ПАКЕТ JTAG ProVision
РАЗРАБОТКА ПРИЛОЖЕНИЙ ПЕРИФЕРИЙНОГО СКАНИРОВАНИЯ

А.Иванов alexey@jtag.com

В процессе разработки цифровых электронных устройств необходимо обеспечить возможность тестирования как отдельных компонентов схемы, так и всего устройства, а также максимально упростить этот процесс при серийном производстве изделий. Программный пакет JTAG ProVision позволяет создавать тестовые приложения любой сложности для устройств с микросхемами, поддерживающими стандарт JTAG.

Технология периферийного сканирования JTAG (boundary-scan) позволяет тестировать и диагностировать электронные цифровые устройства и программировать установленные на платы ПЗУ и ПЛИС. Сегодня периферийное сканирование поддерживается множеством цифровых микросхем, совместимых со стандартами группы IEEE 1149.x (IEEE 1149.1, IEEE 1149.4 и IEEE 1149.6) – микроконтроллерами, ЦСП, ПЛИС, телекоммуникационными ИС и др.

О принципах работы JTAG и преимуществах его использования разработчики цифровой электроники могут узнать из многочисленных публикаций и семинаров. Поэтому далее будет лишь вкратце описана суть метода, а основное внимание уделено процессу разработки и выполнения приложений для тестирования и программирования плат на примере средств компании JTAG Technologies. Наибольшее число вопросов сегодня вызывает именно тема практической реализации метода периферийного сканирования.

Предположим, что есть плата цифрового устройства, на которой установлена одна или несколько микросхем с поддержкой стандарта периферийного сканирования IEEE 1149.1. Для управления тестовой логикой у этих микросхем имеется интерфейс JTAG, который для удобства тестирования и программирования устройства должен быть выведен на внешний разъем или, в крайнем случае, на контактные площадки. Порой случается, что на плате слишком много компонентов с JTAG, и даже, несмотря на возможность последовательного их объединения в цепочки сканирования, приходится использовать несколько внешних разъемов JTAG. Современные контроллеры периферийного сканирования (назовем их JTAG-тестерами) имеют нескольких JTAG-интерфейсов, что позволяет им подключаться одновременно ко всем портам. Если микросхема поддерживает стандарт IEEE 1149.1, то с помощью JTAG-интерфейса мы можем отключать ее основные функции и использовать ее выводы как тестовые, данные на которых выставляются и считываются через подключенный к ним независимый регистр периферийного сканирования. Таким образом, с помощью этих компонентов можно диагностировать неисправности на следующих участках:

- связи между самими JTAG-компонентами, цепи, идущие напрямую через связующую логику или на внешние разъемы (для тестирования цепей разъемов JTAG-тестер можно оснастить дополнительным модулем IO) либо неподключенные;

- связи JTAG-компонентов с микросхемами ОЗУ и ПЗУ. Для этого используются функциональные модели памяти, по которым система тестирования генерирует записывающие и считывающие последовательности. В тестировании задействуются подключенные к памяти JTAG-компоненты;
- связи JTAG-компонентов с различной функциональной логикой (так называемыми "кластерами", т.е. компонентами или узлами без поддержки периферийного сканирования). Для такого рода тестов также используются функциональные модели этих кластеров.

В дополнение к операциям тестирования JTAG позволяет программировать следующие типы устройств:

- флеш-ПЗУ (NAND, NOR, ПЗУ с последовательным интерфейсом). Управление контрольными сигналами и записываемыми данными ведется с окружающих JTAG-компонентов;
- ПЛИС. Используется JTAG-интерфейс самой ПЛИС.

Логично, что перечисленные выше приложения необходимо сначала создать, затем отладить и только после этого запускать на серийном

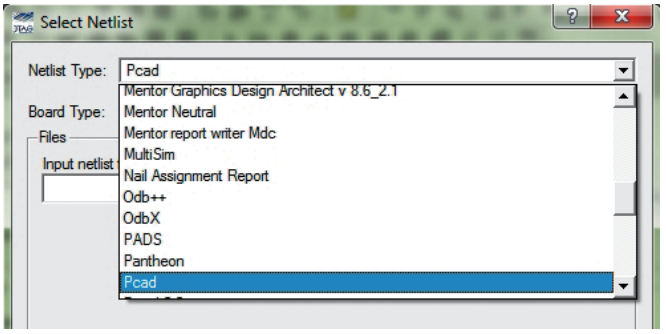

Рис.1. Выбор типа САПР, в которой разрабатывалась схема изделия

продукте. Что ж, взглянем на то, как весь этот процесс происходит на практике. В качестве примера будем использовать программный пакет JTAG ProVision компании JTAG Technologies.

