НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ
сентябрь–октябрь 2018 Том 18 № 5 ISSN 2226-1494 http://ntv.ifmo.ru/
SCIENTIFIC AND TECHNICAL JOURNAL OF INFORMATION TECHNOLOGIES, MECHANICS AND OPTICS
September–October 2018 Vol. 18 No 5 ISSN 2226-1494 http://ntv.ifmo.ru/en

УДК 004.05+004.2

ТЕСТИРОВАНИЕ И ОТЛАДКА ВСТРАИВАЕМЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ НА ОСНОВЕ УРОВНЕВЫХ МОДЕЛЕЙ

В.Ю. Пинкевич[a], А.Е. Платунов[a]

[a] Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация
Адрес для переписки: vasiliy.pinkevich@yandex.ru

Информация о статье
Поступила в редакцию 31.05.18, принята к печати 26.06.18
doi: 10.17586/2226-1494-2018-18-5-801-808
Язык статьи – русский

Ссылка для цитирования: Пинкевич В.Ю., Платунов А.Е. Тестирование и отладка встраиваемых вычислительных систем на основе уровневых моделей // Научно-технический вестник информационных технологий, механики и оптики. 2018. Т. 18. № 5. С. 801–808. doi: 10.17586/2226-1494-2018-18-5-801-808

Аннотация

Предмет исследования. Рассмотрена проблема организации тестирования и отладки в комплексных (full stack) проектах встраиваемых вычислительных систем со сложной гетерогенной структурой. Работа направлена на создание унифицированной формализованной методики организации процессов тестирования и отладки, применимой для широкого класса встраиваемых систем инвариантно к способу их реализации. Сделан вывод о перспективности использования уровневых моделей представления встраиваемых систем в качестве способа унифицированного рассмотрения комплексных проектов встраиваемых систем. **Метод.** Для разработки методики использованы абстракции HLD-методологии проектирования встраиваемых систем, архитектурный стиль «модель-процесс-вычислитель» и аппарат теории множеств. **Основные результаты.** Разработана методика тестирования и отладки встраиваемых систем на основе уровневых моделей, в рамках которой предложено расширение архитектурного стиля «модель-процесс-вычислитель» для более точного описания отношений виртуализации. Разработан метод многоуровневого тестирования встраиваемых систем, который позволяет в едином стиле описывать тестовое окружение на разных фазах создания встраиваемых систем, обеспечивает формализацию понятий тестирования, верификации и валидации с позиций уровневого представления встраиваемых систем. Приведены примеры описания многоуровневых встраиваемых систем с помощью разработанной методики. **Практическая значимость.** Методика позволила предложить способы документирования тестовых окружений встраиваемых систем на различных фазах их создания, способы организации процессов тестирования, верификации, валидации и отладки в комплексных проектах встраиваемых систем. Предложенный подход повышает эффективность контроля требований к создаваемой встраиваемой системе за счет сквозного и прозрачного представления проекта в целом, формализации процедур тестирования и отладки.

Ключевые слова

встраиваемые системы, тестирование, верификация, валидация, отладка, высокоуровневое проектирование, виртуальная машина, вычислительная платформа

TESTING AND DEBUGGING OF EMBEDDED COMPUTING SYSTEMS BASED ON LEVEL MODELS

V.Yu. Pinkevich[a], A.E. Platunov[a]

[a] ITMO University, Saint Petersburg, 197101, Russian Federation
Corresponding author: vasiliy.pinkevich@yandex.ru

Article info
Received 31.05.18, accepted 26.06.18
doi: 10.17586/2226-1494-2018-18-5-801-808
Article in Russian

For citation: Pinkevich V.Yu., Platunov A.E. Testing and debugging of embedded computing systems based on level models. *Scientific and Technical Journal of Information Technologies, Mechanics and Optics*, 2018, vol. 18, no. 5, pp. 801–808 (in Russian). doi: 10.17586/2226-1494-2018-18-5-801-808

Abstract

Subject of Research. The paper deals with the problem of organization of testing and debugging in complex (full stack) projects of embedded computing systems with heterogeneous structure. The work is aimed at unified formal method

development to organize testing and debugging that is applicable to a wide range of embedded systems and invariant to their implementation. The conclusion is drawn about the prospects of using level models of embedded systems representation as a method of unified modeling of complex projects of embedded systems. **Method.** To develop the method, we used the abstractions of the HLD-methodology of embedded systems design, the "model-process-processor" architectural style and methods of set theory. **Main Results.** The embedded systems testing and debug method based on level models is developed. We enhanced the "model-process-processor" architectural style, as a part of the method, for a more accurate description of the virtualization relations. The method of embedded systems multi-level testing is developed, which allows describing the test environment at different phases of embedded systems creation in the same style, provides the formalization of the concepts of testing, verification and validation from the embedded systems level representation point of view. Examples of description of multi-level embedded systems using the developed method are given. **Practical Relevance.** The method provided documentation techniques of embedded systems test environments at different phases of their creation, ways to organize testing, verification, validation and debugging in complex projects of embedded systems. The proposed approach increases requirements control efficiency in the embedded systems projects owing to the end-to-end and transparent project representation as a whole and formalization of the testing and debugging procedures.

