ИНСТРУМЕНТ АВТОМАТИЗАЦИИ РАЗРАБОТКИ ГЕНЕРАТОРОВ ТЕСТОВЫХ ПРОГРАММ ДЛЯ МИКРОПРОЦЕССОРОВ НА ОСНОВЕ ФОРМАЛЬНЫХ СПЕЦИФИКАЦИЙ

А. Д. Татарников
Институт системного программирования РАН

Аннотация

В докладе рассказывается об инструменте, позволяющем автоматизировать разработку генераторов тестовых программ для микропроцессоров. В основе работы инструмента лежит использование высокоуровневых формальных спецификаций в качестве источника знания об архитектуре тестируемого микропроцессора. Такой подход помогает сократить трудоемкость разработки тестовых программ и повысить качество тестирования.

Введение

Высокая сложность современных микропроцессоров и сжатые сроки выхода на рынок делают их верификацию нетривиальной задачей. Основной подход к функциональной верификации микропроцессоров на системном уровне - генерация тестовых программ и анализ результатов их выполнения. Несмотря на существование мощных средств автоматической генерации тестовых программ, тестирование занимает свыше 70% от общего объема трудозатрат на разработку микропроцессора [1]. Основная причина этого заключается в трудоемкости настройки существующих генераторов на создание тестов для микропроцессоров с новой архитектурой. В большинстве случаев генератор создается под конкретный микропроцессор и при переходе на новую архитектуру основную часть логики генерации приходится реализовывать заново. Одно из перспективных решений данной проблемы – автоматизированное построение генераторов тестовых программ для конкретной архитектуры на основе формальных спецификаций [2, 3]. При этом генератор состоит из двух частей: (1) ядро, которое реализует общие для всех микропроцессоров методы генерации, и (2) модель, которая содержит информацию о конкретном микропроцессоре и строится автоматически на основе предоставленных формальных спецификаций. Подход, использующий модели, уже применяется в коммерческих средствах генерации, таких как Genesys-Pro [4] и RAVEN [5]. Однако разработка модели остается трудоемкой задачей, требующей специальных навыков, которыми инженер-верификатор обычно не обладает. Использование высокоуровневых формальных спецификаций, схожих по формату с псевдокодом, применяемым в руководствах по архитектуре микропроцессоров для описания семантики команд, поможет значительно упростить эту задачу. Представленный подход реализован инструменте MicroTESK [3], разрабатываемом в ИСП РАН. Данный инструмент позволяет построить генераторы тестовых программ для широкого спектра микропроцессоров (RISC, CISC, VLIW, DSP) на основе формальных спецификаций их архитектуры на языке nML [6]. Созданные генераторы осуществляют генерацию на основе тестовых шаблонов, описывающих тестовые сценарии на абстрактном уровне, и позволяют создавать случайные, комбинаторные и направленные тесты. Сценарии для направленных тестов описываются в терминах тестовых ситуаций, «интересных» для тестирования событий, происходящих во время работы микропроцессора. Информация о тестовых ситуациях извлекается из формальных спецификаций при построении модели микропроцессора. При этом условия возникновения тестовых ситуаций задаются в виде ограничений, которые разрешаются в процессе генерации тестов.

Оставшаяся часть тезиса организована следующим образом. Раздел «Существующие подходы» дает обзор уже имеющихся методов и средств генерации тестов. Раздел «Формальные спецификации» описывает преимущества использования формальных спецификаций системы команд для описания конфигурации генератора тестов. Раздел «Архитектура инструмента» описывает архитектуру инструмента MicroTESK. В разделе «Заключение» резюмируются результаты исследования и его практическая значимость.

Существующие подходы

Верификация микропроцессоров на системном уровне осуществляется при помощи симуляции выполнения тестовых программ на проектной модели уровня регистровых передач (RTL, Register Transfer Level), которая создается инженерами-проектировщиками на языках описания аппаратуры (HDL, Hardware Description Language), таких как Verilog и VHDL, и служит основой для производства готовых интегральных схем. Существуют два подхода к проверке корректности поведения HDL-модели: (1) сравнение трасс выполнения с трассами, полученными в результате выполнения на эталонной модели, и (2) использование тестовых программ со встроенными проверками. В первом случае требуется эталонная модель, которая обычно представляет собой симулятор уровня команд и создается на языках высокого уровня. Во втором случае необходимо предсказывать состояние модели после выполнения теста. При автоматической генерации, такая возможность должна быть реализована в генераторе.

