Р. А. Зеленов, Ю. А. Степченков, В. Н. Волчек, Д. В. Хилько, А. Ю. Шнейдер, А. А. Прокофьев, Система капсульного программирования и отладки, *Системы и средства информ.*, 2010, том 20, выпуск 1, 24–30

УДК 004.4'233

СИСТЕМА КАПСУЛЬНОГО ПРОГРАММИРОВАНИЯ И ОТЛАДКИ[1]

Р. А. Зеленов, Ю. А. Степченков, В. Н. Волчек, Д. В. Хилько, А. Ю. Шнейдер, А. А. Прокофьев

Аннотация: Представлен результат разработки инструментального средства создания и отладки программ-капсул для рекуррентного обработчика сигналов. Показаны способы, позволяющие наделить данный продукт свойством гибкости для поддержки изменений в аппаратуре. Описаны способы построения и отладки капсул.

Ключевые слова: инструментальное средство отладки; программный симулятор; капсула; рекуррентность; VHDL

1. Введение

Использование инструментальных средств разработки и отладки программ позволяет значительно сократить срок создания программного обеспечения (ПО) и облегчить работу программиста. Особенно это касается низкоуровневых программ, которые должны выполняться непосредственно на микропроцессоре. В случае, когда архитектура вычислительного устройства, такого как рекуррентный обработчик сигналов (РОС) [1], еще только разрабатывается и может изменяться с течением времени, создание ПО сильно затрудняется. В такой ситуации весьма полезно иметь инструмент разработки, который может оперативно и корректно отображать изменения в архитектуре и создавать не только рабочие алгоритмы, но и отладочные примеры, используемые для проверки работоспособности нововведений в аппаратуре.

Изначально для программирования РОС было разработано программное средство ОПЕРА [2], которое позволяло отлаживать капсулы [3] в различных режимах и представлялось достаточно эффективным и универсальным. Но архитектура РОС постоянно дорабатывалась, вводились новые механизмы, которые

[1]Работа выполнена при частичной финансовой поддержке по Программе фундаментальных исследований ОНИТ РАН на 2010 г.

не были учтены в программе ОПЕРА. Таким образом, одним из ее недостатков оказалась затрудненность или даже невозможность модернизации для обеспечения поддержки новых механизмов работы процессора.

Для решения проблемы потребовалась разработка нового программного средства. После анализа всех необходимых требований было принято решение о создании программного обеспечения СКАТ (Система КАпсульного программирования и оТладки) [4], которое должно обеспечивать следующие возможности: создание капсул; пошаговое исполнение и отладка; выполнение числовой капсулы; удобная визуализация результатов исполнения; проверка возникновения исключительных ситуаций, обработка которых не заложена в аппаратуре. При этом программная среда должна обладать гибкостью, позволяющей оперативно подстраиваться под изменения в архитектуре РОС.

2. Программный симулятор

После анализа всех известных принципов построения отладочных средств и сопоставления их с многочисленными требованиями к СКАТ, главным из которых является необходимость исполнения алгоритмов на не реализованном, разрабатываемом устройстве, было решено использовать в качестве ядра системы программный симулятор [5], который в своем составе должен иметь виртуальный процессор. Виртуальный процессор — модель устройства, отображающая поведение реальной аппаратуры. Поскольку при разработке РОС используется язык VHDL (Very High Speed Integrated Circuits Hardware Description Language) [6], в качестве виртуального процессора весьма удобна VHDL-модель РОС. Это позволит не тратить время на создание поведенческой модели вычислительного устройства для СКАТ (с использованием, например, языка C++, как сделано во многих программных симуляторах) и будет в значительной мере способствовать гибкости программной системы с точки зрения поддержки нововведений в аппаратуре.

В результате сравнительного анализа существующих средств моделирования было принято решение использовать в качестве программного симулятора пакет ModelSim корпорации Mentor Graphics [7], что обусловлено целым рядом уникальных характеристик этого пакета [8]. Для того чтобы обеспечить взаимодействие между СКАТ и ModelSim, необходимо подготовить

ряд служебных файлов, содержащих информацию об объекте моделирования и ряд команд для запуска симуляции посредством CLI (Command Line Interface — интерфейс командной строки). Эти файлы формируются по результатам синтаксического разбора VHDL-модели РОС и выделения списка портов, сигналов и переменных, значение которых необходимо отслеживать в процессе выполнения программы.

При реализации СКАТ в виде программного симулятора с VHDL-моделью в качестве виртуального процессора появляется возможность отлаживать не только программную, но и аппаратную составляющую РОС.