Keywords
embedded systems, testing, verification, validation, debug, high-level design, virtual machine, computing platform

Введение

В настоящее время встраиваемые вычислительные системы (встраиваемые системы, ВсС) лежат в основе автоматизации различных отраслей промышленности, транспорта, объектов инфраструктуры. Они являются важной составной частью киберфизических систем, Интернета вещей, других категорий информационно-коммуникационных систем, в том числе систем критической важности. Каждый проект ВсС уникален, так как решает конкретную прикладную задачу. Проектирование ВсС, даже при использовании готовых аппаратно-программных платформ и компонент, по-прежнему представляет собой трудоемкий и затратный процесс. Радикальное повышение эффективности создания новых ВсС остается актуальной задачей.

ВсС представляют собой комплексные проекты, при создании которых затрагивается значительное число уровней логической и физической организации системы – от разработки специализированных микросхем, системного и коммуникационного программного обеспечения до прикладного программирования. Для создания таких систем необходима вертикальная и горизонтальная интеграция решений из разных областей вычислительной техники, сегодня слабо совместимых между собой по применяемым методам и технологиям проектирования. Данная проблема стоит особенно остро, когда невозможно использовать готовую платформу с интегрированной средой разработки и требуется создать уникальное решение в рамках заказного проектирования [1–3]. Для этого должен быть выстроен индивидуальный маршрут проектирования ВсС, который обеспечивает интеграцию необходимых наборов проектных процессов из разных областей вычислительной техники, таких как прикладное и системное программирование, проектирование аппаратных функциональных блоков, коммуникационных протоколов. Слабая совместимость методов выполнения проектных процессов для разных областей вычислительной техники, вызванные этим барьеры в общении между разработчиками и сложность комплексного контроля качества от технического задания до итоговой реализации приводят к снижению эффективности проектирования в целом и высокой вероятности возникновения ошибок системного уровня, включая выбор базовых технологий [4–6].

Тестирование и отладка в широком смысле являются одними из важнейших проектных процессов, поскольку включают процессы верификации и валидации, а также исправление обнаруженных ошибок. При современном уровне сложности проектов ВсС выполнение тестирования в широком смысле для достижения требуемого уровня качества является одной из основных задач, требующей значительных затрат [1, 3].

Существующие работы по проблемам комплексного проектирования ВсС в основном направлены на способы описания целевой системы и на организацию процессов проектирования в целом. В них подчеркивается необходимость формализации и унификации проектных решений, отмечается слабая поддержка существующими методами проектирования сложных гетерогенных систем и работы в условиях неопределенности требований, указывается на недостаточную эффективность существующих методов исследования пространства проектных решений [4, 7, 8]. При этом работы, посвященные проблемам формализации тестирования и отладки ВсС в широком смысле, либо относятся к конкретным технологиям [9, 10], что не позволяет использовать их в комплексном проектировании ВсС, либо носят описательный характер [5, 11], что затрудняет их применение непосредственно в разработке маршрутов проектирования и тестирования.

Необходима методика организации проектных процессов тестирования и отладки ВсС в широком смысле, которая позволит обеспечить их формализацию и унификацию для комплексных проектов ВсС в течение всех фаз их создания. В работе предлагается оригинальная методика, отвечающая перечисленным требованиям.

Использование уровневых моделей в проектировании ВсС

Уровневое представление вычислительных систем является широко используемым и эффективным способом описания их вертикальной организации, отражающим набор используемых уровней абстракции и соответствующих технологий разработки, в том числе из разных областей вычислительной техники. Значительными выразительными возможностями для моделирования уровневой организации вычислительных систем обладает архитектурный стиль «модель-процесс-вычислитель» (МПВ) [12], что определило его использование в предлагаемой методике тестирования и отладки в качестве способа описания комплексных проектов ВсС.

