УДК 681.3
DOI: 10.17586/0021-3454-2015-58-3-197-202

МЕТОД ВСТРОЕННОЙ ДИНАМИЧЕСКОЙ АКТУАЛИЗАЦИИ ФУНКЦИОНАЛЬНЫХ МОДЕЛЕЙ СИСТЕМ НА КРИСТАЛЛЕ

С. В. Быковский

Университет ИТМО, 197101, Санкт-Петербург, Россия
E-mail: bsv.serg@gmail.com

Представлен оригинальный метод генерации формальной модели реального функционирования системы на кристалле в процессе натурных испытаний или эксплуатации. Метод предусматривает реализацию на базе встроенных средств системы на кристалле. По сравнению с альтернативными решениями существенно снижены требования к объему встроенной инструментальной памяти.

Ключевые слова: *верификация, функциональная модель, сбой, система на кристалле, конечный автомат, структура Крипке.*

Введение. При проектировании систем на кристалле (далее — система, СнК) компьютерное моделирование позволяет обнаружить и исправить порядка 53 % ошибок [1], тогда как значительная часть ошибок проявляется только при натурных испытаниях и в процессе эксплуатации.

В связи с этим актуальной задачей является разработка встраиваемых механизмов мониторинга и верификации поведения системы, рассчитанных на долговременную автономную работу в рамках ограниченных вычислительных ресурсов систем на кристалле.

Существующие методы встроенной верификации с использованием журнала событий [2] не позволяют вести наблюдение за системой на протяжении длительного времени, что объясняется прямой зависимостью размера журнала событий от времени работы системы при ограниченном объеме встроенной памяти.

Другой метод — проверка системы с использованием мониторов утверждений (assertion-based verification) [3] — позволяет фиксировать лишь факт некорректного поведения системы, но не позволяет устанавливать последовательность событий, вызвавших ее сбой.

Для решения задачи уменьшения размера данных наблюдения с сохранением возможности анализа причин выявленных сбоев перспективным направлением является создание встроенных мониторов, способных во время функционирования системы сохранять только информацию об ее ошибочном поведении, не накапливая протоколов нормальной работы. Это возможно при описании поведения системы с помощью формальных моделей представления вычислительного процесса, таких как сети процессов, конечные автоматы, сети Петри и др. [4].

В общем виде проблема построения формальной модели на основе эмпирических данных изучается в области искусственного интеллекта при решении задачи индуктивного вывода [5]. Однако в области встроенной верификации задача индуктивного вывода не получила должного внимания.

В настоящей статье на основе формируемой методологической базы оригинального механизма встроенной верификации системы на кристалле предложен метод динамической актуализации функциональной модели системы, алгоритм работы которой описывается в виде детерминированного конечного автомата (Deterministic Finite Automaton — DFA).

Эталонная функциональная модель системы. Предлагаемый метод динамической актуализации (уточнения по факту) функциональной модели системы рассмотрим на примере верификации взаимодействия нескольких параллельных процессов в информационно-измерительной системе (ИИС). Здесь следует отметить, что функциональность рассматриваемой ИИС

не является жестко привязанной к платформе системы на кристалле, тем не менее метод динамической актуализации ориентирован на встроенные средства СнК, а именно — разработан с учетом ограничений по ресурсам памяти.

В процессе верификации осуществляется проверка соответствия реального поведения системы ожидаемому, специфицированному в виде эталонной функциональной модели (далее — эталонная модель).

Рассматриваемая ИИС состоит из трех параллельно функционирующих узлов, реализующих независимые процессы: коммуникационного узла (процесс *P1*), устройства управления (процесс *P2*) и обрабатывающего узла (процесс *P3*). Сеть процессов ИИС представлена на рис. 1. Для упрощения восприятия каналы взаимодействия с окружением ИИС не показаны.

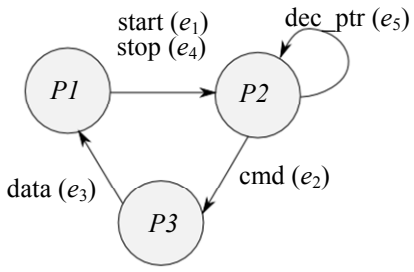

Рис. 1

Во время работы ИИС узел *P1* принимает из внешнего окружения (персональный компьютер оператора) и ретранслирует на узел *P2* команды “start” (событие e_1) и “stop” (событие e_4). По команде “start” устройство управления *P2* начинает формировать последовательность команд (событие e_2) к обрабатывающему узлу *P3*, который „опрашивает“ внешние измерительные устройства (датчики), получает данные измерений, обрабатывает их и передает результаты узлу *P1* (событие e_3). В процессе сбора данных и их обработки извне может поступить команда “stop” (событие e_4). Если команда “stop” поступила после новой команды от устройства *P2* (событие e_2), то узел *P1* проигнорирует данные (не будет осуществлена обработка события e_3), а следующий рабочий цикл начнется с повторного выполнения предыдущей команды (событие e_2) к узлу *P3*. Для этого *P2* после приема команды “stop” модифицирует свой внутренний указатель команд (событие e_5).