Сразу отметим, что современные средства создания приложений периферийного сканирования основываются на автоматической генерации тестовых векторов (ATPG – Automatic Test Pattern Generation). Тестовые векторы (или паттерны) – это последовательности сигналов, которые во время

Рис.2. Менеджер моделей компонентов JTAG ProVision

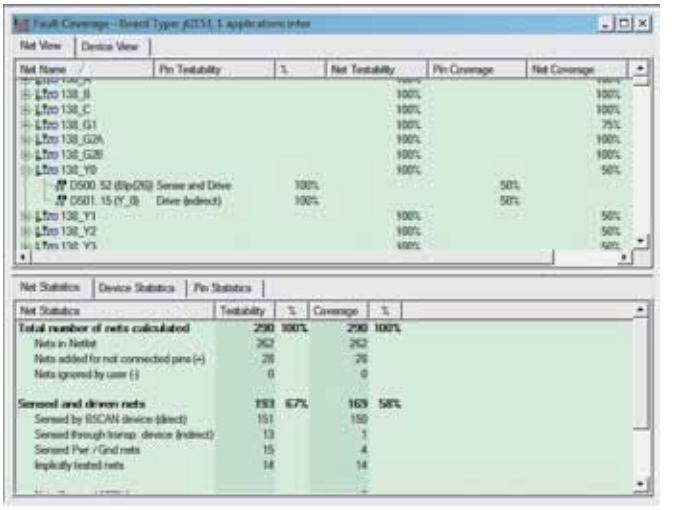

Рис.3. Расчет тестового покрытия

теста или операции программирования выставляются на выводах компонентов, поддерживающих периферийное сканирование, или считываются этими компонентами с окружающих цепей. Как выставляемые, так и ожидаемые векторы создаются в JTAG ProVision автоматически на основе анализа схемы изделия, JTAG-компонентов (а точнее их части, отвечающей за периферийное сканирование) и других компонентов и узлов платы, которые имеют некоторую функциональность и могут воздействовать на тестовые векторы или изменять их.

Пакет JTAG ProVision имеет проектную структуру. Один проект может содержать свою базу моделей компонентов, несколько плат, разные виды приложений для тестирования и программирования и варианты их последовательностей. Создание проекта начинается с ввода исходных данных, в первую очередь, списков соединений (netlist) плат, входящих в проект и участвующих в тестировании. В программу уже встроены конверторы для файлов со списком соединений всех возможных САПР и универсальных трассировочных форматов (рис.1).

После конвертации в программе имеется информация о компонентах тестируемого устройства и их связях между собой. Но пока что информация о компонентах – это всего лишь их названия и информация о выводах, не несущие никакой функциональности. Поэтому необходимо определить для них модели. Для JTAG-компонентов это BSDL-файлы, описывающие их архитектуру периферийного сканирования: регистры, команды и т.д. BSDL является универсальным языком, описанным в стандарте IEEE 1149.1 и его надстройках, BDSL-файл с описанием той или иной микросхемы можно загрузить, например, с сайта ее производителя или других общедоступных ресурсов. Для остальных элементов платы (вплоть до резисторов и разъемов), не поддерживающих периферийное сканирование, назначаются модели, которые в основном берутся из готовой библиотеки ProVision, содержащей десятки тысяч функциональных и пассивных моделей. Может возникнуть справедливый вопрос: для чего нужны пассивные модели, например, для конденсаторов или разъемов? Ведь модель логической ИС содержит таблицу истинности, учитываемую при создании тестовых векторов, модель ОЗУ описывает циклы чтения и записи для автоматического создания теста памяти, но пассивный компонент не несет никакой функциональности. Дело в том, что если для компонента не определена модель (даже если он пассивный), то ProVision будет генерировать тестовые векторы, не затрагивающие цепи, связанные с этим "неизвестным" компонентом, опасаясь возможных конфликтов драйверов. Определив элемент как "пассивный", мы разрешаем системе "вмешиваться" в его связи. Установка соответствия моделей компонентам платы производится с помощью менеджера компонентов (рис.2). Следует отметить, что библиотечные модели микросхем содержат сразу несколько типов корпусов, которые можно выбирать в самом менеджере. Модели можно определять с помощью функции автоматического поиска по библиотекам (в случае, если они имеют корректные названия в электронной схеме) или вручную.

