

Vivado в системе контроля версий (на примере GIT)

Владимир Викулин, инженер по применению Xilinx

20.10.2020 г.

План вебинара

- ◆ Что такое система контроля версий (СКВ)
- ◆ Универсальная СКВ GIT. GIT Workflow
- ◆ Почему возникают проблемы сопряжения Vivado с СКВ
- ◆ Как решать эти проблемы: организация проектирования в Vivado с использованием СКВ
- ◆ Пример совместной работы Vivado с GIT

Что такое Система Контроля Версий (СКВ)

[Википедия:](#)

Система управления версиями (от [англ.](#) *Version Control System, VCS* или *Revision Control System*) — [программное обеспечение](#) для облегчения работы с изменяющейся информацией. Система управления версиями позволяет хранить несколько версий одного и того же документа, при необходимости возвращаться к более ранним версиям, определять, кто и когда сделал то или иное изменение, и многое другое.

Git Workflow



Git – универсальная распределенная система контроля версий

1. Универсальность – не зависит от обслуживаемого проекта
2. Распределенность – проекты хранятся в одном или нескольких репозиториях. Все репозитории проекта равноправны (но есть “самый равноправный” - origin). Контроль за соответствием репозитория друг другу лежит на разработчиках
3. Храниться должны только исходники
4. Предпочтение – текстовым файлам
5. Хранятся изменения
6. Коллективная работа достигается за счет разделения на независимые ветви. Обязательно имеется главная ветка (MASTER)
7. Каждая новая функция реализуется в своей отдельной ветви
8. Изменения добавляются в MASTER путем слияния ветвей. При этом возможны коллизии
9. Коллизии находятся автоматически, но разрешаются разработчиками вручную

Подробнее: <https://git-scm.com/book/ru/v2>

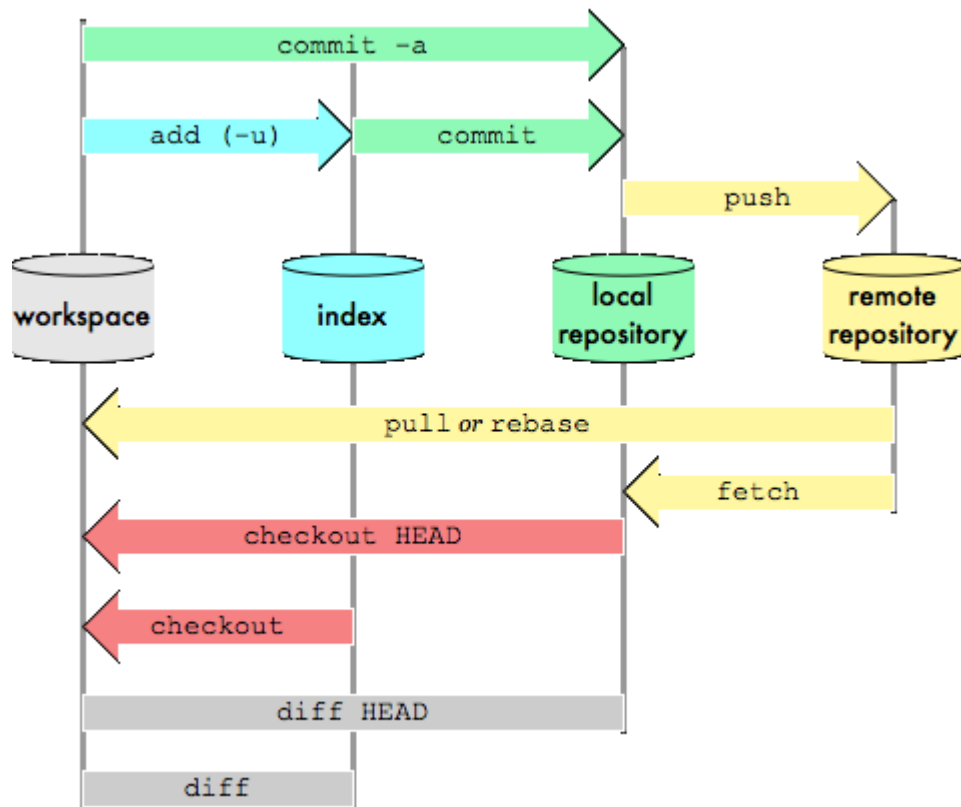
GIT Workflow (2)

Функции СКВ GIT

- Хранение сорцов
- Перенос проекта на другое рабочее место
- Организация коллективной работы
- Отслеживание изменений
- Полная история процесса разработки
- ...