Обозначим все множество контролируемых событий как $E = \{e_1, e_2, e_3, e_4, e_5\}$ и представим эталонную модель в виде графа DFA (рис. 2).

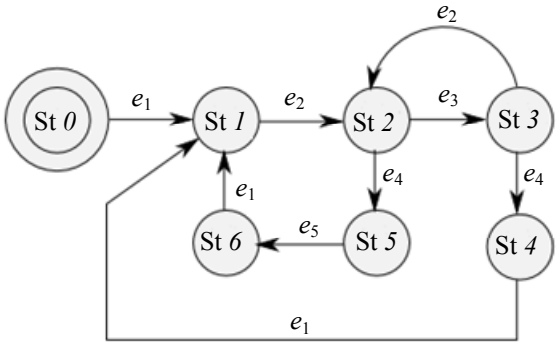

Рис. 2

Метод динамической актуализации функциональной модели. В основу метода положен процесс наблюдения за последовательностями событий (допустимыми/недопустимыми), а не за состояниями системы (запрещенными/неизвестными). Для уменьшения сложности алгоритма процесс наблюдения ограничен допустимыми парами событий, следующими друг за другом (цепочками):

$$c_k : e_i \to e_j ,$$

где c_k — идентификатор цепочки из двух событий — e_i и e_j, $e_i, e_j \in E$.

Все цепочки образуют множество $C = \{c_1, ..., c_k, ..., c_N\}$, где N — количество цепочек, допустимых в эталонной модели. Определим три действия, связанные с проверкой цепочек.

Каждая цепочка (событий) может быть открыта для наблюдения (c_k), закрыта (c_k^+) и удалена (c_k^-). Множество открытых для наблюдения цепочек обозначим как A. При возникновении первого события цепочки из множества C цепочка открывается для наблюдения (добавляется к множеству A). При возникновении второго события в ранее открытой цепочке цепочка закрывается (c_k^+) и исключается из множества A. Если при этом в множестве A остаются цепочки, в которых совпадают первые события, то эти цепочки также удаляются из множества A (c_k^-). Это верно для состояний, из которых существуют переходы в несколько состояний (например, St *2* на рис. 2).

Формально актуализированная модель определяется конечным множеством состояний S, начальным состоянием $s_0 \in S$, множеством допустимых цепочек событий C, множеством действий $R = \{c_k,\ c_k^+,\ c_k^-\}$ над цепочками из множества C, функциями пометок состояний $L: S \to 2^{R \times C}$.

Обратимся вновь к примеру ИИС: множество допустимых цепочек C для эталонной модели ИИС представлено выражениями (см. рис. 1, 2)

$$c_1 : e_1 \to e_2, c_2 : e_2 \to e_3,\ \ c_3 : e_2 \to e_4, c_4 : e_3 \to e_2,$$
$$c_5 : e_3 \to e_4, c_6 : e_4 \to e_5,\ \ c_7 : e_5 \to e_1, c_8 : e_4 \to e_1.$$

Допустим, что при разработке модели не учтено наступление следующих ситуаций: 1) команда “stop” поступает сразу после команды “start”; 2) время обработки данных в узле *P3* превысит интервал между командами, поступающими от узла *P2* (событие e_2). Указанные ситуации не учтены в эталонной модели, однако, используя метод динамической актуализации, их можно выявить на этапе испытаний.

Актуализированная функциональная модель формируется в виде структуры Крипке [6], состояния которой размечаются фактами выполнения действий с цепочками. Таким образом, полученная структура Крипке является детерминированным конечным автоматом (DFA), как и эталонная модель.

Актуализированная функциональная модель ИИС представлена на рис. 3. На графе сплошными дугами показаны переходы, допустимые эталонной моделью (см. рис. 2), а пунктирные дуги фиксируют отклонения от эталонной модели. Для наглядности дуги помечены событиями, которые инициируют переходы между состояниями актуализированной DFA-модели.

