели измерительного прибора в проект добавили две модели матричного дисплея. Для этих моделей установили разрешение 61×32 точек. Открыли окна устройств и расположили их рядом (рис. 6). Эти два окна имитировали экран реального прибора.

Для взаимодействия с управляющей программой была использована модель клавиатуры. Эту модель настроили так, чтобы она соответствовала реальной клавиатуре используемой в приборе (рис. 7). Теперь разработчик мог взаимодействовать с моделью прибора, нажимая мышью кнопки K1, K2, K3 и K4; или используя клавиши 1,2,3,4 на клавиатуре.

Для имитации EEPROM использовалась соответствующая модель с объемом памяти 512 байт (рис. 8). При выходе из среды моделирования данные, хранимые в EEPROM, сохранялись, а при запуске среды — восстанавливались.

Для имитации АЦП использовалась модель AD7712 (рис. 9). Окно АЦП отображает состояние внутренних регистров устройства (управляющего регистра, регистра данных и калибровочного регистра). С помощью данного окна можно выяснить, что записал драйвер АЦП в управляющий и калибровочный регистры. А также изменять значения регистра данных, имитируя изменения входного аналогового сигнала.

Для моделирования ЦАП использовалась модель DAC7612 (рис. 10). Используя это окно, разработчик мог увидеть: от какого значения зависит выходной аналоговый сигнал и какое значение находится в сдвиговом регистре.

Для моделирования порта RS-232 использовалась модель устройства с окном, представляющим собой терминал (рис. 11). С помощью этого окна разработчик мог видеть данные, переданные через RS-232, а также мог посылать команды управляющей программе. В данном случае прибор сообщал:

- информацию о текущих значениях измеряемых величин, по команде «?.»;
- информацию о калибровочных значениях, по команде «?K.»;
- информацию о производителе, модели и версии прибора, а также дату последнего изменения управляющей программы, по команде «?P.»

Для настройки взаимодействий между всеми компонентами измерительного прибора использовался редактор модельных соединений. В результате было получено следующее описание связей:

```
$I8051.Memory.External Data Memory[0x4000] <~~>
                                        $SED1521_1.INSTR
$I8051.Memory.External Data Memory[0x4001] <~~>
                                        $SED1521_1.DATA
$I8051.Memory.External Data Memory[0x2000] <~~>
                                        $SED1521_2.INSTR
$I8051.Memory.External Data Memory[0x2001] <~~>
                                        $SED1521_2.DATA
```

```
$I8051.Pin.INT1 <?- $Keyboard.Key
$I8051.Pin.P1.7 -?> $Keyboard.Clk
$I8051.Pin.P1.6 -~> $Keyboard.DIn
$I8051.Memory.External Data Memory[0xE000.0] ~~> $AD7712.TFS
$I8051.Memory.External Data Memory[0xE000.1] ~~> $AD7712.RFS
$I8051.Memory.External Data Memory[0xE000.2] ~~> $AD7712.A0
$I8051.Pin.P1.2 <?-?> $AD7712.SDATA
$I8051.Pin.P1.3 -?> $AD7712.SCLK
$I8051.Pin.P1.4 -~> $DAC7612.SDI.0
$I8051.Pin.P1.5 -~> $DAC7612.CLK.0
$I8051.Memory.External Data Memory[0xE000.6] ~~>
                                        $DAC7612.LD.0
$I8051.Pin.INT0 <?- $AD7712.DRDY
$I8051.Pin.P1.0 -?> $AT24Cxx.SCL
$I8051.Pin.P1.1 <?-?> $AT24Cxx.SDA
$I8051.Bit.RI <?-~> $Terminal.RI
$I8051.Bit.TI <?-~> $Terminal.TI
$I8051.Register.SBUF <~-?> $Terminal.SBUF
```

Теперь модель измерительного прибора полностью готова (пример окна приложения приведен на рис. 12). Осталось написать управляющую программу. Сначала разрабатывались и отлаживались драйвера устройств, затем механизм поддержки окон с отображением динамически изменяющихся данных. Далее был создан механизм переключения между этими окнами. Затем команда разработчиков реализовала все необходимые математические вычисления и преобразования. В самом конце работ поддержали взаимодействие через RS-232, выключение экрана при простое прибора и парольную защиту. Полученную программу протестировали на модели, а затем проверили на реальном приборе. Программа содержала приблизительно 8200 строк на языке C и ассемблере, машинный код занимал 20 кбайт. На всю работу было потрачено 80 человеко-часов. То есть средняя производительность была 102,5 строки в час и 256 байт в час для системы из восьми взаимодействующих компонентов.

