МОДЕЛИРОВАНИЕ
MODELING

УДК 004.3.02 DOI: 10.17587/it.26.259-266

А. Д. Иванников, д-р техн. наук, гл. науч. сотр., e-mail: ADI@ippm.ru,
Институт проблем проектирования в микроэлектронике РАН,

Сравнительный анализ методов отладки цифровых систем на этапе проектирования

Рассматриваются методы отладки, т. е. выявления и исправления ошибок в проектах цифровых систем, заданных схемой технических средств и текстом программного или микропрограммного обеспечения. Описывается отладка на изготовленном макете технических средств и на программной модели цифровой системы, а также на сочетании макета отдельных блоков и программной модели остальной части системы. Анализируются методы организации отладочных режимов для всех рассматриваемых методов. Указывается, что для цифровых систем на кристалле наиболее эффективным методом является отладка проектов на программной модели.

Ключевые слова: *проектирование цифровых систем, отладка проектов, организация отладочных режимов*

Введение

При проектировании цифровых систем, результатом которого является схема соединения блоков и элементов цифровых систем, а также текст программного или микропрограммного обеспечения, важным этапом разработки является отладка, т. е. выявление и исправление ошибок проектирования [1—4]. При постоянно возрастающей сложности современных цифровых систем этап отладки схемы технических средств, программного обеспечения, а также их совместной работы может занимать существенную часть всего этапа проектирования [1, 3, 5].

1. Составляющие процесса отладки

Для проведения отладки проектов цифровых систем необходимы следующие составляющие.

1. Объект отладки, позволяющий по заданному входному воздействию определить реакцию на выходе. Объектом отладки может являться как макет или прототип цифровой системы, так и ее программная модель.

2. Конечный набор конечных по времени входных воздействий (назовем их отладочными тестами), которые подаются на объект отладки во время отладочных экспериментов. Набор отладочных тестов должен обладать свойством полноты, т. е. из правильности реакции объекта отладки на все тесты набора должна следовать правильность его работы при любых допустимых входных воздействиях.

3. Критерий определения правильности выходной реакции объекта отладки на поданное входное воздействие.

4. Возможность локализации ошибок в цифровой системе при неправильной реакции объекта отладки на заданное входное воздействие.

5. Средства для внесения исправлений в объект отладки и повторения отладочного эксперимента.

Проведем анализ возможных методов отладки цифровых систем на этапе проектирования, рассмотрев поочередно составляющие процесса отладки.

2. Объект отладки и методы отладки цифровых систем на этапе проектирования

Отладка цифровой системы может быть проведена как на изготовленном макете цифровой системы, так и на некоторой модели последней [6—9]. В соответствии с различным представлением разработанной цифровой системы можно выделить четыре метода отладки, возможных на этапе проектирования, а именно:

- метод полного макетирования;

- метод частичного макетирования при использовании типового ядра технических средств;
- метод моделирования на ЭВМ;
- метод моделирования на ЭВМ с использованием реальных блоков технических средств.

При использовании *метода полного макетирования* начало отладки технических средств и программно-микропрограммного обеспечения и их совместной работы возможно только после изготовления макета, что требует материальных затрат и занимает при современной структуре разрабатывающих организаций достаточно длительные сроки.

В процессе анализа работы цифровой системы и локализации ошибок проектирования принципиальной схемы и программно-микропрограммного обеспечения необходимо следить за изменениями содержимого регистров системы и ячеек памяти, а также за логическими сигналами на шинах и линиях технических средств, иметь возможность останавливать и возобновлять выполнение программы или микропрограммы, задавать и модифицировать содержимое внутренних регистров системы и ячеек запоминающих устройств (ЗУ). Если слежение за изменениями логических сигналов на внешних выводах системы в макете может быть осуществлено с помощью логических анализаторов [3, 6, 10], то контроль за изменениями содержимого внутренних регистров и ячеек ЗУ, не имеющих внешних выводов, останов и повторный запуск программы или микропрограммы, задание и изменение содержимого внутренних регистров и ячеек ЗУ, т. е. отладочные режимы, невозможны, если не предусмотрены специальные средства внутри макета. Эти операции могут быть осуществлены только при наличии в макете дополнительных аппаратных затрат, специального программного обеспечения и подключения интерактивных устройств ввода-вывода, например, системы JTAG [7, 11], которая часто предусматривается в разрабатываемой цифровой системе. В ряде случаев в разрабатываемых цифровых системах предусматриваются располагающиеся на кристалле встроенные логические анализаторы, подключаемые к определенным логическим узлам с помощью управляющих логических сигналов [3]. Встраиваемые в цифровую систему дополнительные средства во многом облегчают процесс отладки, но не в полной мере. Кроме того, разработка дополнительных аппаратно-программных средств для эффективной организации отладочных режимов на макете является нетривиальной.