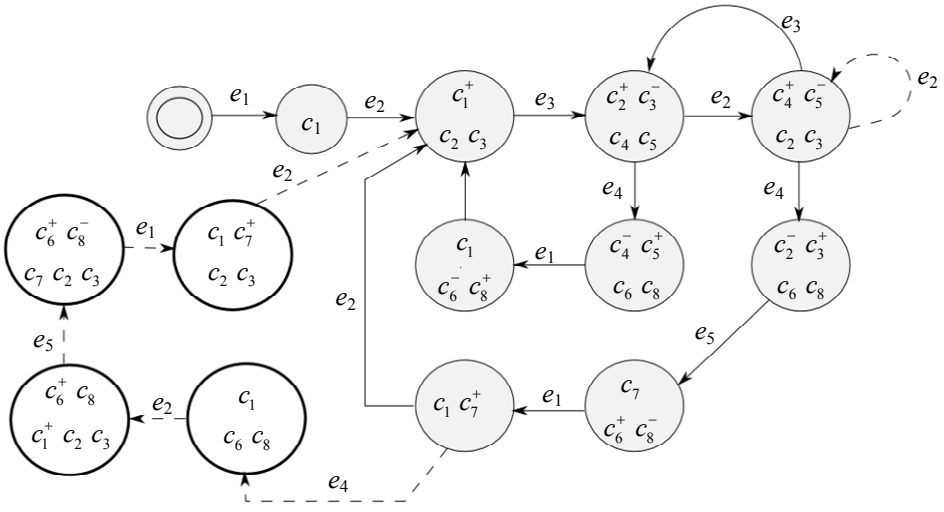

Рис. 3

Модель формируется динамически встроенными в СнК специальными блоками — мониторами — посредством фиксации событий E и выполнения действий R над цепочками из

множества C. В установившемся режиме размер актуализированного графа является фиксированным и увеличивается только при обнаружении факта некорректного (не соответствующего эталонной модели) поведения системы.

Посредством удаления из графа актуализированной модели допустимых состояний можно выявить пути некорректного поведения системы: 1) $e_1 \to e_4 \to e_2 \to e_5 \to e_1 \to e_2 \to \ldots$ 2) $e_2 \to e_2 \to \ldots$ Таким образом, в течение одного цикла испытаний была сформирована актуализированная модель, анализ которой позволил обнаружить сразу несколько сбоев; при этом если с каждой дугой графа связать счетчик переходов, то можно получить информацию о частоте возникающих сбоев.

В рассмотренном примере не учитываются временные ограничения на последовательность событий и переходов — для этого требуется расширить правила спецификации цепочек, как показано в работе [7].

Следует также отметить, что в сложных системах переходы между разными состояниями эталонной модели могут описываться с помощью одинаковых цепочек. Так, переходы через состояния St *3* и St *6* модели, представленной на рис. 4, *а*, описываются одинаковой цепочкой $c_5 : e_3 \to e_4$. Определение цепочек, состоящих только из двух соседних событий (цепочки $c_1 — c_7$), может привести к построению некорректной актуализированной модели. На рис. 4, *б* показан актуализированный граф, где объединены состояния, обозначающие два различных пути эталонной модели. Для предотвращения этого необходимо определить дополнительные цепочки-маркеры (c_8, c_9), тогда актуализированная модель примет вид, показанный на рис. 4, *в*. Определение достаточного множества допустимых цепочек C может быть автоматизировано средствами САПР.

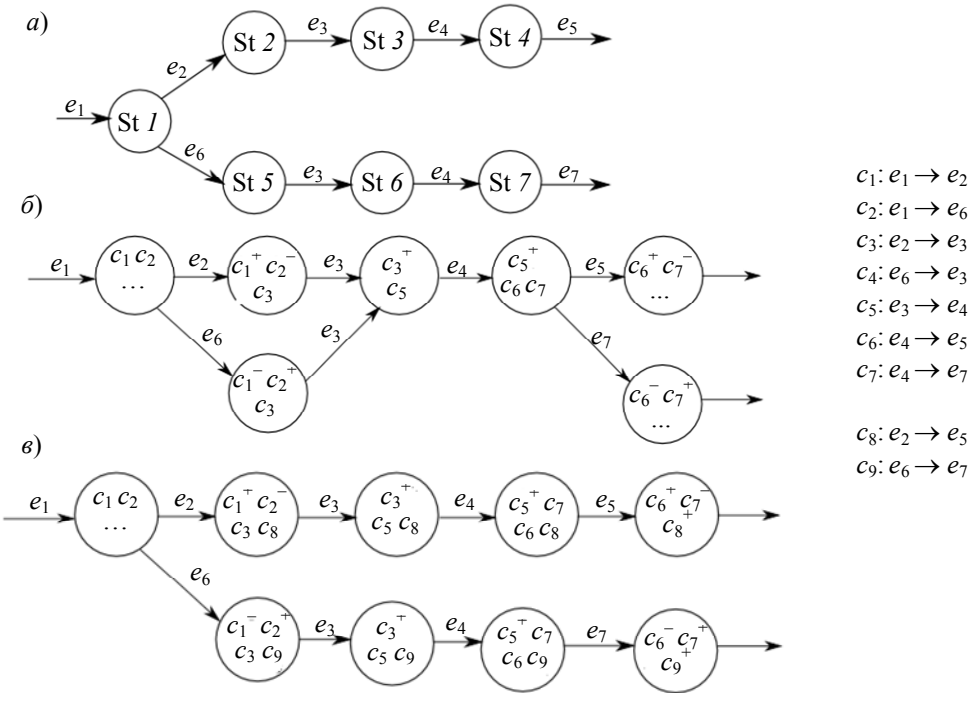

Рис. 4

Граф актуализированной модели является входными данными для задачи верификации. Проверка актуализированной модели на соответствие эталонной модели (функциональная верификация СнК) осуществляется средствами, встроенными или внешними по отношению к верифицируемой системе.