Стиль МПВ опирается на представление о процессе создания ВсС как об организации вычислительного процесса, включающего различные фазы выполнения вычислений: на этапах создания системы (design time), ее конфигурирования (config time), работы (run time) [7, 8, 13]. Стиль предлагает способ описания организации комплексных гетерогенных ВсС как взаимосвязанного набора «уровней», каждый из которых представляет собой тройку «модель-процесс-вычислитель» (рис. 1). Уровни охватывают значимые в рамках проекта вычислительные платформы и разрабатываемые в рамках данных платформ элементы целевой системы. Выделяемые платформы обычно относятся к конкретным областям вычислительной техники. Под «процессом» понимается вычислительный процесс, протекающий во время работы системы. «Модель» – это спецификация вычислительного процесса на некотором уровне абстракции, которая настраивает вычислитель для выполнения конкретного вычислительного процесса (примеры: код программы, проект вычислительного блока для программируемых логических интегральных схем (ПЛИС) и т.п.). Такая зависимость между моделью и процессом называется отношением соответствия. «Вычислитель» – это платформа, на которой реализуется вычислительный процесс, задаваемый моделью. Вычислитель может рассматриваться как физическая или виртуальная вычислительная машина в широком смысле [12, 14].

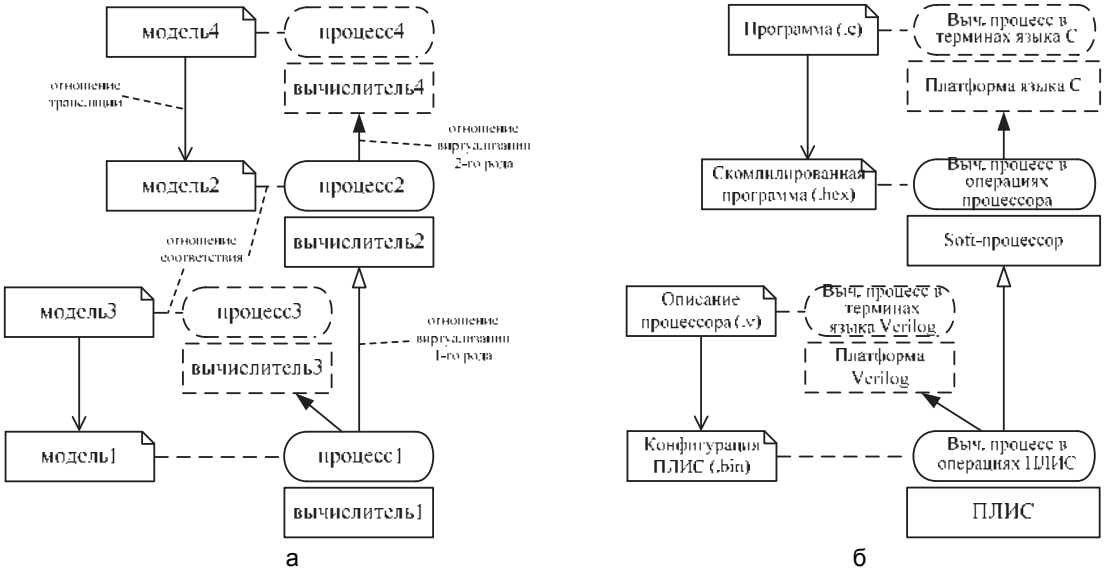

Рис. 1. Элементы уровневой организации встраиваемых систем в рамках расширенного архитектурного стиля «модель-процесс-вычислитель» (а). Пример описания системы на программируемой логической интегральной схеме с soft-процессором (б)

Для того чтобы отобразить зависимости между уровнями, вводятся специальные отношения между элементами троек разных уровней (рис. 1, а):

1. отношение трансляции между моделями показывает, что одна модель преобразуется (компилируется, синтезируется, переписывается вручную) в другую;
2. отношение виртуализации между уровнями показывает, что вычислительный процесс «порождает» вычислитель более высокого уровня.

Для более точного описания природы виртуализации архитектурный стиль МПВ расширяется авторами путем деления отношения виртуализации на два рода.

1. Отношение виртуализации 1-го рода имеет место, когда модель верхнего уровня не связана с моделью нижнего уровня отношением трансляции, а вычислительный процесс нижнего уровня порождает вычислитель или модель более высокого уровня.
2. Отношение виртуализации 2-го рода имеет место, когда модели связаны отношением трансляции, а вычислительный процесс нижнего уровня порождает и вычислитель, и вычислительный процесс

верхнего уровня, так как в результате выполненной трансляции модель нижнего уровня содержит в себе описание обоих элементов.

Вычислители, связанные с более низкими уровнями отношениями виртуализации 1-го рода или не имеющие нижележащих уровней, можно рассматривать как «реальные» (существующие во время работы целевой системы; показаны на рис. 1 сплошной линией), а остальные – как виртуальные (показаны на рис. 1 пунктиром).