Отладка цифровой системы на макете технических средств при конкретных значениях динамических параметров блоков не гарантирует работоспособность технических средств системы при любых допустимых по ТУ значениях динамических параметров блоков.

Отладка макета совместно с управляемым объектом в реальном времени часто затруднена в связи с невозможностью создания реальных, например, аварийных ситуаций на объекте. Такая отладка может быть осуществлена путем совместного моделирования на ЭВМ цифровой системы и поведения управляемого объекта в одинаковом масштабе времени.

Метод частичного макетирования предполагает разделение технических средств разрабатываемой системы на типовое ядро (центральную часть), одинаковое для данного класса цифровых систем, и часть, например, схемы сопряжения с конкретным объектом управления, присутствующую только в разрабатываемой системе. Типовое ядро, включающее аппаратно-программные средства организации отладочных режимов, может быть изготовлено и отлажено один раз, после чего в сочетании с макетом нетиповой части используется при отладке различных цифровых систем определенного класса.

Так, во многих простейших цифровых системах управления (контроллерах) основная по объему центральная часть (процессорный блок, ЗУ, сопряжение с типовыми внешними устройствами, типовые каналы ввода-вывода) является типовой, в связи с чем метод частичного макетирования для таких систем применим в большей степени, чем для цифровых систем других классов [8, 12—14].

В случае микропрограммируемых систем, для которых характерно большое разнообразие структур и архитектур, в типовое ядро включаются блок микропрограммного управления и ЗУ микропрограмм большой разрядности, а блок обработки данных вместе со схемами сопряжения с обслуживаемым объектом макетируется в соответствии со схемой технических средств разрабатываемой системы. При этом макетируемая часть системы существенно больше, чем в случае систем на основе вычислителей с фиксированной системой команд.

Ввиду фиксированной организации блока микропрограммного управления типового ядра метод частичного макетирования не позволяет получить объекты отладки для всего разнообразия микропрограммируемых цифровых систем.

В случае многопроцессорных цифровых систем метод частичного макетирования может быть также использован. В этом случае макетируемая часть представляет собой нестандартные схемы сопряжения с обслуживаемым объектом и, возможно, обрабатывающие блоки, аппаратно реализующие определенные операции, доля которых в общем объеме технических средств, как правило, не велика. При нетиповой организации многопроцессорной системы применение метода частичного макетирования затруднено в связи с невозможностью выделения существенного по объему типового ядра технических средств.

Таким образом, характеризуя метод частичного макетирования в целом, можно отметить следующее:

1) метод требует меньших затрат на изготовление макета, чем метод полного макетирования;

2) метод может быть эффективно использован при проектировании систем, имеющих существенное по объему типовое ядро технических средств. Применение метода при нетиповой структуре и архитектуре разрабатываемой цифровой системы невозможно;

3) для метода частичного макетирования характерен ряд недостатков, свойственных методу макетирования вообще, а именно:

- трудоемкость внесения изменений в принципиальную схему в процессе отладки;
- невозможность гарантировать работоспособность технических средств при любом допустимом разбросе задержек блоков;
- трудность отладки макета системы совместно с обслуживаемым объектом в реальном времени в связи с невозможностью искусственного создания ряда ситуаций на объекте.

Метод моделирования на ЭВМ предполагает использование в качестве объекта отладки программной модели проектируемой системы [1, 9, 15—18]. Этот метод является универсальным в том смысле, что модель может быть получена для цифровой системы любой структуры и архитектуры, в том числе для нетиповых микропрограммируемых и мультипроцессорных систем.