Таким образом вводится вся необходимая информация о цепях платы и компонентах,

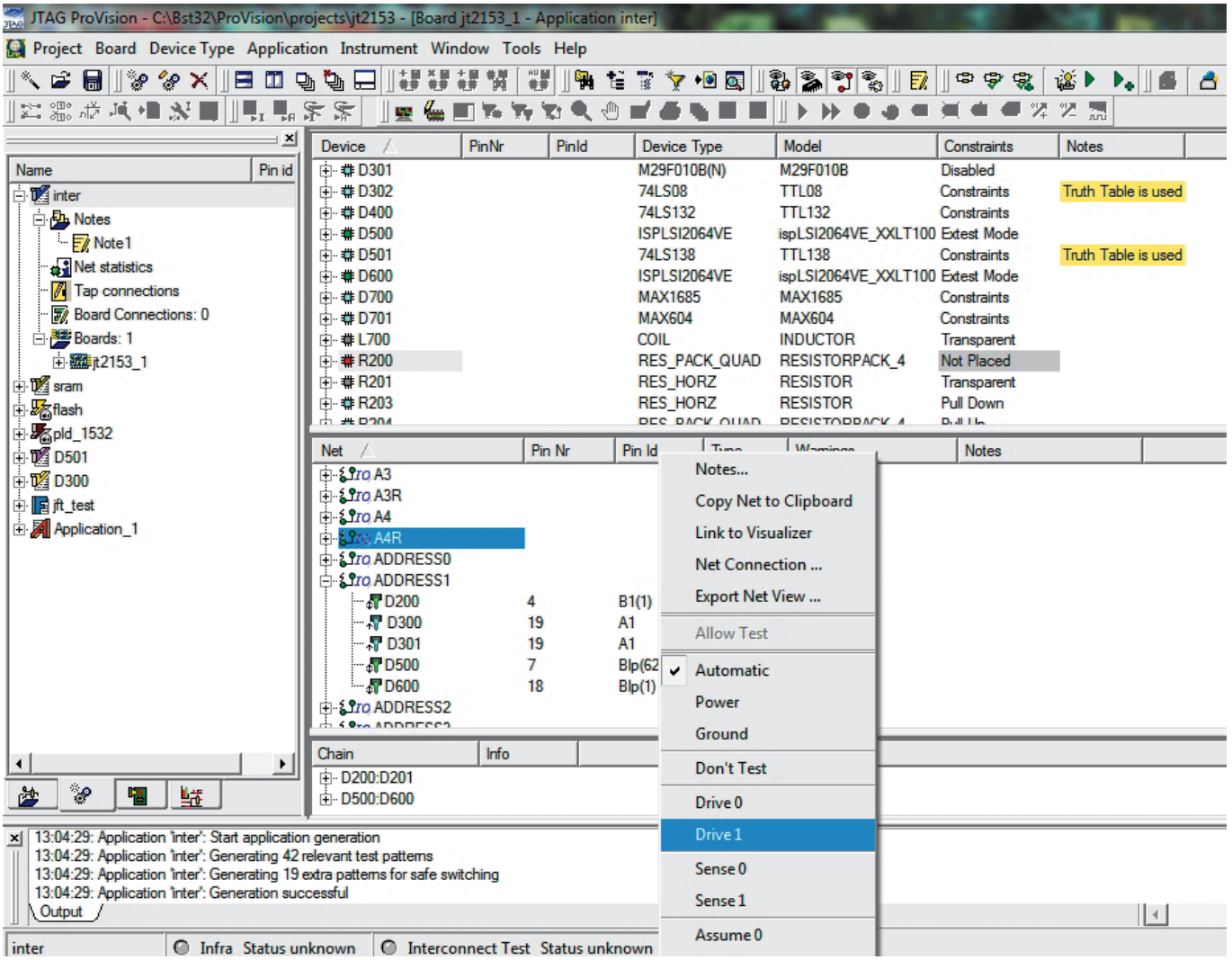

Рис.4. Редактирование установок сгенерированного теста

которая необходима для периферийного сканирования. Этих данных достаточно для создания полноценного набора тестовых векторов с требуемыми тестовым покрытием и уровнем безопасности (о нем будет рассказано ниже). Однако прежде чем создавать приложения для тестирования и программирования, инженер уже может воспользоваться встроенной утилитой анализа тестового покрытия (рис.3). На рисунке видно, что развернутый отчет о покрытии по каждой из цепей содержит две колонки: максимально возможное тестовое покрытие (теоретическое) и достигнутое с помощью создания тех или иных тестов. При генерации такого отчета непременно используются все составляющие исходной информации: межсоединения, BSDL-модели и функциональные модели. Это важно, так как многие связи компонентов с поддержкой периферийного сканирования с микросхемами ОЗУ, ПЗУ, внешними разъемами проходят через связующую логику (буферы, мультиплексоры и т.