Предлагаемая методика тестирования и отладки ВсС базируется на расширенном архитектурном стиле МПВ и включает следующие части:

1. метод организации процесса создания ВсС от исходных спецификаций к итоговой реализации в виде набора последовательно расширяемых уровневых моделей;
2. метод многоуровневого тестирования ВсС и введенные на его основе формальные определения тестирования (в узком смысле), верификации и валидации в контексте уровневой модели представления ВсС (далее эти понятия используются в таком контексте, если не указано другое);
3. методы организации процессов верификации, валидации и отладки на основе уровневой модели ВсС.

В целом методика направлена на создание и развитие инструментов и технологий «прозрачного» и сквозного представления проекта ВсС разработчиком на всех этапах.

Организация процесса создания ВсС на основе уровневого представления

Процесс создания ВсС предлагается разбивать на фазы, представленные в [7, 8]. Каждая фаза характеризуется определенным набором технологий и средств разработки. Основные фазы, встречающиеся в большинстве проектов – это высокоуровневое проектирование, рабочее проектирование (разработка), изготовление, испытания. Процесс создания целевой системы при переходах между фазами удобно представлять с помощью уровневых моделей в стиле МПВ. Проектирование системы начинается на «высоком» уровне абстракции, затем ее представление постепенно детализируется, при этом выбираются базовые технологии разработки и вводятся для рассмотрения новые уровни. Таким образом, смена уровневых моделей характеризует повышение детализации проекта целевой системы при движении от исходных спецификаций к итоговой реализации.

На каждой фазе разрабатываются представления целевой системы в рамках уровневых моделей (возможно более одного представления на каждой фазе). После анализа полученных результатов определяется уровневая модель для следующей фазы, обычно путем расширения «вниз». При этом уровни, на которых велось проектирование на предыдущей фазе, связываются с новыми уровнями отношениями виртуализации 1-го рода (если модель транслируется в другое представление) или 2-го рода (если модель переносится на другой вычислитель). Пример перехода к новой уровневой модели показан на рис. 2. К исходному уровню один раз применена виртуализация 1-го рода и два раза – 2-го рода.

За счет явного введения связей между уровнями и преемственности уровневых моделей проектная информация с предыдущих фаз не теряется и может быть отслежена на более низких уровнях реализации. Необходимо отметить, что существующие системы автоматизированного проектирования обычно покрывают максимум одно–два отношения виртуализации между уровнями целевой системы в середине всего стека уровней, поэтому многие операции трансляции, особенно на высоких уровнях, приходится выполнять вручную.

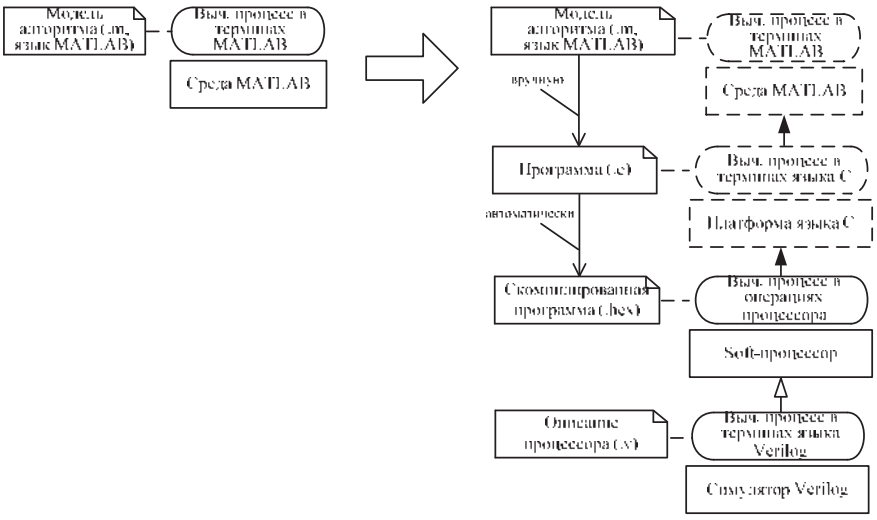

Рис. 2. Пример смены уровневой модели при переходе с фазы высокоуровневого проектирования на фазу разработки

Многоуровневое тестирование ВсС на основе исходных требований

Традиционно тестирование (в узком смысле) выполняется только на последней фазе создания вычислительной системы, когда имеется ее итоговая реализация. Однако современные средства моделирования и симуляции разных уровней позволяют включать в маршрут проектирования операции тестирования системы в целом и ее отдельных компонентов на абстрактных уровнях представления, причем оцениваются и функциональные, и нефункциональные характеристики (такие как энергопотребление, габаритные размеры, стоимость и т.п.).