Проверка работоспособности технических средств цифровой системы для всех допустимых по ТУ значениях динамических параметров блоков одновременно возможна только методом моделирования на ЭВМ. Этот метод принципиально позволяет проводить отладку путем совместного моделирования цифровой системы и обслуживаемого объекта во всех возможных режимах работы последнего [13]. Использование метода моделирования сокращает сроки и стоимость процесса отладки в связи с отсутствием необходимости изготовления макета до окончания отладки проекта.

Недостатком метода моделирования является меньшее относительно макета быстродействие программной модели отлаживаемой системы. Так, при моделировании на уровне системы команд на эмуляцию одной команды процессорного элемента цифровой системы приходится от нескольких десятков до нескольких сотен команд инструментальной ЭВМ. При функционально-логическом моделировании системы на уровне блоков затраты машинного ресурса, как правило, еще больше.

Тем ни менее, использование машинных моделей цифровых систем в качестве объекта отладки весьма эффективно вследствие своей универсальности относительно типа отлаживаемых систем, отсутствия необходимости изготовления макета, возможности параллельного анализа нескольких технических решений.

Применение *метода моделирования на ЭВМ с использованием реальных блоков* обусловлено следующим. Как указывалось выше, недостатком использования машинных моделей цифровых систем является большое время моделирования. При моделировании на функционально-логическом уровне обычно каждому блоку соответствует своя программная модель, а модель технических средств в целом генерируется путем комбинации моделей блоков в соответствии с заданной схемой соединения блоков. Модели цифровых блоков сверхвысокой степени интеграции сложны в разработке, занимают большой объем памяти и требуют больших затрат машинного времени при моделировании. В связи с этим известно использование автоматизированных рабочих мест для проектирования, в которых для отладки используется функционально-логическое моделирование на уровне блоков, а центральное вычислительное ядро подключается к системе физически. При этом, однако, возникают две проблемы: отсутствие в процессе отладки доступа к внутренним регистрам физически подключаемого блока и существенно разные скорости работы физического блока и программной модели.

Таким образом, метод моделирования на ЭВМ с использованием реальных блоков позволяет уменьшить затраты времени на моделирование. Тем не менее, этот метод не гарантирует работоспособность отлаживаемой системы при любых допустимых значениях динамических

параметров аппаратно подключаемых блоков. Кроме того, поскольку скорость работы объекта отладки в этом случае определяется моделирующей частью и существенно меньше скорости работы реальной системы, трудно гарантировать работоспособность системы при реальном режиме работы.

Метод моделирования с использованием реальных блоков следует рассматривать как расширение метода моделирования.

3. Локализация ошибок в проектах цифровых систем

Единственным универсальным методом локализации ошибок как в схеме технических средств, так и в программно-микропрограммном обеспечении на этапе отладки проекта является анализ разработчиком внутренних переменных: содержимого ячеек памяти и регистров, а также логических сигналов на шинах и линиях цифровой системы в процессе ее работы. Избирательный и детальный анализ указанных внутренних переменных в сочетании со знанием требуемого функционирования разрабатываемой системы позволяет разработчику предположить и проверить наличие конкретных ошибок в проекте.

Таким образом, возможность удобного и детального анализа изменения внутренних переменных цифровой системы является основным средством локализации ошибок проектирования.

При использовании *метода полного макетирования* слежение за логическими сигналами на шинах системы и выводах блоков осуществляется с помощью логических анализаторов различных типов, в том числе встроенных в проектируемую систему. Эти приборы позволяют проверить логические диаграммы в узлах принципиальной схемы при возникновении заданных условий.

Слежение за изменением содержимого внутренних регистров блоков и ячеек ЗУ, а также организация отладочных режимов программно-микропрограммного обеспечения при методе полного макетирования осуществляется с помощью загружаемых в ОЗУ системы отладочных мониторов. Однако без дополнительных аппаратных средств организации отладочных режимов выполнение программного обеспечения осуществляется медленнее, чем в реальной аппаратуре. Отладочные мониторы, ориентированные на слежение за содержимым ячеек ЗУ и регистров типовой части, не позволяют анализировать состояние регистров нетиповых схем сопряжения с внешними устройствами.

В случае нетиповых микропрограммируемых и мультипрцессорных систем использование типовых отладочных мониторов невозможно. Создание же мониторов для каждой конкретной системы дорого и является нетривиальной задачей.