Использование СКВ – обязательное условие современного процесса R&D

Рабочий процесс в GIT



Почему возникают проблемы сопряжения Vivado с СКВ

Vivado не Git-friendly. Почему?

1. Vivado производит огромное число вспомогательных файлов и каталогов, большинство из которых не требуют сохранения в git
2. Git ориентирован на текстовые файлы, а Vivado ориентирован визуальное проектирование и содержит много нетекстовых файлов
3. Визуальное проектирование плохо совместимо с контролем версий
4. При визуальном проектировании трудно организовать слияние ветвей и разрешение коллизий

Организация проектирования в Vivado с использованием СКВ

Проблемы сопряжения можно изящно решить:

1. На самом деле Vivado – script-ориентированная система, все действия в которой могут быть выполнены с помощью TCL
2. Поэтому необходимо переходить на использование TCL-скриптов вместо команд GUI
3. Визуальные диаграммы допустимы, но текстовые файлы предпочтительнее
4. Продуманная структура каталогов проекта помогает в работе

Организация проектирования в Vivado с использованием СКВ (2)

Самое простое решение

1. Продумать подходящую структуру каталогов
 2. Разделить проект на две части “песочницу” и “каталог сорцов”
 3. “Каталог сорцов” включить в СКВ
 4. “Песочницу” исключить из СКВ и генерировать tcl-скриптом
 5. Этот tcl-скрипт включить в СКВ
 6. Восстановить полный проект с помощью tcl-скриптов
- В идеале можно перейти в non-project mode и управлять Vivado только tcl-скриптами (отлично работает для больших проектов)
 - Имеется много вариантов решения. В зависимости от ваших требований и возможностей выбирайте наилучший для вас вариант.

Организация проектирования в Vivado с использованием СКВ (3)

Что необходимо сохранять в СКВ?

- Исходные тексты (как проекта, так и IP)
- Скрипты
- Конфигурационные файлы и данные кастомизации
- Блочный дизайн

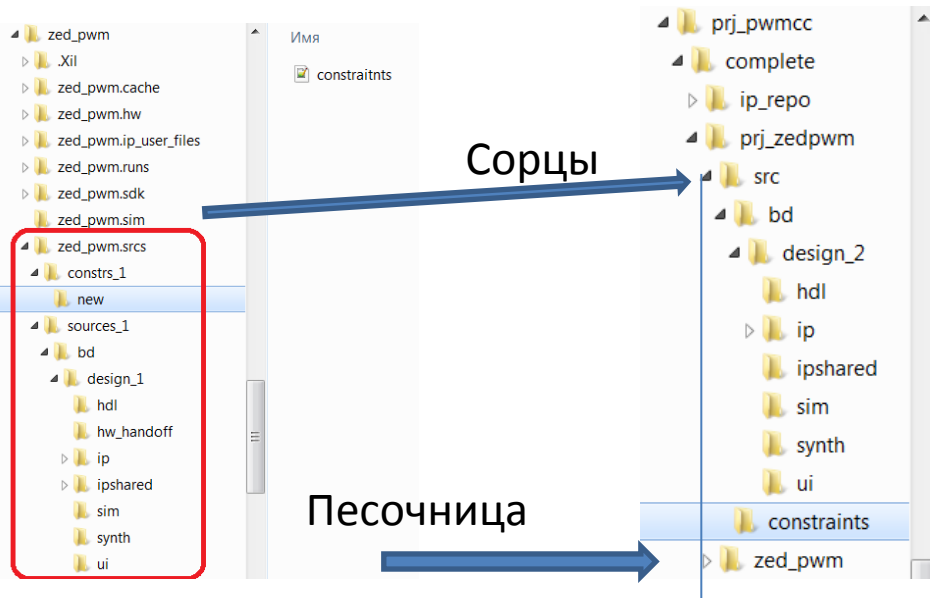
Что не нужно сохранять?