По сути, наборы тестов на разных уровнях абстракции отражают требования к целевой системе, которые последовательно детализируются вместе с ней от технического задания до итоговой реализации. Считаем целесообразным рассматривать тестовые наборы разных фаз в качестве особой формы представления требований к создаваемой ВсС.

В данной работе определим тест как набор требований к системе (функциональных и (или) нефункциональных) и способ определения того, выполняются ли эти требования. Для каждого уровня в рамках уровневой модели ВсС может быть определено множество тестов, которое обеспечивает проверку всех необходимых требований на этом уровне. Максимальная доля тестов должна выполняться автоматизированными способами. При условии успешного выполнения всего множества тестов можно считать, что данная реализация целевой системы корректна.

Понятие тестирования в узком смысле определим как разработку множества (множеств) тестов для системы в контексте уровневой модели и их выполнение. Для полного тестирования ВсС необходимо разработать совокупность множеств тестов для всех «реальных» уровней в ее уровневой модели.

Отметим, что при изменении уровневой модели целевой системы (например, см. рис. 2) необходимо проводить верификацию для проверки корректности новой реализации.

На этапе составления технического задания исходные требования к ВсС часто невозможно сформулировать с достаточной точностью. Составленные по ним множества тестов не будут отражать реальные требования к системе. В связи с этим необходимо выполнять валидацию, т.е. тестировать целевую систему не только с помощью искусственно разработанных множеств тестов, но и в реальном целевом окружении с целью уточнения требований.

Организация процессов верификации, валидации и отладки на основе уровневой модели ВсС

Верификация. В рамках введенных понятий верификацией будем называть тестирование (в узком смысле) некоторой реализации целевой системы, которая является развитием исходной реализации, с другой уровневой моделью, с целью проверки соответствия новой системы исходной системе.

Обозначим исходную и новую системы как A и B соответственно, а множества тестов для них как T_A и T_B. Для этих множеств должно быть справедливо всюду определенное сюръективное соответствие φ, т.е. $T_A \xrightarrow{\varphi} T_B$. Это соответствие означает, что если выполняется тест $t_A \in T_A$, то при условии отсутствия ошибок реализации целевой системы будет выполняться и $t_B \in T_B$, такой что $(t_A, t_B) \in \varphi$. Тогда, если все тесты из T_B прошли успешно, верификация считается успешно выполненной.

Рассмотрим особенности выполнения верификации, связанные с изменением уровневой модели на примере одного уровня и соответствующего множества тестов в разных случаях.

1. Если к уровню была применена виртуализация 1-го рода, то это означает, что в рассмотрение вводится новый вычислитель, спецификация которого не связана со спецификацией исходного уровня отношением трансляции. Следовательно, набор тестов, необходимый для проверки данного нового вычислителя, не может быть получен из набора тестов исходного уровня и должен быть разработан независимо. В таком случае верификация новой реализации выполняется на том же уровне, что и исходной, и дополнительно выполняется тестирование нового вычислителя.
2. Если к уровню была применена виртуализация 2-го рода (т.е. модель была транслирована в более низкоуровневую), то верификация новой реализации выполняется также на более низком уровне.
3. Если для уровня был только заменен вычислитель (без виртуализации), то верификация выполняется аналогично 1-му случаю.

Следует отметить, что при переходе к новой уровневой модели виртуализация может быть применена сразу несколько раз подряд (рис. 2).

Валидация. Тестирование может проводиться как с помощью искусственно разработанных тестов и тестового окружения, так и в целевом окружении с реальными условиями эксплуатации. Обычно, если имеется такая возможность, применяются оба способа. Тестирование в целевом окружении является обязательным для проверки соответствия полученной целевой системы своему назначению. При проектировании стремятся к тому, чтобы тесты, которые будут проведены в целевом окружении, были предусмотрены в качестве требований и в искусственном тестовом окружении. Однако это не всегда возможно как из-за неопределенности требований на ранних стадиях проектирования, так и из-за технологических или иных ограничений. В связи с этим часть тестов, которые будут проведены в реальных условиях, могут не входить во множество искусственных тестов. Также из-за различных ограничений может отсутствовать

возможность провести тестирование одной и той же реализации системы и в искусственном, и в реальном тестовом окружении. Искусственное тестовое окружение зачастую присутствует только у моделей и прототипов системы, а целевое окружение – только у итоговой реализации.

