УДК *621.3.049.771.14*

### Опыт сопряжения системы непрерывной интеграции Jenkins и ПО Cadence vManager для верификации СБИС «Система-на-кристалле»

Андрианов Андрей Владимирович

*andrianov@module.ru*

*ЗАО НТЦ «Модуль»*

*125190, г. Москва, а/я 166*

*Тел. +7 (495) 531-30-80*

[*http://www.module.ru*](http://www.module.ru)

**Аннотация** — В статье рассмотрены особенности совместного использования платформы непрерывной интеграции Jenkins совместно с решением Cadence vManager для обеспечения сбора статистической информации о состоянии проекта СБИС в ходе регрессионного тестирования.

### *Ключевые слова* — верификация, тестирование, СБИС, система-на-кристалле, MDV, vManager.

### Combining Jenkins CI and Cadence vManager for SoC project verification

Andrianov Andrew Vladimirovich

*andrianov@module.ru*

*JSC RC «Module»*

*P.O. Box 166, Moscow, Russia, 125190*

[*http://www.module.ru*](http://www.module.ru)

***Abstract –*** This paper shares experience of integrating Jenkins, a popular CI solution and Cadence vManager to automate metrics collection during verification process of “System-on-Chip” projects.

***Key words: verification, testing, VLSI, System-on-Chip, vmanager, jenkins, ci***

## Введение

В связи с ростом сложности современных СБИС класса “Система-на-Кристалле”, а так же объемами верификации становится все сложнее спрогнозировать сроки окончания процесса верификации и тестирования, а также оценить объемы работ. Количество пройденных и упавших тестов при условии регулярного запуска регрессионного тестирования могут дать лишь примерную картину при условии регулярного автоматизированного регрессионного тестирования. Для того, чтобы получить более точную информацию о том, в каком состоянии находится проект СБИС необходимо в ходе регрессионного тестирования собирать и анализировать ряд метрик, которые позволяют более точно сделать заключение о том, в каком состоянии находится проект.

К этим метрикам можно отнести процент покрытия RTL-кода тестами, процент переключения логики в ходе выполнения тестов, процент покрытия программного кода тестами и т.п.

В статье описан опыт совмещения системы контроля версий git (на платформе gitlab), системы непрерывной интеграции Jenkins CI и продукта Cadence vManager для автоматизированного тестирования, сбора, и анализа метрик и сопоставления с планом верификации.

**Сравнение возможностей Gitlab, Jenkins и vManager**

Прежде чем описывать особенности взаимодействия трех программных решений, необходимо кратко рассмотреть, как и для каких целей, они позиционируются разработчиками, а так же их самые сильные стороны, которые хотелось бы использовать.

**Gitlab –** Изначально построен вокруг системы контроля версий git, который позиционируется как полноценная платформа, представляющая полный цикл управления программными проектами: хранение и совместный доступ к исходному коду, сборка, тестирование, планирование и фиксация задач. Имеет клиент-серверную архитектуру, где клиентом выступает git. *Сильные стороны:* Управление репозиториями с исходным кодом.

**Jenkins –** Платформа ориентирована на автоматизированную сборку и тестирование программного обеспечения. Сервер имеет модульную архитектуру и интеграцию со всеми современными системами контроля версий. *Сильные стороны:* Наиболее гибкое описание процесса тестирования и хорошая интеграция с системами контроля версиями

**Cadence vManager –** Платформа сочетает в себе инструмент для составления плана верификации, инструменты для сбора и анализа покрытия кода, совместно с интерфейсом для запуска регрессионного тестирования на серверах используя свой собственный формат описания. vManager имеет клиент-серверную архитектуру, предоставляет графический клиент и веб-интерфейс пользователя. *Сильные стороны:* Инструменты для задания верификационного плана, сбор и анализ покрытия по RTL коду СБИС и сопоставление с верификационным планом.

Так как функционал этих программных продуктов частично пересекается, то для наглядности их сравнение приведено сводно в Таблице 1.