д.), которая без наличия соответствующих моделей не будет являться для системы "прозрачной", и тестовое покрытие будет рассчитано не корректно. Исследуя отчет

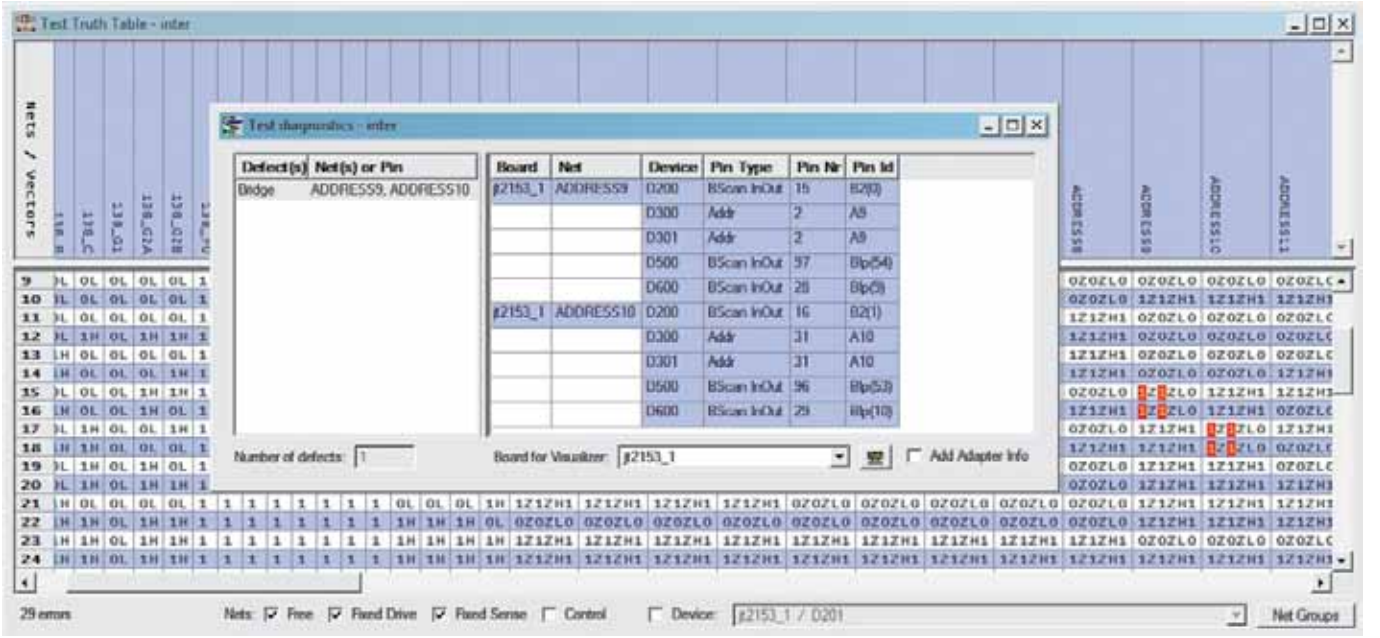

Рис.5. Результаты тестирования и диагностика неисправностей

о тестовом покрытии, тест-инженер может увидеть не только недостатки изделия, допущенные при проектировании и приводящие к потере тех или иных участков платы при тестировании, но и полноту собственноручно созданных тестов: какие модели еще необходимо определить, какие приложения создать и к каким разъемам применить дополнительный IO-модуль. По мере создания приложений периферийного сканирования вторая колонка отчета будет заполняться процентами полученного фактического тестового покрытия. При этом система дает подсказки, какие виды приложений требуется дополнительно создать для достижения максимального покрытия.