3. Визуальный конструктор капсул

Рекуррентный процессор выполняет специальные программы — капсулы. Каждая капсула содержит набор операндов — самодостаточных данных, т. е. собственно данных и указаний по их обработке. Всего в архитектуру РОС заложено четыре типа операндов, каждый из которых имеет определенный набор полей и их значений. Для создания капсул в СКАТ реализован компонент, формирующий удобный графический интерфейс.

С течением времени, например, при модификации архитектуры процессора, формат операндов может изменяться. Соответственно, в СКАТ должен быть заложен механизм, обеспечивающий гибкость системы в плане формирования капсул. Он реализуется с применением технологии XML [9], с помощью которой описан формат всех возможных операндов капсулы. Это описание универсально, легко модифицируемо и используется в нескольких компонентах отладочной системы. Анализируя XML-представление формата операндов, СКАТ автоматически формирует графический интерфейс конструктора капсул, предоставляя пользователю всю нужную информацию — мнемонические значения полей операнда, информационные подсказки и др.

4. Отладка капсул

СКАТ предоставляет пользователю средства, необходимые для отладки капсулы, в том числе стандартные подходы: пошаговый режим выполнения, аналогичный решению в программе ОПЕРА; режим просмотра файла-отчета, полученного с по-

мощью ModelSim и содержащего полную информацию о состоянии РОС (значения внутренних регистров, состояние памяти и др.) на всем периоде работы. Файл-отчет с помощью ModelSim может быть визуализирован в виде временны́х диаграмм, что дает пользователю возможность отлаживать не только программную, но и аппаратную составляющую процессора в виде, привычном для разработчиков аппаратуры.

Многолетний опыт программирования алгоритмов под рекуррентный процессор поставил требование формирования графа выполнения капсулы, узловым элементом которого является операция, выполняемая на вычислительном ядре РОС. Это представление дает возможность легко отслеживать потоки данных в процессоре и, соответственно, способствует ускорению процесса отладки. Такая возможность не была представлена в ОПЕРА. Для этого был разработан алгоритм, согласно которому происходит анализ файла-отчета работы процессора, и выделяются элементы и значения, ключевые для построения графа. Было опробовано несколько способов построения визуального представления. Наиболее удобной и универсальной показала себя технология масштабируемой векторной графики (SVG) [10], с помощью которой строится графическая составляющая графа выполнения алгоритма. Для обеспечения гибкости в СКАТ заложен механизм, позволяющий модифицировать шаблоны отдельных компонентов формируемого графа.

В большинстве современных систем отладки предоставлены только средства просмотра (в том или ином виде) результата выполнения программы; анализ функционирования с точки зрения логики перекладывается на программиста. Таким образом, весьма затрудняется процесс отлова ошибок времени выполнения (исключительные ситуации) [11], которые не приводят к аварийному завершению программы, а влияют на правильность выходных результатов.

Спецификация РОС предусматривает большое число исключительных ситуаций, которые могут произойти во время выполнения капсулы. Естественно, возложить обработку их всех на сам процессор не представляется возможным, так как это потребовало бы огромных аппаратных затрат и привело бы к ухудшению временных характеристик устройства. Таким образом, возникло еще одно немаловажное требование к СКАТ: обрабатывать ошибки времени выполнения и предоставлять пользователю информацию о причине их возникновения. Для реализа-

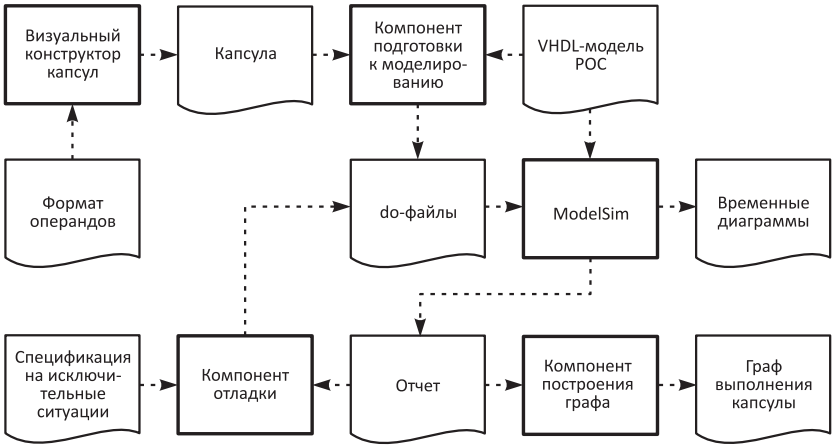

Рис. 1 Диаграмма взаимодействия компонентов СКАТ

ции этой задачи планируется разработать специальный формат, в котором были бы описаны исключительные ситуации. Эти данные подаются на вход компонента СКАТ, который, анализируя файл-отчет, полученный после выполнения капсулы, будет сопоставлять его данные и отслеживать появление моментов в программе, ошибочных с точки зрения логики. Затем пользователю предоставляется отчет, содержащий все обнаруженные исключительные ситуации, причины их возникновения и меры по устранению. Следуя принципу гибкости, добавлять или изменять обработку исключительных ситуаций можно будет практически «на лету», изменяя содержимое файла, в котором описаны все ошибки времени выполнения, предусмотренные спецификацией.