Создание целевой системы проходит ряд итераций, в течение которых создаются промежуточные реализации (модели и прототипы) для проведения экспериментов, направленных на уточнение реальных целевых условий эксплуатации системы и прочих требований.

Обозначим множество тестов, которые проводятся для реализации системы с уровневой моделью A на уровне i в искусственном окружении, как T_{Ai}, а множество тестов, которые проводятся для той же реализации на том же уровне в целевом окружении, как T'_{Ai}. Тогда валидацией будем называть совокупность процессов тестирования, в результате которых устанавливается соответствие между множествами тестов T_{Ai} и T'_{Ai}. В общем случае валидация производится при участии двух реализаций системы – модельной A, для которой тесты проводятся в искусственном окружении на уровне i, и целевой B, для которой тесты проводятся в целевом окружении на уровне j. В реализации B данные уровни должны быть связаны между собой отношением (или цепочкой отношений) виртуализации 2-го рода. Два типа множеств тестов обеих реализаций должны быть связаны отношениями φ и ψ, а также обратными отношениями для установления связи между тестами разных уровней: $T_{Ai} \xrightarrow{\varphi} T_{Bj}$, $T'_{Ai} \xrightarrow{\psi} T'_{Bj}$ и $T'_{Bj} \xrightarrow{\psi^{-1}} T'_{Ai}$.

При выполнении валидации возможны следующие случаи:

1. $T'_{Ai} = T_{Ai}$, значит, все требования к системе были выполнены, и избыточных требований заложено не было – валидация прошла успешно;
2. $T'_{Ai} \setminus T_{Ai} \neq \varnothing$, значит, часть требований не была предусмотрена при проектировании целевой системы – валидация выявила необходимость доработки;
3. $T_{Ai} \setminus T'_{Ai} \neq \varnothing$, значит, исходные требования к системе оказались избыточны.

Выявленные невостребованные свойства системы требуют анализа, поскольку могут представлять собой недокументированные возможности компонентов сторонних разработчиков, представляющие угрозу информационной безопасности, или избыточную функциональность, поддержка которой увеличивает расход ресурсов и снижает общую надежность.

Отладка. Если при выполнении тестирования, верификации или валидации возникают ошибки, то необходимо иметь возможность определить, при запуске какого именно теста возникла ошибка (это может быть неочевидно, если проводится тестирование в реальном окружении) и что является ее причиной. Если тестовая инфраструктура и соответствия между множествами тестов корректны, то для любого теста t справедливо $t \in T'$ и (или) $t \in T$. Тогда возможны следующие случаи и соответствующие им способы отладки (рассматривается на примере валидации; для остальных случаев аналогично).

1. $t \in T_{Bj}$, значит, данный тест предусмотрен в искусственном тестовом окружении, и ошибка содержится в модели или вычислителе уровня j реализации B. Выяснить источник ошибки возможно путем отладки вычислителя уровня j и трансляторов (методик трансляции) более высоких уровней до уровня i включительно. Для этого возможна замена вычислителя и трансляторов на аналоги.
2. $t \notin T_{Bj}$, но $t \in T'_{Bj}$, т.е. данный тест не был предусмотрен в искусственном тестовом окружении; это означает, что обнаружена неточность в требованиях к системе на данном уровне и всех вышележащих до уровня i: следует вернуться на уровень i, скорректировать целевую реализацию и множество тестов T_{Ai}, после чего провести тестирование, верификацию и валидацию заново.

Модификация стандартного маршрута проектирования ВсС

Для применения разработанной методики тестирования и отладки в стандартный маршрут проектирования ВсС включаются следующие шаги:

1. на этапе проектирования при выборе технологий реализации:
   - явно определяется набор и структура уровневых моделей, которые будут создаваться в процессе проектирования целевой системы;
   - разрабатываются множества тестов и определяются операции соответствия между ними, которые будут показывать, каким образом выполняется контроль требований при переходах между реализациями системы с разными уровневыми моделями;
2. на всех фазах создания ВсС выполняется тестирование, верификация и валидация системы с применением разработанных множеств тестов и операций соответствия в рамках предложенных методов.

Заключение

Применение предлагаемого уровневого подхода к организации процессов тестирования и отладки встраиваемых систем позволяет:

- интегрировать технологии тестирования, верификации и валидации различных платформ в рамках единого комплексного проекта;

– с новых позиций в унифицированном стиле рассматривать вновь разрабатываемые и готовые сторонние компоненты создаваемой вычислительной системы как равноправные объекты тестирования, верификации, валидации и отладки;
– контролировать эволюцию проекта встраиваемой системы в ходе проектирования, выявляя и фиксируя вносимые разработчиками значимые технические проектные решения, которые требуют введения новых и пересмотра применяемых уровней представления встраиваемой системы и соответствующих методов тестирования и отладки.