Метод частичного макетирования предполагает использование в качестве типового ядра специальных аппаратно-программных отладочных комплексов, которые представляют собой универсальные цифровые системы с большим объемом ЗУ (включая внешнее ЗУ на магнитных носителях), широким набором внешних устройств, в том числе логических анализаторов, и резидентной системой разработки и отладки программного обеспечения. В отладочных комплексах предусматриваются специальные аппаратные и программные средства организации отладочных режимов, позволяющие выполнять останов в нужных точках программы или микропрограммы, следить за изменениями и модифицировать содержимое ячеек ЗУ и регистров типового ядра системы [8, 12, 14].

Существенно расширяет возможности отладочных комплексов наличие внутрисхемных эмуляторов — аппаратных блоков, включающих процессор, архитектурно эквивалентный используемому в отлаживаемой системе, и аппаратно реализованные схемы слежения за логическими сигналами на шине системы, которые подключаются к макету технических средств отлаживаемой системы вместо микропроцессора. Использование внутрисхемных эмуляторов позволяет проводить совместную отладку технических средств и программного обеспечения в реальном времени, а также подключать макет цифровой системы к обслуживаемому объекту для совместной отладки. При этом возможно поэтапное наращивание и отладка технических средств путем последовательной передачи функций от плат типового ядра, расположенного в отладочном комплексе, к добавляемым платам макета. В сочетании с логическими анализаторами аппаратно-программные отладочные комплексы являются мощным средством отладки цифровых систем.

В случае цифровых систем, использующих микропрограммирование, отладочный комплекс, как правило, включает ОЗУ, моделирующее ПЗУ микрокоманд, а также универсальную ЭВМ для управления работой комплекса.

Наличие типового блока микропрограммного управления позволяет создать системное программное обеспечение, обеспечивающее организацию отладочных режимов выполнения микропрограмм и слежения за логическими сигналами на шине системы, что в случае отладки микропрограммируемых систем особенно необходимо. Однако такие комплексы могут использоваться при отладке микропрограммируемых цифровых систем только со стандартной структурой шин и стандартным протоколом обмена информацией по шинам, а также типовой микроархитектурой, что ограничивает применение отладочных комплексов для микропрограммируемых систем типовыми или близкими к типовым системами.

В случае мультипроцесорных систем аппаратно-программные отладочные комплексы также используются. Ряд известных аппаратно-программных отладочных комплексов имеют внутрисхемные эмуляторы, позволяющие одновременно следить за работой двух и более процессорных блоков, входящих в сосредоточенную мультипроцесорную систему.

Применение аппаратно-программных отладочных комплексов, используемых одновременно как стандартное ядро с встроенными средствами организации отладочных режимов и как автоматизированное рабочее место для разработки программного и, в ряде случаев, микропрограммного обеспечения, является одним из основных факторов, обусловливающих широкое применение метода частичного макетирования при отладке цифровых систем.

При использовании для отладки *метода моделирования* слежение как за изменениями содержимого ячеек ЗУ и внутренних регистров всех блоков системы, так и за логическими значениями сигналов во всех узлах схемы не вызывает трудностей, так как они представляются переменными в моделях. Удобство задания любых отладочных режимов, условий вывода информации и самой информации о функционировании модели определяется организацией системы моделирования и языком управления ее работой.

В последние годы в системах моделирования цифровых систем нашли применение методы работы с большими данными, т. е. не выделение наиболее значимых переменных при моделировании и поиске ошибок, а запоминание при моделировании системы на уровне регистровых передач всех данных независимо от их значимости. При этом при обнаружении ошибки в каких-либо переменных — логических сигналах или содержимым регистров — возможен обратный просмотр хода моделирования для обнаружения причины ошибки [1].

При применении *метода моделирования с использованием реальных блоков* в качестве последних необходимо выбирать те блоки, слежение за внутренними регистрами которых не является обязательным для отладки.

4. Внесение исправлений в проект

Внесение исправлений в текст программно-микропрограммного обеспечения, находящегося в ОЗУ, осуществляется обычными способами и трудностей не вызывает. Внесение же изменений в схему технических средств при методе полного или частичного макетирования требует физического изменения схемы: изменения подключения блоков к узлам схемы, удаления, добавления или модификации блоков. Это требует определенных временных затрат, особенно при существенных изменениях в схеме. Внесение изменений в моделируемую схему технических средств выполняется путем редактирования машинной модели разработчиком в интерактивном режиме и практически не требует временных затрат.

