МОДЕЛЬ ТЕСТИРОВАНИЯ ПРОГРАММНЫХ И АППАРАТНЫХ КОМПОНЕНТОВ ЦИФРОВЫХ СИСТЕМ НА КРИСТАЛЛАХ

Хаханов В.И., Баранник В.В., Краснояружская К.Ш., Каминская М.А.
Харьковский национальный университет радиоэлектроники
Украина, 61166, Харьков, пр. Ленина 14
Тел., факс: (057) 702-13-26, E-mail: {hahanov,maryna4329}@kture.kharkov.ua

Представлены модели и методы тестопригодного проектирования программных продуктов, основанные на модификациях технологических и технических решений синтеза и анализа цифровых систем на кристаллах, представленных моделями программных и аппаратных компонентов в виде ориентированных графов регистровых передач и управления вычислительным процессом, которые используются для тестопригодного анализа и синтеза тестов.

1. Актуальность, цель и задачи исследования

Инициирующим ядром появления новых технологий тестирования и верификации в программной и компьютерной инженерии следует считать силиконовый кристалл, являющийся основой для создания вычислительных и/или коммуникационных устройств. Технологические решения, выдержавшие испытания временем в микроэлектронике, далее захватываются и адаптируются в макроэлектронике, представленной компьютерными системами и сетями. Вот некоторые исторические факты, связанные с преемственностью развития технологических инноваций.

1. Стандарты граничного сканирования [1] на уровне платы и кристалла привели к появлению механизма ассерций для тестирования и верификации программных продуктов. 2. Средства анализа тестопригодности [2] цифровых структур можно адаптировать для оценки программного кода в целях определения критических мест и последующего улучшения программного продукта. 3. Технологии анализа качества покрытия наперед заданных неисправностей тестовыми блоками следует использовать для создания таблицы покрытия дефектов программных продуктов в целях оценки достоверности тестов и установления диагноза. 4. Графовые модели регистровых передач [3-5] используются при тестировании программных продуктов, которые приводятся к более технологичной форме. 5. Разделение автомата на управляющую и операционную части также применяется для упрощения процесса верификации программного кода на основе предварительного синтеза графов. 6. Применение технологии Electronic System Level (ESL) [7] в программировании дает возможность использовать программные компоненты готовых функциональностей из базовых библиотек, разработанных для создания новых программных продуктов. 7. Понятие testbench [8], используемое для тестирования и верификации аппаратных проектов с помощью HDL-компиляторов, появляется и в программных продуктах, реализованных на уровне языков C++ и выше. 8. Платформенно-ориентированный синтез testbench с использованием существующих библиотек тестов (ALINT) для компонентов – стандартизованных функциональностей F-IP SoC под управлением GUI следует использовать для генерирования тестов программных модулей на основе наработанных библиотек ведущих компаний. 9. Стандартные решения сервисного обслуживания F-IP в рамках I-IP можно использовать для встроенного тестирования компонентов программной системы, и для восстановления работоспособности дефектного модуля. 10. Обеспечение планарности (двумерности) в структуре взаимосвязанных функциональных компонентов (IP-cores) создаваемого программного продукта ориентировано на использование мультиядерных архитектур для технологичного распараллеливания вычислительных процессов [9]. 11. Создание адресного пространства для функциональностей SoC, реализованных как в аппаратном,

так и в программном исполнении [6,10], предоставляет цифровой системе замечательное свойство самовосстановления работоспособности программных и аппаратных компонентов средствами I-IP. 12. Решение проблемы автономного внутрикристального самотестирования, самодиагностирования и самовосстановления с применением внешних средств [10]. 13. Сближение и взаимопроникновение технологий приводит к изоморфным методам проектирования, тестирования и верификации по отношению к программным и аппаратным комплексам, что по существу является закономерным процессом ассимиляции прогрессивных концепций. Тому способствует факт, что наиболее важные параметры жизненного цикла изделия, такие как time-to-market и yield становятся соизмеримыми по времени и выходу годной продукции.

Стоимость верификации программно-аппаратных продуктов на основе ASIC, IP-core, SoC составляет 70% от общих затрат проектирования. Аналогичная оценка, около 80%, определяет размерность testbench-кода от общей длины формального описания проекта. Введение в проект программной избыточности – механизма ассерций [11] – позволяет выполнять анализ основных специфицированных условий в процессе моделирования проекта и диагностировать ошибки в случае их обнаружения на ранних стадиях проектирования программы (C++) или аппаратуры (HDL).

Цель исследования – разработка эффективных моделей тестопригодного проектирования программных продуктов путем адаптации hardware design технологий и приведения программных структур к существующим стандартам и шаблонам тестирования и верификации. Задачи исследования: 1) разработать модель процесса верификации и валидации software; 2) Разработать модель представления программного продукта, ориентированную на тестопригодный анализ и синтез тестов для операционной и управляющей частей software.