Приложения для тестирования и программирования также создаются при помощи мастера. Пользователю предлагается выбрать тип будущего приложения – тест межкомпонентных связей, тест ОЗУ и ПЗУ, логики и соединений между платами, высокоскоростных цифровых цепей, поддерживающих стандарт IEEE 1149.6, приложения для программирования ПЛИС и различных видов флеш-памяти. При выборе, к примеру, теста межкомпонентных связей, пользователю сперва будет предложено выбрать параметры тестовых векторов, которые будут сгенерированы позже. Плата может содержать тысячи цепей и компонентов, десятки ИМС с поддержкой периферийного сканирования, поэтому количество тестовых векторов и их объем могут быть достаточно большими. Пользователю предлагается решить, нужно ли тестировать неподключенные выводы JTAG-компонентов, использовать ли все драйверы и сенсоры выводов компонентов на сложных шинах (в случае, когда одна цепь идет на две и более ИМС с поддержкой периферийного сканирования), какие типы дефектов требуется диагностировать с помощью тестовых векторов и т.д. От всех этих установок зависят размеры и количество генерируемых тестовых векторов. Нужно отметить, что даже с максимально расширенными параметрами локализации дефектов время тестирования при выполнении периферийного сканирования межсоединений невелико (единицы – десятки секунд), поэтому многие пользователи ставят "галочки" практически на всех опциях за исключением особых случаев, для которых эти опции собственно и создавались.

Вероятно, в этот момент у читателя появился вопрос: а можно ли редактировать созданное приложение? Конечно, при генерации векторов система ProVision автоматически анализирует связи электрической схемы, определяет цепочки сканирования, отключает потенциально опасные компоненты, находит резисторы подтяжки и т.д. Но в реальной жизни часто возникают такие ситуации, как появление неучтенных в электронной версии схемы доработок, версии изделия без каких-то компонентов или сигналы, подающиеся через внешние разъемы (в случае, если тестируемое изделие работает в составе стенда или блока). В JTAG ProVision предусмотрена возможность

учета подобных ситуаций, причем такие изменения можно установить как для всего проекта, так и для каждого приложения в отдельности (для теста межсоединений, памяти, логики, программирования ПЗУ и т.д.). Это делается не на уровне тестовых векторов, где можно увидеть только логические "0" и "1", а в графической оболочке, где отображаются компоненты, цепи со всеми подходящими к ним выводами компонентов и каналы сканирования с соответствующими JTAG-сигналами (рис.4). Таким образом, после автоматической генерации приложения можно выбрать для каждой отдельной цепи такие установки, как Don't Test, Drive 0/1, Sense 0/1, описать неучтенные в схеме проводные соединения, установить для компонентов атрибуты Not Placed (Не установлен), Assumed Passive (Считать пассивным) и т.д. Например, если для какой-либо цепи выбрать Drive 1, то во всех тестовых векторах в итоге на данной цепи будет выставляться логическая "1", тогда как на остальных цепях будут перебираться различные комбинации данных для автоматической диагностики дефектов.

Особо следует отметить, что все модели компонентов платы учитываются при генерации даже теста межсоединений и окружающих цепей JTAG-компонентов. Если, к примеру, модель какого-нибудь устройства содержит, кроме его функций, информацию о возможности отключения (сигналы OE, CS и т.д.), то такие компоненты отключаются автоматически – программа сама выставляет нужные атрибуты Drive 0/1 или Don't Test. Это видно на рис.3: напротив компонента D301 (флеш-память) стоит атрибут Disabled. Таким образом, львиную долю работы по безопасности тестовых векторов берет на себя среда JTAG ProVision, во-первых, обходя стороной неизвестные компоненты, для которых не описана модель, во-вторых, автоматически отключая активные устройства при наличии такой возможности. В случае, когда инженер знает, что какой-либо компонент неактивен, но для него по ряду причин не определена модель, можно для данного конкретного компонента установить атрибут Assumed Passive.