5. Отладочные тесты и определение правильности выходной реакции

Выбор набора отладочных тестов и определение правильности выходной реакции является общей задачей для всех методов отладки цифровых систем.

Задача выбора отладочных тестов существенно отличается от задачи выбора проверяющих и диагностических тестов цифровой аппаратуры, используемых на этапе производства и обслуживания, тем, что при отладке цифровых систем на этапе проектирования отсутствуют модели возможных ошибок. Задача выбора отладочных тестов как для технических средств, так и для цифровых систем в целом по своей постановке в определенной степени близка задаче синтеза тестовых примеров для отладки программ. Отладочные тесты должны выбираться в первую очередь исходя из функций, выполняемых цифровой системой и определенных техническим заданием на ее разработку, а также, возможно, структуры разработанной системы [19—22].

Независимо от метода выбора тестов при отладке необходимо определять правильность

Сравнительный анализ методов отладки проектов цифровых систем

Показатель	Метод полного макетирования	Метод частичного макетирования	Метод моделирования	Метод моделирования с использованием физических блоков
Класс отлаживаемых систем	Любые	Системы, имеющие типовое ядро	Любые	Системы, имеющие выбранные блоки
Организация отладочных режимов	Затруднена	Проста, исключая слежение за регистрами нетиповой части	Проста	Проста, исключая слежение за регистрами физических блоков
Трудоемкость внесения изменений в схему	Высока	Высока	Низка	Низка
Возможность отладки проекта до изготовления макета	Нет	После изготовления макета нетиповых блоков	Да	Да
Стоимость	Высокая	Средняя	Низкая	Низкая
Длительность, включая время изготовления макета, если необходимо	Большая	Средняя	Малая	Малая

или ошибочность реакции объекта отладки. Наиболее удобным, но не всегда возможным в практике проектирования является случай, когда разработчик в точности знает требуемую реакцию объекта отладки на любой тест. Часто разработчик заранее в точности не знает правильную реакцию на отладочный тест, но после получения результата и, возможно, анализа промежуточных данных может сказать, соответствует ли полученная реакция объекта отладки требуемой. В любом случае возможность разработчика определить правильность реакции объекта отладки на поданное входное воздействие является необходимым условием проведения отладки вообще.

Таким образом, на основе анализа методов отладки цифровых систем (см. таблицу) можно сказать, что потенциально только метод моделирования на ЭВМ является универсальным относительно структуры и архитектуры отлаживаемых цифровых систем. Этот метод наиболее широко применяется при разработке современных цифровых систем в микроэлектронном исполнении, однако и другие методы также находят применение.

Список литературы

1. **Kishore Karnane, Corey Goss.** Automated Root-Cause Analysis to Reduce Time to Find Bugs by Up to 50 %. URL: www.cadence.com.

2. **Вишнеков А. В., Ерохин В. В., Иванова Е. М.** Верификация СНК: выбор стратегии // Нано- и микросистемная техника. 2014. № 12. С. 30—36.

3. **Слинкин Д. И.** Анализ современных методов тестирования и верификации проектов сверхбольших интегральных схем // Программные продукты и системы. 2017. Т. 30, № 3. С. 401—408.

4. **Иванников А. Д., Стемпковский А. Л.** Формализация задачи отладки проектов цифровых систем // Информационные технологии. 2014. № 9. С. 3—10.

5. **Гаращенко А. В., Николаев А. В., Путря Ф. М., Сардарян С. С.** Система комбинируемых специализированных генераторов тестов для нового поколения VLIW DSP процессоров с архитектурой Elcore50 // Проблемы разработки перспективных микро- и наноэлектронных систем (МЭС). 2018. № 2. С. 9—15.

6. **Ilias Alexopoulos.** How to Debug Embedded Systems. URL: https://www.EDN.com (date of access Dec. 11, 2012).

7. **Жезлов К. А., Колбасов Я. С., Козлов А. О., Николаев А. В., Путря Ф. М., Фролова С. Е.** Автоматизация процесса создания тестовых окружений, обеспечивающая сквозной маршрут разработки, верификации и исследования СФ-блоков и СНК // Проблемы разработки перспективных микро- и наноэлектронных систем (МЭС). 2016. № 2. С. 46—53.