Проблема реализации метода связана со значительным временем обработки одного события. Например, в процессе натурных испытаний рассмотренной ИИС на ПЛИС обработка

одного события заняла 4 мс при тактовой частоте 50 МГц и среднем интервале между событиями 100 мс. Таким образом, метод не может быть использован для мониторинга „быстрых“ событий, связанных с изменением низкоуровневых сигналов, но применим для наблюдения за событиями со значительным интервалом следования, например коммуникационными транзакциями.

Выводы. Рассмотрены преимущества подхода к встроенной верификации системы на кристалле с использованием динамической актуализации функциональной модели, который, по сравнению с альтернативными методами, позволяет:

— проводить натурную верификацию системы на длительных интервалах времени (дни, недели, месяцы), что дает возможность выявлять плавающие, случайные и накапливающиеся ошибки, а также ошибки в результате непрогнозируемых внешних воздействий и т.п.;

— фиксировать последовательность событий, приводящих к сбою системы.

Предложенный метод — только часть методологии встроенной верификации, поэтапное развитие которой является перспективной задачей для будущих исследований.

СПИСОК ЛИТЕРАТУРЫ

1. *Wagner I., Bertacco V.* Post-Silicon and Runtime Verification for Modern Processors. Springer, 2011. P. 224.
2. *Metz E., Lencevicius R., Gonzalez T.* Performance data collection using a hybrid approach // Proc. of the 10th Europ. Software Engineering Conf. and 13th ACM SIGSOFT Intern. Symp. on Foundations of Software Engineering, Lisbon, Portugal, 5—9 Sept. 2005. P. 126—35.
3. *Zilic Z., Boule M.* Generating Hardware Assertion Checkers: for Hardware Verification, Emulation, Post-Fabrication Debugging and On-Line Monitoring. Springer, 2010. P. 280.
4. *Lee E. A., Sangiovanni-Vincentelli A.* Comparing models of computation // IEEE Computer Society. 1997. P. 234—241.
5. *Потапов А. С.* Распознавание образов и машинное восприятие: Общий подход на основе принципа минимальной длины описания. СПб: Политехника, 2007. 548 с.
6. *Карпов Ю. Г.* MODEL CHECKING. Верификация параллельных и распределенных программных систем. СПб: БХВ-Петербург, 2010. 560 с.
7. *Kustarev P., Bikovsky S., Pinkevich V.* Hardware violation monitor of transaction level real-time constraints for reliable systems on a chip // Proc. of 14th Intern. Multidisciplinary Scientific GeoConference (SGEM2014), Albena, Bulgaria, 17—26 June 2014. P. 201—208.

Сведения об авторе

Сергей Вячеславович Быковский — аспирант; Университет ИТМО; кафедра вычислительной техники; E-mail: bsv.serg@gmail.com

Рекомендована кафедрой вычислительной техники

Поступила в редакцию 22.12.14 г.

Ссылка для цитирования: *Быковский С. В.* Метод встроенной динамической актуализации функциональных моделей систем на кристалле // Изв. вузов. Приборостроение. 2015. Т. 58, № 3. С. 197—202.

METHOD OF EMBEDDED DYNAMIC ACTUALIZATION OF FUNCTIONAL MODEL OF SYSTEM ON A CHIP

S. V. Bikovsky

ITMO University, 197101, Saint Petersburg, Russia
E-mail: bsv.serg@gmail.com

A new method of formal model generation for system on chip (SoC) real-time functioning is proposed. The model is generated by embedded means of SoC during field tests or end-user operation.

The method can be used as a basis of the mechanism of SoC embedded runtime verification. In contrast to existing methods, requirements on instrumental memory size are decreased considerably.

Keywords: verification, functional model, fault, SoC, FSM, Kripke structure.

Data on author

Sergey V. Bikovsky — Post-Graduate Student; ITMO University; Department of Computer Science; E-mail: bsv.serg@gmail.com

Reference for citation: *Bikovsky S. V.* Method of embedded dynamic actualization of functional model of system on a chip // Izvestiya Vysshikh Uchebnykh Zavedeniy. Priborostroenie. 2015. Vol. 58, N 3. P. 197—202 (in Russian).

DOI: 10.17586/0021-3454-2015-58-3-197-202