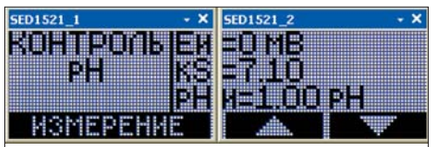

Рис. 6. Пример окон матричных дисплеев

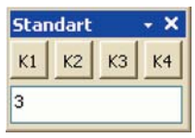

Рис. 7. Пример окна клавиатуры

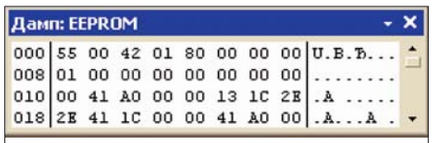

Рис. 8. Пример окна EEPROM

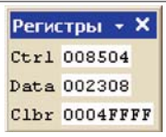

Рис. 9. Пример окна AD7712

Рис. 10. Пример окна DAC7612

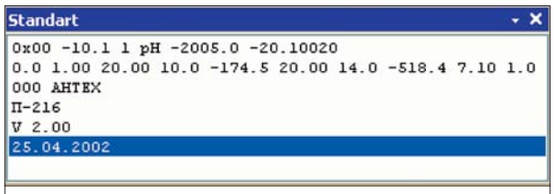

Рис. 11. Пример окна RS-232

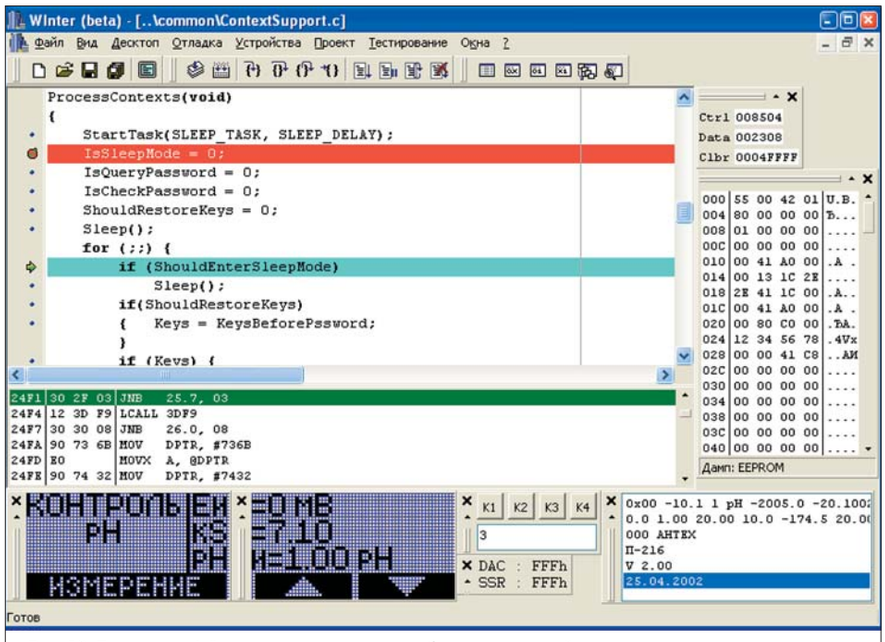

Рис. 12. Пример окна WInter-настроенного для разработки управляющей программы

Разработка программного обеспечения измерительного прибора И-160М

Для прибора И-160М была проведена работа, аналогичная прибору П-216М. Только управляющая программа для И-160М:

- обладала более сложным интерфейсом пользователя (использовались такие элементы управления, как горизонтальные, вертикальные и прямоугольные меню);
- позволяла измерять около 30 ионов;
- занимала 50 кбайт кодовой памяти, а не 20 кбайт, как программа для П-216М.

Измерительный прибор И-160М состоит из следующих компонентов:

- процессор Intel 8051;
- матричный дисплей Powertip PG12864LRU-FTA-H-SA с разрешением 128×64 точек;
- шестикнопочная клавиатура;
- EEPROM объемом 2 кбайт;
- АЦП ADS1212;
- ЦАП DAC7611;
- порт RS-232.

Модель процессора и часть моделей устройств уже были реализованы, остальные сделали за 15 дней трое разработчиков. После настройки модельных соединений были написаны и отлажены драйвера устройств. А затем и вся управляющая программа. Полученная программа содержала приблизительно 9800 строк на языке С и ассемблере, машинный код занимал 50 кбайт. На всю работу было потрачено 100 человеко-часов. То есть средняя производительность была 98 строк в час и 512 байт в час для системы из семи взаимодействующих компонентов.