8. **Považan I., Krnjetin M., Četić N.** Communication interface libraries as an extension to the debugging framework for DSP applications // *2015 23rd Telecommunications Forum Telfor (TELFOR).* Belgrade. 2015. P. 1005—1008.

9. **Абрамов Е. М., Егоров А. В., Козлов А. О., Поперечный П. С., Путря Ф. М., Фролова С. Е.** Выбор платформ прототипирования для СФ-блоков и подсистем СНК // Вопросы радиоэлектроники. 2017. № 8. С. 76—83.

10. **Зотов В.** Инструментальные средства разработки и отладки цифровых устройств и встраиваемых микропроцессорных систем, проектируемых на основе ПЛИС FPGA фирмы XILINX серии Kintex-7 // Компоненты и технологии. 2012. № 4 (129). С. 124—132.

11. **Hu X., Jin Y., Li Z.** A Parallel JTAG-based Debugging and Selection Scheme for Multi-core Digital Signal Processors // 2018 IEEE International Conference of Safety Produce Informatization (IICSPI). Chongqing, China. 2018. P. 527—530.

12. **Иванов С. А., Орешкин Д. М.** Аппаратно-программный комплекс для разработки и отладки электронных систем на цифровых микропроцессорных модулях // Новая наука: от идеи к результату. 2016. № 8-1 (96). С. 27—29.

13. **Klinachev N. V., Shaburov P. O.** Technique for debugging of the data exchange between the PC and microprocessor-controlled electromechanical systems based on the modbus RTU protocol // *2017 International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM).* St. Petersburg. 2017. P. 1—5.

14. **Стрелец А. И., Протопопова Ю. Д., Иванников В. С.** Универсальная система для тестирования цифровых устройств // Научный журнал. 2018. № 5 (28). С. 41—44.

15. **Lin Yi-Li, Su Alvin W. Y.** Functional Verification for SoC Software/Hardware Co-Design: From Virtual Platform to Physical Platform // 2011 IEEE International SOC Conference (SOCC). 2011. P. 201—206.

16. **Shi Jin, Liu Weichao, Jiang Ming** el al. Software Hardware Co-Simulation and Co-Verification in Safety Critical System Design // 2013 IEEE International Conference on Intelligent Rail Transportation (ICIRT). 2013. P. 71—74.

17. **Nguen M. D.** Hardware/software formal co-verification using hardware verification techniques // Fourth Int. Conf. on Communications and Electronics (ICCE). 2012. P. 465—470.

18. **Иванников А. Д.** Теоретические основы выбора множества отладочных тестов проектов цифровых систем на основе алфавита выполняемых функций // Информационные технологии. 2019. Т. 25, № 11. С. 657—662.

19. **Камкин А. С., Коцыняк А. М., Смолов С. А., Татарников А. Д., Чупилко М. М., Сортов А. А.** Средства функциональной верификации микропроцессоров // Сб. Трудов Института системного программирования РАН. 2014. Т.26, № 1. С. 149—200.

20. **Мешков А. Н., Рыжов М. П., Шмелев В. А.** Развитие средств верификации микропроцессора "ЭЛЬБРУС" // Вопросы радиоэлектроники. 2014. Т. 4, № 3. С. 5—17.

21. **Иванников А. Д.** Формирование отладочного набора тестов для проверки функций цифровых систем управления объектами // Мехатроника, автоматизация, управление. 2017. Т. 18, № 12. С. 795—801.

22. **Иванников А. Д.** Автоматизация генерации отладочных тестов для проектов цифровых систем управления объектами // Мехатроника, автоматизация, управление. 2018. Т. 19, № 12. С. 770—776.

A. D. Ivannikov, Doctor of Tech.Sc., e-mail: ADI@ippm.ru,
Institute for Design Problems in Microelectronics of Russian Academy of Sciences

Comparative Analysis of Methods for Digital Systems Debugging at the Design Stage

The paper discusses debugging methods, that is, identifying and correcting errors in projects of digital systems specified by the hardware diagram and the text of software or firmware. Debugging is considered on a fabricated layout of hardware and on a software model of a digital system, as well as on a combination of the layout of individual blocks and a software model of the rest of the system. The methods of organizing debugging modes for all considered methods are analyzed. It is indicated that for digital systems on a chip, the most effective method is to debug projects using a software model.