|  |  |  |  |
| --- | --- | --- | --- |
|  | gitlab | Jenkins | vManager |
| Интеграция с системами контроля версий | Да (продукт основан на упралении репозиториями с кодом) | Да (Получение данных из репозитория, связывание сборки с конкретным коммитом) | Отсутствует |
| Описание CI/CD | На основеописания в .gitlab-ci.yml, формат yaml | Графический интерфейс  Declarative Jenkinsfile  Scripted Jenkinsfile | Описание сессий через VSIF-файлы (формат NTF) |
| Балансировка нагрузки | Узлы для выполнения задач подключаются через gitlab-runner, возможно использование kubernetes/docker | Jenkins agent запускается на целевых узлах, узлы объединяются по тегам.  Дополнительные возможности балансировки через модули расширения | Поддерживает либо локальный запуск, либо запуск с параллелизацией на основе коммерческих grid-систем: OGE, LGE, OpenLava и т.п. |
| Визуализация тестового покрытия | Формат cobertura  Istanbul, coverage.py через встроенные просмотр кода.  Отображение HTML отчетов | Формат cobertura  Отображение HTML отчетов | Формирование HTML отчетов, Просмотр через встроенный интерейс. |
| Поддержка vManager vAPI | Нет | Да, Jenkins vManager Plugin | - |
| Запуск тестирования по событию | Внесение изменений в репозиторий (push), расписание, создание тега, вручную | Внесение изменений в репозиторий (push), расписание, создание тега, вручную | Вручную и по расписанию  Нет интеграции с системами контроля версий |
| Интеграция с системами управления проектами (JIRA, и т.п.) | Имеет встроенную системы. Поддерживает интеграцию с другими продуктами, такими как JIRA | Поддерживается интеграция с JIRA и другими подобными системами при помощи расширений | Через сторонний коммерческий продукт OpsHub |
| Лицензирование | Коммерческая (EE) и бесплатные (CE) редакции. Продукт с открытым исходным кодом. | Бесплатный продукт с открытым исходным кодом. Коммерческая поддержка. | Коммерческий продукт. |

Таблица Сравнение основных функций gitlab, jenkins и vManager

Как видно из таблицы, из всех трех систем только Cadence vManager обладает возможностью собирать, анализировать и отображать метрики, полученные в ходе моделирования RTL кода СБИС средствами симулятора Cadence XCellium, что фактически делает использование этого продукта безальтернативным (если, конечно, не рассматривать переход на другие инструменты моделирования СБИС). Однако, ряд его недостатков, таких как отсутствие интеграции с системами контроля версий, отсутствие поддержки сбора и отображения покрытия по программным компонентам и ограничения встроенного формата описания сценария тестирования делают использование без вспомогательных инструментов затруднительным.

При моделировании больших СБИС распределение нагрузки между серверами может существенно сократить время тестирования. Более того, на этапе сборки и на этапе непосредственно тестирования нагрузка на дисковую подсистему и на процессор сервера существенно различаются, что тоже необходимо учитывать.

vManager рассчитан на работу с системами распределения нагрузки, такими как OpenLAVA, OGE, LGE и т.п. Если такие системы уже не используются, то их развертывание потребует радикально поменять существующую инфраструктуру серверов для моделирования СБИС, а также приобрести лицензии на данные продукты. Такое радикальное изменение может сломать устоявшиеся процессы разработки, нарушить работу ПО, не рассчитанного на работу в таком окружении, что не всегда является удобным. Без использования этих систем vManager предоставляет интерфейсы запуска тестов локально, на том же компьютере, на котором запускается клиентское приложение vManager. Дополнительные накладные расходы при таком запуске теста через графический интерфейс vManager занимают на момент написания данной статьи около 40-55 секунд.

Для описания процесса тестирования (сессии, в терминологии vManager), используется формат NTF (Nested Text Format), который представляет собой высокоуровневое описание процесса тестирования. Формат позволяет описать процесс запуска инструментов моделирования в декларативном стиле без динамического формирования списка тестов.

При отображении результатов регрессии и при запуске сессий сервер vManager ожидает, что на всех узлах, включая узел, где развернут сервер vManager - единая файловая система.

При планировании нагрузки - что все узлы “равноправны”. Таким образом, нельзя привязать средствами только vManager какой-то этап сессии к конкретному серверу.

Вышеописанные особенности делают реализацию некоторых полезных практик, сокращающих время моделирования, затруднительными:

* Нет возможности привязать начальную сборку к конкретному серверу (Например, где есть прямой доступ к быстрому SSD-накопителю)
* Нет возможности динамически формировать список тестов и выстраивать порядок запуска, исходя из времени выполнения, зависимостей между тестами, и т.п.

В отличие от vManager, Jenkins CI и gitlab позволяют управлять более тонко процессом запуска тестов на всех этапах и, самое главное, не требуют радикально менять инфраструктуру.

Jenkins использует для описания процесса тестирования язык Apache Groovy, на основе которого доступно два вида синтаксиса: декларативный (Declarative Pipeline) и скриптовый (Scripted Pipeline). Последний позволяет наиболее тонко управлять процессом регрессионного тестирования, формируя динамически список тестов и определяя параметры, сервер и порядок запуска отдельных тестов.