Автоматическая генерация является основным способом создания тестовых программ, т. к. ручная разработка из-за высокой трудоемкости применима только к проверке сложно формализуемых и маловероятных ситуаций. Методы автоматической генерации разделяются на следующие категории [2]: (1) случайная генерация; (2) комбинаторная генерация; (3) генерация на основе шаблонов. Подходы к генерации тестов эволюционировали от случайных к нацеленным на конкретные ситуации и основанным на формальных методах. Каждый подход имеет свою область применения и, как правило, они используются в комбинации.

Первые два метода являются наиболее простыми с точки зрения реализации. Как правило, компании-производители микропроцессоров разрабатывают случайные и комбинаторные генераторы для своих микропроцессоров на языках высокого уровня. Однако эти методы не могут гарантировать покрытия всех «интересных» для тестирования ситуаций и подобные инструменты крайне сложно адаптируются к использованию под другие архитектуры. Кроме того их разработка и поддержка требует трудозатрат и они, как правило, не включают эталонную модель, которую необходимо разрабатывать отдельно.

Наиболее известными коммерческими генераторами являются Genesys-Pro (IBM Research) [4], и RAVEN, (Obsidian Software, в настоящее время используется в ARM) [5]. Оба генератора позволяют генерировать случайные и направленные тесты на основе тестовых шаблонов и состоят из архитектурно независимого ядра и архитектурно зависимой модели микропроцессора. Архитектурно зависимое знание задается в виде декларативного описания, а для предсказания состояния процессора используется сторонняя эталонная модель (симулятор уровня команд). Ввиду закрытости этих

инструментов, доступно мало информации об особенностях создания моделей для них, но есть основания полагать, что это является достаточно трудоемкой задачей.

Текущее положение дел создает мотивацию для поиска решений, позволяющих снизить трудоемкость создания генераторов тестовых программ, которые могли бы интегрировать различные методы генерации.

Формальные спецификации

Упрощение создания архитектурно зависимой части генератора и симулятора уровня команд, используемого в качестве эталонной модели, может быть достигнуто путем использования формальных спецификаций на языках описания архитектуры (ADL, Architecture Description Language) [7]. Из данного семейства языков для этой цели лучше всего подходит язык nML [6, 7], разработанный в начале 1990-х годов в Берлинском техническом университете и изначально предназначенный для создания симуляторов микропроцессоров и настраиваемых компиляторов. Он позволяет специфицировать синтаксис и семантику системы команд, абстрагируясь от деталей реализации. Спецификация включает описание следующих сущностей: (1) элементов хранения данных (память, регистры), определяющих состояние микропроцессора; (2) режимов адресации, реализующих абстракцию доступа к элементам хранения данных; (3) команд, выполнение которых изменяет состояние микропроцессора. В Примере 1 приведен код команды add микропроцессора MIPS.

```
op add (rd: R, rs: R, rt: R)
  action = {
    temp = rs<31>::rs + rt<31>::rt;
    if temp<32> != temp<31> then
      exception("IntegerOverflow");
    else
      rd = temp<31..0>;
    endif;
  }
  syntax = format("add %s, %s, %s",
    rd.syntax, rs.syntax, rt.syntax)
```

Пример 1. Команда add на языке nML.

Семантика команды описана в атрибуте action и может быть использована для построения симулятора и получение информации о тестовых ситуациях. Атрибут syntax описывает ассемблерный формат команды и может быть использован для генерации кода тестовых программ. Как можно заметить, конструкции языка схожи с конструкциями псевдоязыка, используемого в руководствах по архитектуре, что делает создание спецификации на языке nML тривиальной задачей.