После того, как пользователь внес необходимые изменения в автоматически сгенерированный

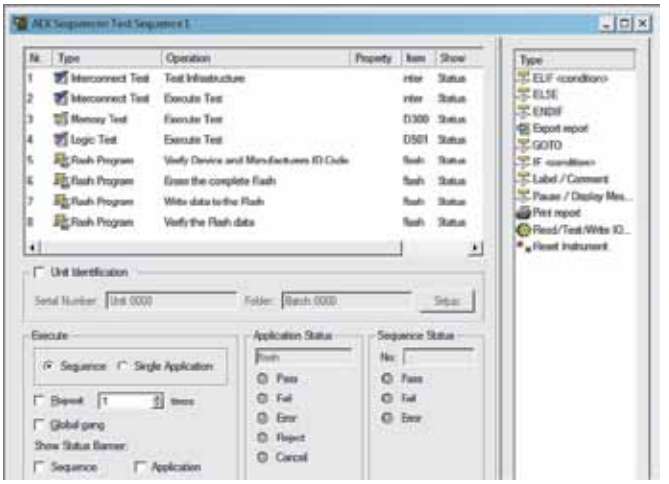

Рис.6. Последовательность приложений для тестирования и программирования

тест, его можно запускать, подключив тестируемую плату по JTAG-интерфейсу. Таблицу с тестовыми векторами можно увидеть уже после запуска теста, и если в каких-то из них получены неправильные результаты, то диагностический программный модуль проанализирует их и выдаст текстовую информацию о неисправности (рис.5).

По похожему сценарию генерируются, редактируются и запускаются остальные виды приложений, разница лишь в том, что приложения для тестирования памяти или логики (т.е. компонентов без поддержки периферийного сканирования) используют для генерации функциональности, описанной в их моделях. Если модели определены в должном объеме для всех значимых компонентов устройства, то процесс генерации теста также прост: выбирается тип приложения (тест памяти, тест логики и т.д.), сам тестируемый компонент из списка, который формируется в зависимости от наличия в проекте моделей, и "нажимается" кнопка генерации. Редактирование установок осуществляется так же, как и в описанном выше примере с тестом межсоединений. Установки для цепей и компонентов могут быть индивидуальными для каждого приложения, а могут быть скопированы.

Итак, создание приложений и их отладка на тестируемом изделии закончены. Пока что они существуют отдельно. Очевидно, что для проведения тестов и процедур программирования на производстве необходимо создать единую последовательность, запускаемую одним нажатием кнопки. Для этого в JTAG ProVision есть встроенный секвенсор, где можно задавать "тестовый план" – очередность выполнения всех приложений периферийного сканирования при нажатии кнопки Start (рис.6). В зависимости от результатов того или иного теста можно делать переходы к различным участкам последовательности, используя операторы IF, GOTO, Display Message и др. Такую последовательность можно запускать как в полноценной среде разработки приложений JTAG ProVision, так и на отдельной производственной станции на базе ProVision Platform, содержащей только секвенсор без возможности генерации и редактирования.

Опишем вкратце некоторые другие сопутствующие возможности среды разработки JTAG ProVision. Во время генерации теста межсоединений система автоматически создает векторы для максимально возможного числа цепей и выводов компонентов. У читателя может возникнуть вопрос: что делать, если необходимо протестировать всего лишь несколько цепей на плате? Например, может потребоваться зажечь светодиоды на плате для проверки их работы при помощи векторов периферийного сканирования, выставляемых с JTAG-компонентов. Для таких целей существует интерактивный графический редактор ActiveTest, встроенный в ProVision. Пользователь может выбрать одну или несколько групп цепей и создать для них вручную любое количество тестовых векторов с необходимыми комбинациями данных. Таким образом, использование ActiveTest – это еще один метод создания векторов, альтернативный автоматической генерации. Приложения, созданные с помощью ActiveTest, можно затем встраивать в производственные тестовые последовательности наряду с остальными.

Еще одна альтернатива автоматической генерации – программная среда JTAG Functional Test (JFT). С ее помощью можно создавать приложения для сложных кластеров, для которых не подходят описания в виде однозначной модели. Обычно при генерации тестов кластеров их алгоритмы базируются на однозначных функциональных моделях компонентов (как, например, таблицы истинности логических ИС) и не изменяются в процессе тестов. Но в случае, если кластер представляет собой сложное устройство с коммуникационным протоколом и/или разными режимами работы, его невозможно описать в виде статичной таблицы истинности. Именно для таких случаев программный пакет JTAG ProVision был дополнен встроенной средой программирования на базе языка Python. Работая в JFT, программист получает абсолютную свободу творчества, так как в этом случае у него есть возможность написать тестовый алгоритм любой сложности, применив, например, функции, в качестве переменных использующие выводы JTAG-компонентов, с операциями сравнения, ветвления и т.д.