Исследование производительности симуляции однопроцессорной системы

Исследование проводилось по следующей схеме:

- для каждого процессора написана программа, в которой в цикле выполняются наиболее часто используемые инструкции;
- экспериментальным путем подбирается ограничение по выполненным тактам таким образом, чтобы моделирование длилось 5–10 секунд;
- с помощью интерактивного тестирования замеряются временные характеристики;
- составляется таблица, результаты приведены в таблице 7. (Замеры производительности выполнялись на ПК Celeron (P-2) с частотой 1 ГГц и ОЗУ 256 Мб.)

Из таблицы видно, что для CISC-архитектуры исполнение инструкций моделируется в среднем быстрее в 4,7 раза, чем выполнение на реальном процессоре! Для RISC-архитектуры скорость моделирования немного медленнее, чем на реальном процессоре.

Исследование производительности симуляции мультипроцессорной гетерогенной системы

Производительность симуляции мультипроцессорной системы зависит не только от скорости исполнения инструкций моделями, но и от скорости выбора очередного процессора, инструкцию которого необходимо выполнить. Моделировалось взаимодействие четырех процессоров:

- AT90S2313 получает и отображает данные о расходе газа, электричества и воды;
- I8051 передает по протоколу I²C данные о расходе газа;
- MC68HC08 передает по протоколу I²C данные о расходе электричества;
- TMS370 передает по протоколу I²C данные о расходе воды.

Наращивание количества процессоров осуществлялось путем дублирования описанной системы из четырех процессоров. В качестве критерия эффективности взято суммарное количество инструкций, выполненное всеми процессорами за секунду времени. Замерена производительность для 4, 100, 200, ..., 900 и 1000 процессоров. Результаты приведены в таблице 8.

Таблица 8. Замеры производительности моделирования мультипроцессорной системы

1×4	1 032 828
25×4	768 947
50×4	536 991
75×4	453 332
100×4	409 245
125×4	372 123
150×4	360 567
175×4	351 419
200×4	345 737
225×4	339 485
250×4	337 268

Таблица 7. Замеры производительности моделей процессоров

	Atmel 90S2313	Intel 8051	Motorolla C68HC08	PIC17C4x	TMS370
Архитектура	RISC	CISC	CISC	RISC	CISC
Тактовая частота (МГц)	10	12	10	12	10
Модельное время (с)	5	67	20	6	30
Реальное время (с)	6	9	7	7	8
Отношение «Модельное/Реальное»	0,83	7,44	2,86	0,86	3,75
Инструкций в секунду	4917085	5247104	8567241	10828966	18312983

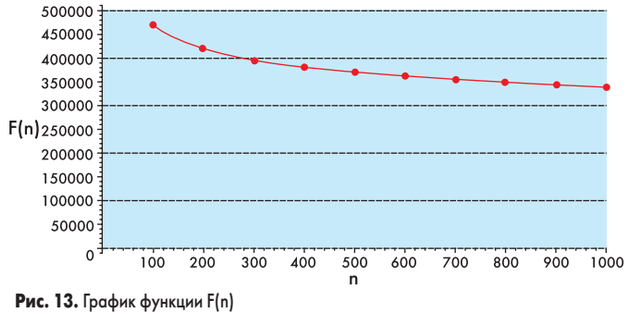

Рис. 13. График функции F(n)

Из таблицы видно, что с увеличением процессоров скорость моделирования падает. Сначала быстро, а потом все медленнее и медленнее. Это связано с затратами времени самой системы моделирования на выбор очередного процессора. Чем больше процессоров, тем больше этих затрат.

Количество выполненных инструкций в секунду может быть экстраполировано формулой (где n — количество симулируемых процессоров):

$$F(n) = 107 / (4{,}668747823 + \\ + 2{,}506703209 \times \log_2(n))$$

График этой функции приведен на рис. 13.

Заключение

Система WInter, предназначенная для отладки программного обеспечения мультипроцессорных систем, внедрена в учебный процесс Гомельского государственного университета, неоднократно применялась в реальных разработках, регулярно экспонируется на республиканских и международных выставках. В марте 2002 года в составе средств разработки встроенных цифровых систем от NewIT Research Labs (NewIT.gsu.unibel.by) демонстрировалась на крупнейшей международной выставке новых компьютерных технологий CeBIT 2002 (Ганновер, Германия) на стенде научно-технических достижений Республики Беларусь. В ноябре 2002 года на международной выставке «Перспективные технологии и системы» (Минск) комплекс средств автоматизации разработок встроенных цифровых систем (включающий WInter) от NewIT Research Labs был отмечен специальным дипломом Оргкомитета выставки. Уже известно, что разработки NewIT Research Labs будут представлены также и на CeBIT 2003.