Предлагаемый подход был апробирован при разработке метода и инструментальных средств модельно-ориентированного тестирования и отладки встраиваемых систем [15, 16]. Данный метод, наряду с самостоятельным применением, может встраиваться в общий маршрут проектирования, позволяя решить задачи выявления и воспроизведения нетривиальных ситуаций с некорректным поведением создаваемой встраиваемой системы. Это позволяет эффективно диагностировать ошибки проектирования, опираясь на уровневое представление системы.

Методика тестирования и отладки встраиваемых систем на основе уровневых моделей внедрена в ряд проектов в области инфраструктурной автоматики (распределенные системы класса Интернета вещей) и в области специализированных гетерогенных вычислительных платформ для телекоммуникационных систем и аналитических приборов. Усилия авторов направлены на реализацию инструментальных средств, максимально автоматизирующих генерацию уровневых моделей и цепочек тестовых множеств для эффективного применения представленной методики массовым разработчиком встраиваемых систем.

Литература

1. Marwedel P. Embedded System Design: Embedded Systems, Foundations of Cyber-Physical Systems. 2nd ed. Springer, 2011. 400 p. doi: 10.1007/978-94-007-0257-8
2. Lee E.A., Seshia S.A. Introduction to Embedded Systems: A Cyber-Physical Systems Approach. 2nd ed. MIT Press, 2017.
3. Карпов Ю.Г. Model Checking. Верификация параллельных и распределенных программных систем. СПб.: БХВ-Петербург, 2010. 560 с.
4. Sangiovanni-Vincentelli A., Damm W., Passerone R. Taming Dr. Frankenstein: contract-based design for cyber-physical systems // European Journal of Control. 2012. V. 18. N 3. P. 217–238. doi: 10.3166/EJC.18.217-238
5. Lettnin D., Winterholer M. Embedded Software Verification and Debugging. Springer, 2017. 208 p. doi: 10.1007/978-1-4614-2266-2
6. Непейвода Н.Н., Скопин И.Н. Основания программирования. Москва-Ижевск: Институт компьютерных исследований, 2003. 913 с.
7. Platunov A., Kluchev A., Penskoi A. Expanding design space for complex embedded systems with HLD-methodology // Proc. 6th Int. Congress on Ultra Modern Telecommunications and Control Systems and Workshops. 2014. P. 157–164. doi: 10.1109/icumt.2014.7002096
8. Платунов А.Е. Реконфигурируемые встраиваемые системы и системы на кристалле // Известия вузов. Приборостроение. 2014. Т. 57. № 4. С. 49–52.
9. Adir A., Copty S., Landa S., Nahir A., Shurek G., Ziv A., Meissner C., Schumann J. A unified methodology for pre-silicon verification and post-silicon validation // Proc. Design, Automation & Test in Europe. Grenoble, France, 2011. doi: 10.1109/DATE.2011.5763252
10. Wagner I., Bertacco V. Reversi: post-silicon validation system for modern microprocessors // Proc. IEEE Int. Conf. on Computer Design. Lake Tahoe, USA, 2008. doi: 10.1109/ICCD.2008.4751878
11. Broekman B., Notenboom E. Testing Embedded Software. Addison-Wesley, 2003. 368 p.
12. Пенской А.В. Архитектурное документирование встроенных систем с многоуровневой конфигурацией // Известия вузов. Приборостроение. 2015. Т. 58. № 7. С. 527–532.
13. Ключев А.О., Кустарев П.В., Палташев Т.Т., Платунов А.Е. Применение HLD-методологии для проектирования реконфигурируемых встраиваемых систем // Научно-технический вестник информационных технологий, механики и оптики. 2014. № 4(92). С. 74–82.
14. Carloni L., De Bernardinis F., Pinello C., Sangiovanni-Vincentelli A.L., Sgroi M. Platform-based design for embedded