2. Тестирование, диагностирование и ремонт F-IP

Стандарт IEEE 1500 SECT следует рассматривать как эффективный компонент инфраструктуры сервисного обслуживания цифровых систем на кристаллах. Его основное назначение – тестирование всех функциональностей F-IP, а также гальванических связей между ними. Следующим шагом в эволюции стандарта на пути создания ремонтопригодных кристаллов является разработка компонентов I-IP с сервисными функциями диагностирования и восстановления работоспособности SoC, которые в совокупности с модулем тестирования являются рыночно привлекательными: $I = \{T, D, R\}$. Процедуры диагностирования и ремонта пока не регламентируются стандартами тестопригодного проектирования, ввиду сложности и неоднозначности универсального решения данной проблемы для различных типов вычислительных устройств. Для нерегулярных или уникальных структур решение всех трех задач основывается на априорной избыточности – диверсификации функциональностей всех компонентов, входящих в состав SoC. Только при таких условиях можно говорить о внутрикристалльном ремонте отказавшего элемента. Что касается регулярных структур, имеющих избыточность по определению, таких как мульти- или матричные процессоры, то здесь можно предложить структуру контроллера, который будет совмещать выполнение все трех упомянутых функций с помощью стандарта граничного сканирования:

$$I = \{T, F, S, D, R\}, T = \{T^1, T^2, ..., T^i, ..., T^n\}, F = \{F^1, F^2, ..., F^i, ..., F^n\};$$

$$S = \{S^1, S^2, ..., S^i, ..., S^n\}, D = f(T, F, S) = F^D \in F, R = g(D, F) = (F^R \subseteq F) \& (F^R \cup F^D = F).$$

Здесь первые три компонента (T,F,S) модели есть тесты для функциональностей; компоненты, представляющие функции, а также ячейки регистра граничного сканирования для идентификации технического состояния функциональностей. Следующие два представлены функциями для выполнения диагностирования и ремонта

SoC. Первая функция (D) определяет совокупность неисправных компонентов, вычисляемых на основе вектора экспериментальной проверки S и теста, покрывающего все дефекты функционалов, что записывается в форме таблицы неисправностей. Вторая функция (R) формулирует правила уменьшения мощности компонентов путем исключения из адресации дефектных элементов и образования нового работоспособного подмножества F-IP SoC для его использования по назначению.

Весьма актуальным представляется создание формальной модели программного продукта, к которой можно применить технологии от CAD и EDA в целях применения формальных методов синтеза тестов, оценки их качества покрытия неисправностей, определения тестопригодности для последующей модернизации структуры программных компонентов. Для решения упомянутых задач предлагается использовать автоматную модель, представленную двумя ориентированными графами:

$$\begin{cases} M = (M^{OR}, M^{CG}), M^{OR} = \{R, I\}; R = \{R_1, R_2, ..., R_i, ..., R_n\}, I = \{I_1, I_2, ..., I_j, ..., I_m\}; \\ R_i = f(R_k, I_j); M^{CG} = \{S, T\}; R = \{S_1, S_2, ..., S_i, ..., S_p\}, T = \{T_1, T_2, ..., T_j, ..., T_q\}; S_i = f(S_k, T_j). \end{cases}$$

Здесь M^{OR} – граф регистровых передач С.Г Шаршунова, имеющий множество вершин R, задающих все компоненты памяти, используемые в программе, и дуг, отмеченных инструкциями I, которые активизируют передачу информации между вершинами. Компонент M^{CG} представляется собой содержательный граф управляющего автомата, определенный на множестве вершин S, которые соединены ориентированными дугами T, отмеченными условиями переходов. Примеры задания графов регистровых передач и управления представлены на рис.1.

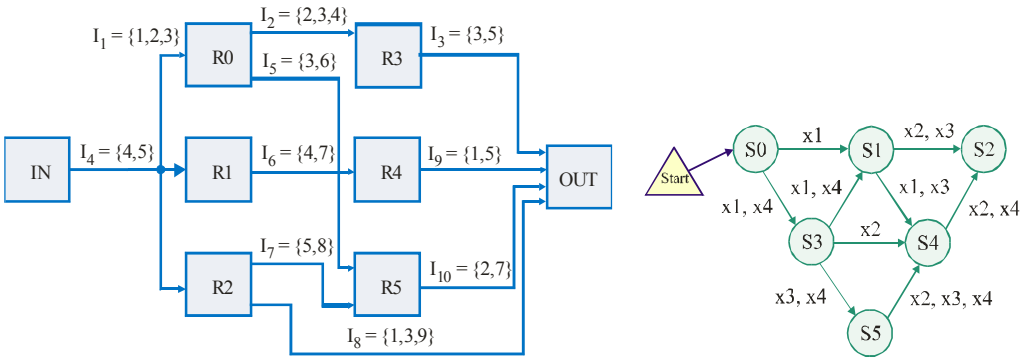

Рис. 1. Граф регистровых передач и управления

Преимущества графовых моделей заключаются не только в структурном отображении взаимодействия функционалов, но и в возможности применять методы анализа тестопригодности, поскольку ориентированные графовые модели имеют выраженные направления информационных потоков, входные и выходные вершины. Исходя их опыта оценивания тестопригодности цифровых схем, можно предложить следующую метрику анализа управляемости и наблюдаемости предложенных структур:

$$G = \{R, I\};\ R = \{R_1, R_2, ..., R_n\},\ I = \{I_1, I_2, ..., I_m\};\ I_{ij} \in I_i \approx (R_p R_q);\ C(R_x) = 1;\ O(R_y) = 1;$$

$$C(R_q) = \frac{1}{k} \times \sum_{i=1}^{k} \left[\frac{1}{m} \times \left| \bigcup_j I_{ij} \in (R_p R_q) \right| \times C(R_p) \right];\ O(R_p) = \frac{1}{k} \times \sum_{i=1}^{k} \left[\frac{1}{m} \times \left| \bigcup_j I_{ij} \in (R_p R_q) \right| \times O(R_q) \right].$$

Здесь модель программного (аппаратного) модуля представлена графом $G = \{R, I\}$, состоящим из вершин (регистров) и дуг (инструкций). Каждая дуга графа отмечена операцией $I_{ij} \in I_i \approx (R_p R_q)$, формирующей подмножество команд, принадлежащее дуге $(R_p R_q)$. Управляемость вершины $C(R_q)$ зависит от управляемости предшествующей

вершины $C(R_p)$, а также от приведенной аддитивной мощности множества команд, активизирующих k дуг, входящих в анализируемую вершину $C(R_q)$. Здесь каждая дуга графа содержит m операций, инициирующих передачу информации в (R_pR_q). По аналогии формулируется критерий оценивания наблюдаемости $O(R_p)$. Преимуществом введенных моделей и критериев оценивания управляемости и наблюдаемости является их универсальность, ориентированная на выполнение прямой и обратной импликации на графе, а также их инвариантность по отношению к тестопригодному анализу и синтезу тестов программных и аппаратных компонентов.

3. Выводы

Рассмотрены инновационные технологии тестопригодного проектирования программных и аппаратных продуктов, ориентированные на эффективную разработку тестов и верификацию компонентов цифровых систем на кристаллах.

Научная новизна: 1. Показаны основные направления использования технологий тестопригодного проектирования цифровых систем на кристаллах в задачах тестирования и верификации программных продуктов. 2. Представлена универсальная модель программного и аппаратного компонента в виде ориентированного графа регистровых передач и управления, на котором можно решать задачи тестопригодного проектирования, синтеза и анализа тестов. 3. Практическая значимость предложенных методик и моделей заключается в высокой заинтересованности софтверных компаний в инновационных решениях проблемы эффективного тестирования и верификации программных продуктов, предложенных выше.

Литература:

1. Francisco DaSilva, Yervant Zorian, Lee Whetsel, Karim Arabi, Rohit Kapur (2003) Overview of the IEEE P1500 Standard // ITC International Test Conference. 988–997.

2. Abramovici M., Breuer M.A. and Friedman A.D. (1998) Digital System Testing and Testable Design, Computer Science Press. 652.

3. Thatte S.M., Abraham J.A. Test generation for microprocessors. IEEE Trans. Comput. 1980.- C-29. No 6. P.429-441.

4. Шаршунов С.Г. Построение тестов микропроцессоров. 1. Общая модель. Проверка обработки данных//Автоматика и телемеханика. 1985. №11. С.145-155.

5. Чипулис В.П., Шаршунов С.Г. Построение тестов микропроцессоров. 2. Проверка хранения и передачи данных // Автоматика и телемеханика. 1986. №1. С.139-150.

6. Zorian Yervant, Gizopoulos Dmytris (2004) Gest editors' introduction: Design for Yield and reliability // IEEE Design & Test of Computers. 177-182.

7. Frank Ghenassia. Transaction Level Modeling with SystemC. TLM Concepts and Applications for Embedded Systems.– Published by Springer.– 2005.– 282 p.

8. Bergeron, Janick. Writing testbenches: functional verification of HDL models.– Boston: Kluwer Academic Publishers, 2001.– 354 p.

9. Shameem Akhter, Jason Roberts. Multi-Core Programming. Intel Press. 2006. 270 p.

10. Zorian Yervant, Shoukourian Samvel (2003) Embedded-Memory Test and Repair: Infrastructure IP for SoC Yield. IEEE Design and Test of Computers. 58-66

11. Foster H., Krolnik A., Lacey D., Assertions-based Design. Kluwer Academic Publishers. 2003. 392p.