**Взаимодействие между Jenkins, Gitlab и vManager**

Учитывая упомянутые выше особенности систем, наибольшую гибкость может дать сочетание трех программных решений для построения общего конвейера тестирования: Gitlab для хранения и управления исходным кодом; Jenkins для запуска процесса тестирования и балансировки нагрузки; Результаты тестирования при необходимости загружать в базу данных Cadence vManager для дальнейшего анализа.

Схематично это представлено на Рисунке 1.

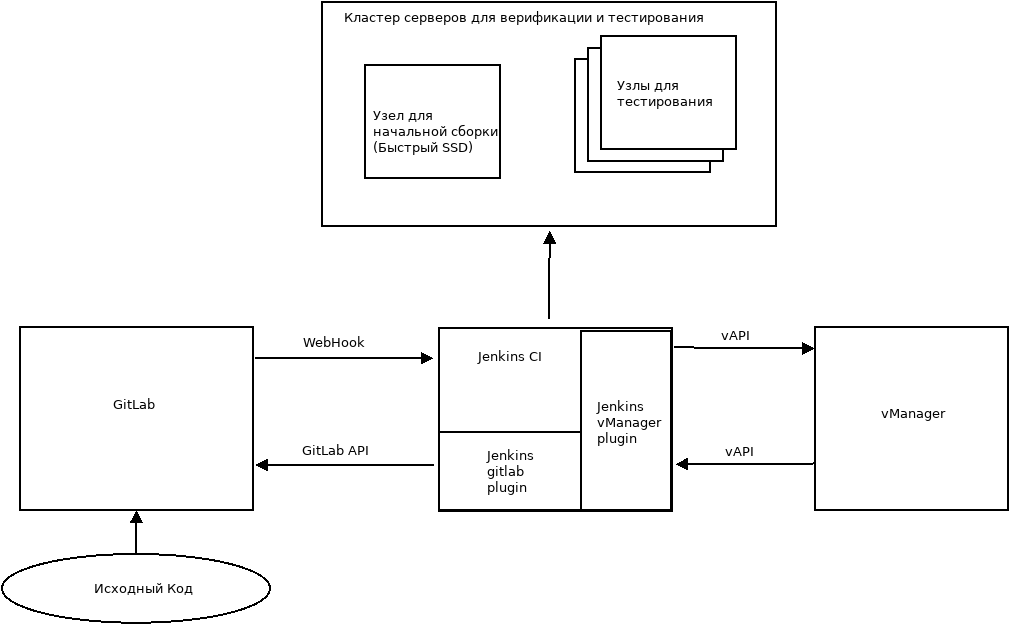


Рисунок Схема взаимодействия gitlab, jenkins и vmanager

Исходный код поступает в репозиторий в gitlab при помощи клиента git. По этому событию gitlab отправляет HTTP POST запрос к Jenkins (Через механизм WebHook) с информацией о новом коммите. Jenkins запускает процесс сборки и сообщает gitlab, что запущено тестирование внесенных изменений. Проект собирается на отдельном сервере, где есть быстрый SSD накопитель, так как на этапе сборки большую роль играет время доступа к большому количеству файлов небольшого размера. Запуск моделирования можно выполнять уже на всех серверах, получая доступ к результатам сборки используя сетевую файловую систему NFS. После окончания тестирования, используя “Jenkins vManager plugin”, отправляется запрос на импорт результатов регрессии в БД vManager. Сводная информация о тестах поступает из vManager в Jenkins, а Jenkins, в свою очередь, отправляет информацию о состоянии сборки в gitlab. Сессия vManager связывается с номером сборки в Jenkins и при удалении сборки в Jenkins удаляется сборка и в vManager. Все этапы настройки этого конвейера хорошо документированы, кроме связки vManager и Jenkins, о которой будет сказано отдельно.

**Импорт результатов в vManager**

При моделировании IP блоков информация о покрытии выгружается в виде файлов с расширением ucm и ucd. Помимо этого сохраняется журнал запуска, в который записывается информация об ошибках и предупреждениях.

vManager предоставляет два интерфейса способа внесения результатов тестирования в базу данных. Первый из них – это интерфейс “Сборки запусков”, или “Collect Builds”, который можно вызывать вручную из графического интерфейса vManager или через vAPI. В этом случае необходимо выполнить два вызова: один для сохранения журнала сборки и выполнения, а второй для добавления к созданной сессии информации о покрытии.

Второй вариант – это интерфейс запуска сессии. Этот вариант предпочтительнее, так как он позволяет за один запуск импортировать как журналы сборки, так и покрытие, но требует создание специального vsif-файла, имитирующего запуск сессии. Помимо этого, при запуске моделирования необходимо использовать опции командной строки xrun: ‘*-write\_metrics* *–vsof\_dir .’.*

Пример такого vsif-файла, который необходимо генерировать на уровне системы сборки проекта, приведен на Листинге 1.