Обобщенная схема взаимодействия основных компонентов (обозначены прямоугольниками) СКАТ, описанных выше, представлена на рис. 1. Пунктирными линиями обозначены направления передачи данных.

5. Заключение

Среди средств отладки программ, предназначенных для выполнения на процессорах, СКАТ занимает особое место, и отнести его к тому или иному классу ПО затруднительно. Это

обусловлено целым рядом нестандартных подходов и решений, примененных при его разработке.

Во-первых, СКАТ в рамках архитектуры РОС обладает исключительной гибкостью практически во всех аспектах своей работы. VHDL-модель устройства выступает в качестве виртуального процессора, обеспечивая полное соответствие актуальной версии архитектуры РОС. Входные (например, формат операндов) и выходные (например, граф выполнения капсулы) данные, которые могут изменяться с течением времени, описаны в формате, позволяющем легко вносить изменения, и СКАТ, соответственно, обеспечивает их корректную обработку.

Во-вторых, СКАТ как система проектирования и отладки предоставляет обширные возможности, снижающие вероятность появления ошибок в программах. Визуальный конструктор капсул обеспечивает создание программ, верных с точки зрения синтаксиса языка, определенного в спецификации. Быстрому устранению логических ошибок способствуют различные варианты отображения хода выполнения капсулы, а также реализуемый в данный момент компонент СКАТ, который обрабатывает исключительные ситуации, не предусмотренные в аппаратуре.

В-третьих, применение ModelSim в качестве средства моделирования позволяет производить отладку не только программ, которые должны выполняться на процессоре, но и самого РОС. Это обеспечивается возможностями способа просмотра результатов моделирования, удобного для разработчиков аппаратуры, а также автоматизацией некоторых операций, необходимых для подачи тестовых воздействий на VHDL-модель устройства.

В ближайшем будущем работы по созданию СКАТ будут полностью завершены, и разработчики РОС получат в свое распоряжение мощный инструмент отладки программ-капсул.

Литература

1. *Степченков Ю. А., Петрухин В. С.* Особенности гибридного варианта реализации на ПЛИС рекуррентного обработчика сигналов // Системы и средства информатики: Доп. вып. — М.: ИПИ РАН, 2008. — С. 118–129.
2. *Морозов Н. В., Андреев Р. В.* Расширенная программная модель рекуррентного операционного устройства для цифрового сигнального процессора // Методы и средства разработки информационно-вычислительных систем и сетей (спец. вып.). — М.: Наука, 2004. — С. 4–11.

3. Исследование программируемости архитектурно-алгоритмических и схемотехнических проблем проектирования рекуррентных компьютеров (заключительный отчет). Шифр «ПАРСЕК». № г.р. 01.20.0412412. — М.: ИПИ РАН, 2006. 201 с.
4. *Зеленов Р. А., Степченков Ю. А., Волчек В. Н., Петрухин В. С., Прокофьев А. А., Хилько Д. В.* Система капсульного программирования и отладки (СКАТ). Свидетельство об официальной регистрации программы для ЭВМ № 2010610023 от 20.01.2010.
5. Программные симуляторы. URL: http://de.ifmo.ru/bk_netra/page.php?tutindex=25&index=72 (дата обращения 15.04.2010).
6. VHDL — обучающий портал. URL: http://www.bsuir.by/vhdl/ (дата обращения 15.04.2010).
7. ModelSim — advanced simulation and debugging. URL: http://model.com (дата обращения 15.04.2010).
8. *Зотов В.* ModelSim — система HDL-моделирования цифровых устройств. URL: http://www.compitech.ru/html.cgi/arhiv/02_06/stat_122.html (дата обращения 15.04.2010).
9. Extensible Markup Language. URL: http://www.w3.org/XML/ (дата обращения 15.04.2010).
10. Scalable Vector Graphics. URL: http://www.w3.org/Graphics/SVG/ (дата обращения 15.04.2010).
11. Ошибки времени выполнения. URL: http://www.realcoding.net/articles/oshibki-vremeni-vypolneniya.html (дата обращения 15.04.2010).