Архитектура инструмента

Инструмент MicroTESK создает генератор тестовых программ для микропроцессора на основе формальных спецификаций его архитектуры. Генератор включает в себя модель микропроцессора, которая содержит информацию о системе команд микропроцессора и связанных с ней тестовых ситуациях (модель покрытия). Кроме того модель реализует симулятор уровня команд, служащий эталонной моделью. Другая важная часть генератора - архитектурно независимое ядро, составленное из переиспользуемых компонентов, которое отвечает за обработку тестовых шаблонов, составление тестовых последовательностей, генерацию тестовых данных, симуляцию тестов на эталонной модели и вывод полученных тестовых программ.

Таким образом, на MicroTESK возложены две основные задачи: (1) создание модели путем трансляции nML-спецификаций и (2) предоставление среды для генерации тестов для созданной модели. Концептуальная схема инструмента MicroTESK показана на Рисунке 1.

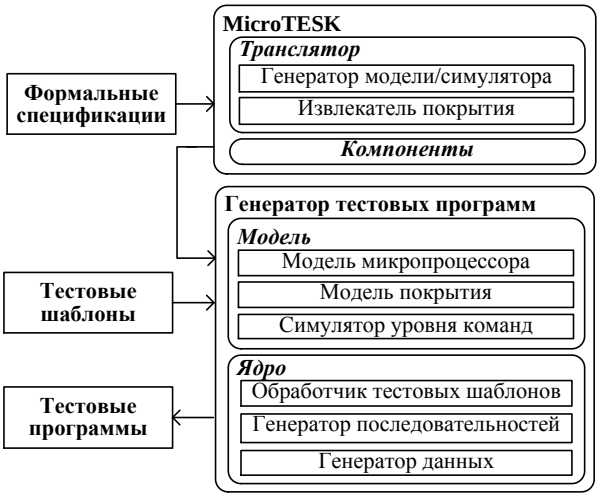

Рисунок 1. Концептуальная схема инструмента.

Транслятор генерирует архитектурно зависимые части генератора на основе извлеченной информации и компонентов, реализующих основные сущности. Генерация осуществляется по шаблонам, описанным в терминах модели микропроцессора. Модель покрытия содержит ограничения, описывающие условия возникновения тестовых ситуаций. Поддерживаемые методы генерации реализованы в компонентах, набор которых можно расширять.

Заключение

Возрастающая по закону Мура сложность современных микропроцессоров требует все более совершенных инструментов верификации. Одно из ключевых направлений — повышение точности и производительности тестирования. Подход, предложенный в данной работе и реализованный в инструменте MicroTESK, помогает минимизировать трудозатраты, связанные с разработкой и поддержкой генераторов тестовых программ для микропроцессоров. В настоящее время разработан прототип инструмента, который был опробован на архитектурах MIPS и ARM. В дальнейших планах стоит использование спецификаций элементов микроархитектуры (система управления памятью, конвейер команд) для создания тестов.

Список литературы

1. А.С. Камкин, А.М. Коцыняк, С.А. Смолов, А.А. Сортов, А.Д. Татарников, М.М. Чупилко, Средства функциональной верификации микропроцессоров, Труды ИСП РАН, том 26, выпуск 1, 2014, с. 149-200.
2. А.С. Камкин, Т.И. Сергеева, С.А. Смолов, А.Д. Татарников, М.М. Чупилко, Расширяемая среда генерации тестовых программ для микропроцессоров, Программирование №1, 2014, с. 3-14.
3. A. Kamkin, A. Tatarnikov. MicroTESK: An ADL-Based Reconfigurable Test Program Generator for Microprocessors. Spring/Summer Young Researchers' Colloquium on Software Engineering (SYRCoSE), 2012. p. 64-69.
4. A. Adir, E. Almog, L. Fournier, E. Marcus, M. Rimon, M. Vinov and A. Ziv. Genesys-Pro: Innovations in Test Program Generation for Functional Processor Verification. IEEE Design & Test of Computers, Volume 21, Issue 2, 2004, p. 84–93.
5. RAVEN test program generator — http://www.slideshare.net/DVClub/introducing-obsidian-software-and-ravengcs-for-powerpc
6. M. Freericks. The nML Machine Description Formalism. Techical Report. TU Berlin, FB20, Bericht, 1991/15. 47 p.
7. Mishra P, Dutt N (Eds.). Processor Description Languages. Systems on Silicon. Morgan Kaufmann, 2008. 432 p.