#define TEST\_LOGS\_DIR '/optane/build/basic.logs'   
  
session rcm\_uvm\_dummy.basic {   
 top\_dir : TEST\_LOGS\_DIR;   
};   
  
test basic {   
 model\_dir: TEST\_LOGS\_DIR;   
 run\_script: "ln -sf $DIR(model)/scope $DIR(session)/scope; mkdir -p .SRvSof; cp $DIR(model)/cdns\_sim.vsof .SRvSof/; cat $DIR(model)/xrun.log; exit `cat $DIR(model)/result.txt`";   
 scan\_script: "vm\_scan.pl ius.flt uvm.flt";    
};   
  
#ifdef TEST\_LOGS\_DIR   
#undef TEST\_LOGS\_DIR   
#endif

Листинг Пример vsif-файла для импорта данных в vManager

Приведенный на Листинге 1 vsif-файл создает в директории, которую выделяет vManager для данной сессии, ссылки на файлы покрытия, сгенерированные в ходе запуска Jenkins. Так же копируется vsof-файл, с информацией о запуске. Результат прохождения теста берется из файла result.txt, который создает верификационное окружение.

Так как такая “сессия” не выполняет реального моделирования и завершается достаточно быстро, то ее можно запускать без параллелизации нагрузки, тем самым избегая необходимости миграции инфраструктуры на одну из поддерживаемых систем распределения нагрузки.

**Заключение**

Из-за специфики САПР разработки микроэлектроники, применять современные практики, позаимствованные из индустрии разработки программного обеспечения, достаточно сложная задача. Описанный в данной статье подход формировался в ходе разработки СБИС 1888ВС048 и 1888ВС058. После добавления поддержки vManager он задействован для разработки ряда отдельных IP-блоков для будущих микросхем. Использование этого подхода позволяет не только автоматизировать регрессионное тестирование СБИС, но производить автоматизированный сбор и анализ метрик в автоматическом режиме.   
Автор считает, что в данной работе новыми являются следующие положения и результаты:

* Систематизирован, описан и проверен на практике опыт интеграции лучших аспектов трех программных решений;
* Получен гибкий и автоматизированный конвейер автоматизированного тестирования, учитывающий особенности инструментов и позволяющий работать с оптимальной производительностью;

Литература

[1] “Использование семейства инструментов CMake для моделирования проектов сложных СБИС в среде Cadence Inсisive”,

Андрианов А.В., Труды НИИСИ РАН, т. 7, № 4, 2017

[2] “Методика гибридной верификации СБИС “Система-на-Кристалле”, Андрианов А.В., Шагурин И.И. "Датчики и системы" , 2018г., №2, с. 14-18.

[3] “Верификция IP-блоков а RTL-модели и ПЛИС-прототипе при помощи высокоуровневого языка lua”,Андрианов А.В.,Тезисы докладов международной научно-технической конференции “Электроника – 2015”, с. 102-103

[4] “Методы обеспечения переносмости тестовых сценариев между различными верификационными окружениями”,Андрианов А.В. Сборник трудов конференции МЭС-2018, т.2 с. 79-84

[5] Щигорев Л.А. Применение шины диагностики в задаче саморемонта блоков статической оперативной памяти // Нано- и микросистемная техника. 2018, Т. 20, № 2. С. 98-106.

[6] “Измерение покрытия кода при работе без операционной системы встроенными средствами компилятора gcc”, Андрианов А.В. Сборник докладов конференции DSPA-2018, т.2. с 657-661

[7] “Реализация возможности пошаговой отладки при отладке тестовых сценариев на модели СБИС СнК”, А.В. Андрианов, Труды НИИСИ РАН, т. 8, № 3, 2018, с. 56-60

[8] “Мультимедийная СнК с процессорными ядрами PowerPC и NMC3”, Филимонова И.П., Бескоровайный И.В., Дрягалкин Д.И., Чумаченко Г.О.,Залетов В.Ю., Андрианов А.В., Сборник трудов 3й международной конференции “Микроэлектроника – 2017”

[9] “Практические способы оптимизации процесса регрессионного тестирования СБИС СнК”, Андрианов А.В., Сборник трудов 4й международной конференции “Микроэлектроника – 2018”

[10] Cadence Incisive Enterprise Simulator Refererence Manual, Cadence 2015

[11] Reducing Snapshot Creation Turnaround for UVM/SV Based TB Using MSIE Approach, Cadence User Conference 2015, <https://www.cadence.com/downloads/cdnlive/in/2015/VER2.pdf>