Keywords: *design of digital systems, design debugging, organization of debugging modes*

DOI: 10.17587/it.26.259-266

References

1. **Kishore Karnane, Corey Goss.** Automated Root-Cause Analysis to Reduce Time to Find Bugs by Up to 50 %, available at: www.cadence.com.

2. **Vishnekov A. V., Erohin V. V., Ivanova E. M.** SoC Verification: Choice of Strategy, *Nano- i Microsystemnaya Technika*, 2014, no. 12, pp. 30—36 (in Russian).

3. **Slinkin D. I.** Analysis of Modern VLSI Project Testing and Verification Methods, *Programmnie Produkti i Systemi*, 2017, vol. 30, no. 3, pp. 401—408 (in Russian).

4. **Ivannikov A. D., Stempkovsky A. L.** Formal Model of Digital System Design Debugging Task, *Informacionnie Technologii*, 2014, no. 9, pp. 3—10 (in Russian).

5. **Garashchenko A. V., Nikolaev A. V., Putrya F. M., Sardaryan S. S.** of Combined Specialized Test Generators for the New Generation of VLIW DSP Processors with Elcore50 Architecture, *Problems of Perspective Micro- and Nanoelectronic Systems Development — 2018*, 2018, iss. 2, pp. 9—15 (in Russian).

6. **Ilias Alexopoulos**. How to Debug Embedded Systems, available at: https://www.EDN.com (date of access Dec. 11, 2012).

7. **Zhezlov K. A., Kolbasov Y. S., Kozlov A. O., Nikolaev A. V., Putrya F. M., Frolova S. E.** Automation of verification environments development process providing a through design flow for design, verification and research of IP-blocks and SoC, *Problems of Perspective Micro- and Nanoelectronic Systems Development — 2016*, 2016, iss. 2, pp. 46—53 (in Russian).

8. **Považan I., Krnjetin M., Četić N.** Communication interface libraries as an extension to the debugging framework for DSP applications, *2015 23rd Telecommunications Forum Telfor (TELFOR)*, Belgrade, 2015, pp. 1005—1008.

9. **Abramov E. M., Egorov A. V., Kozlov A. O., Poperechniy P. S., Putrya F. M., Frolova S. E.** Prototype Platform Choosing for IP Blocks and SoC Subsystems, *Voprosi Radioelectroniki*, 2017, no. 8, pp. 76—83 (in Russian).

10. **Zotov V.** Design and Debugging Means for Digital Devices and In-Circuit Microprocessor Systems on the Bases of XILINX FPGA Kintex-7 Series, *Komponenti i Technologii*, 2012, no. 4 (129), pp. 124—132 (in Russian).

11. **Hu X., Jin Y., Li Z.** A Parallel JTAG-based Debugging and Selection Scheme for Multi-core Digital Signal Processors, *2018 IEEE International Conference of Safety Produce Informatization (IICSPI)*, Chongqing, China, 2018, pp. 527—530.

12. **Ivanov S. A., Oreshkin D. M.** Hardware-Software Complex for Development and Debugging of Electronic Systems Based on Digital Microprocessor Modules, *Novaya Nauka: ot Idei kResultatu*, 2016, no. 8-1 (96), pp. 27—29 (in Russian).

13. **Klinachev N. V., Shaburov P. O.** Technique for debugging of the data exchange between the PC and microprocessor-controlled electromechanical systems based on the modbus RTU protocol, *2017 International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM)*, St. Petersburg, 2017, pp. 1—5.

14. **Strelec A. I., Protopopova U. D., Ivannikov V. S.** Universal System for Digital Device Testing, *Nauchniy Jurnal*, 2018, no. 5 (28), pp. 41—44 (in Russian).

15. **Lin Yi-Li, Su Alvin W. Y.** Functional Verification for SoC Software/Hardware Co-Design: From Virtual Platform to Physical Platform, *2011 IEEE International SOC Conference (SOCC)*, pp. 201—206.

16. **Shi Jin, Liu Weichao, Jiang Ming** el al. Software Hardware Co-Simulation and Co-Verification in Safety Critical System Design, *2013 IEEE International Conference on Intelligent Rail Transportation (ICIRT)*, pp. 71—74.