References

1. Marwedel P. *Embedded System Design: Embedded Systems, Foundations of Cyber-Physical Systems*. 2nd ed. Springer, 2011, 400 p. doi: 10.1007/978-94-007-0257-8
2. Lee E.A., Seshia S.A. *Introduction to Embedded Systems: A Cyber-Physical Systems Approach*. 2nd ed. MIT Press, 2017.
3. Karpov Yu.G. *Model Checking. Verification of Parallel and Distributed Software Systems*. St. Petersburg, BKhV-Peterburg Publ., 2010, 560 p. (in Russian)
4. Sangiovanni-Vincentelli A., Damm W., Passerone R. Taming Dr. Frankenstein: contract-based design for cyber-physical systems. *European Journal of Control*, 2012, vol. 18, no. 3, pp. 217–238. doi: 10.3166/EJC.18.217-238
5. Lettnin D., Winterholer M. *Embedded Software Verification and Debugging*. Springer, 2017, 208 p. doi: 10.1007/978-1-4614-2266-2
6. Nepeivoda N.N., Skopin I.N. *Reasons for Programming*. Moscow-Izhevsk, Institute of Computer Research Publ., 2003, 913 p. (in Russian)
7. Platunov A., Kluchev A., Penskoi A. Expanding design space for complex embedded systems with HLD-methodology. *Proc. 6th Int. Congress on Ultra Modern Telecommunications and Control Systems and Workshops*, 2014, pp. 157–164. doi: 10.1109/icumt.2014.7002096
8. Platunov A.E. Reconfigurable embedded systems and system-on-chip. *Journal of Instrument Engineering*, 2014, vol. 57, no. 4, pp. 49–52. (in Russian).
9. Adir A., Copty S., Landa S., Nahir A., Shurek G., Ziv A., Meissner C., Schumann J. A unified methodology for pre-silicon verification and post-silicon validation. *Proc. Design, Automation & Test in Europe*. Grenoble, France, 2011. doi: 10.1109/DATE.2011.5763252
10. Wagner I., Bertacco V. Reversi: post-silicon validation system for modern microprocessors. *Proc. IEEE Int. Conf. on Computer Design*. Lake Tahoe, USA, 2008. doi: 10.1109/ICCD.2008.4751878
11. Broekman B., Notenboom E. *Testing Embedded Software*. Addison-Wesley, 2003, 368 p.
12. Penskoi A.V. Architectural specification of embedded systems with multi-level configuration. *Journal of Instrument Engineering*, 2015, vol. 58, no. 7, pp. 527–532 (in Russian).
13. Klyuchev A.O., Kustarev P.V., Paltashev T.T., Platunov A.E. HLD-methodology application for reconfigurable embedded systems design. *Scientific and Technical Journal of Information Technologies, Mechanics and Optics*, 2014, no. 4, pp. 74–82. (in Russian)
14. Carloni L., De Bernardinis F., Pinello C., Sangiovanni-Vincentelli A.L., Sgroi M. Platform-based design for embedded systems. In *The Embedded Systems Handbook*. Ed.

systems / In: The Embedded Systems Handbook. Ed. R. Zurawski. Boca Raton: CRC Press, 2005. 1112 p.
15. Pinkevich V., Yanalov R., Platunov A. Hardware computational units design with combined debug capabilities // Proc. 17th Int. Multidisciplinary Scientific GeoConference, SGEM. 2017. V. 17. N 21. P. 77–84. doi: 10.5593/sgem2017/21/s07.011
16. Пинкевич В.Ю. Подход к разработке систем потоковой обработки данных на ПЛИС с возможностью комбинированной отладки // Известия вузов. Приборостроение. 2017. Т. 60. № 10. С. 967–972. doi: 10.17586/0021-3454- 2017-60-10-967-972

Авторы

Пинкевич Василий Юрьевич – аспирант, Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация, Scopus ID: 56951052900, ORCID ID: 0000-0002-8635-5026, vasiliy.pinkevich@yandex.ru

Платунов Алексей Евгеньевич – доктор технических наук, профессор, профессор, Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация, Scopus ID: 35318291200, ORCID ID: 0000-0003-3003-3949, aeplatunov@gmail.com

R. Zurawski. Boca Raton, CRC Press, 2005, 1112 p.
15. Pinkevich V., Yanalov R., Platunov A. Hardware computational units design with combined debug capabilities. *Proc. 17th Int. Multidisciplinary Scientific GeoConference, SGEM*, 2017, vol. 17, no. 21, pp. 77–84. doi: 10.5593/sgem2017/21/s07.011
16. Pinkevich V.Yu. An approach to design of FPGA-based systems for stream data processing with capability of combined debugging. *Journal of Instrument Engineering*, 2017, vol. 60, no. 10, pp. 967–972. (in Russian)

Authors

Vasiliy Yu. Pinkevich – postgraduate, ITMO University, Saint Petersburg, 197101, Russian Federation, Scopus ID: 56951052900, ORCID ID: 0000-0002-8635-5026, vasiliy.pinkevich@yandex.ru

Alexey E. Platunov – D.Sc., Full Professor, ITMO University, Saint Petersburg, 197101, Russian Federation, Scopus ID: 35318291200, ORCID ID: 0000-0003-3003-3949, aeplatunov@gmail.com