- “Стандартные IP”
- “Служебные файлы”
- Временные файлы

При работе в GIT используйте файл ***gitignore*** для облегчения работы

Организация проектирования в Vivado с использованием СКВ (4)

Модификация исходного проекта – разделение на “сорцы” и “песочницу”



Алгоритм:

- Создаем каталог для сорцов и переносим их туда
- Убираем сорцы из песочницы
- Для проверки генерируем битстрим
- Выполняем Reset_project

Для переноса файлов в сорцы:

- SaveAs
- File/SaveBlockDesignAs

Организация проектирования в Vivado с использованием СКВ (5)

Главный инструмент -
скрипт `write_project_tcl`

Он встроен в меню Vivado и напрямую недоступен, но, при необходимости его можно скачать из репозитория Xilinx на Гитхабе и модифицировать

Модификация скрипта

Зачем?

Для “тонкой настройки”.
Например, прописать пути к нестандартным каталогам
либо выполнить требуемые команды

Где взять исходник?

https://github.com/Xilinx/XilinxTclStore/blob/master/tclapp/xilinx/projutils/write_project_tcl.tcl

Организация проектирования в Vivado с использованием СКВ (6)

Полезные ссылки

- [Харр1165](#)
- [Ug1198](#)
- Презентация [Revision Control Methodology](#)
- Блог [Адама Тейлора](#)
- Вики [Лаборатории Навигационных Систем и кафедры РТС НИУ МЭИ](#)

Пример совместной работы Vivado с GIT

Переработаем проект ZED_PWM (из предыдущего вебинара) для работы под контролем GIT. Проект состоит из основного проекта и кастомного IP-ядра, размещенного в пользовательском репозитории IP-ядер

Требуется:

1. Правильно и удобно организовать структуру каталогов
2. Разделить основной проект на “сорцы” и “песочницу”
3. Сгенерировать скрипт восстановления “песочницы”
4. Сохранить сорцы в GIT (commit)
5. Сохранить сорцы в удаленном репозитории GIT (push)
6. Восстановить проект из удаленного репозитория (clone)
7. Построить восстановленный проект

Пример совместной работы Vivado с GIT (2)

Пример организация структуры каталогов

proj_folder

.git

.gitignore

user_ip_repo

xilinx.com_user_myPwM_1.0

HDL

...



Помещаем под version control

prj_workspace

← **текущий каталог (.)** Относительно него задаются пути всех остальных каталогов

restore_project.tcl

← Скрипт генерации проекта (включен в version control)

src

← исходники под version control

...

vivado_prj

← Сгенерированный проект (исключен из version control)

Пример совместной работы Vivado с GIT (3)

Организация основного проекта

<code>prj_workspace</code>	← текущий каталог (.) Относительно него задаются пути всех остальных каталогов
<code>.git</code>	← Если основной проект и IP-репозиторий находятся под раздельным контролем, локальный репозиторий находится здесь
<code>restore_project.tcl</code>	← TCL-скрипт генерации проекта (включен в version control)
<code>restore_project.cmd</code>	← командный файл проекта (включен в version control)
<code>src</code>	← исходники под version control
<code>bd</code>	← файлы блок-диаграмм
<code>constraints</code>	← констрейнты
<code>hdl</code>	← исходники на языках verilog и VHDL
<code>vivado_project</code>	← Сгеперированный проект (исключен из version control)

Пример совместной работы Vivado с GIT (4)

Организация пользовательского IP-каталога

user_ip_repo ← Каталог для репозитория пользовательских ядер.
 Должен иметь фиксированное положение
 относительно пользовательского проекта

xilinx.com_user_myPwM_1.0 ← каталог отдельного IP-ядра. Под контроль берем
 все, кроме запакowanego IP-ядра

hdl ← каталог для текстовых исходников

...

Пример совместной работы Vivado с GIT (5)

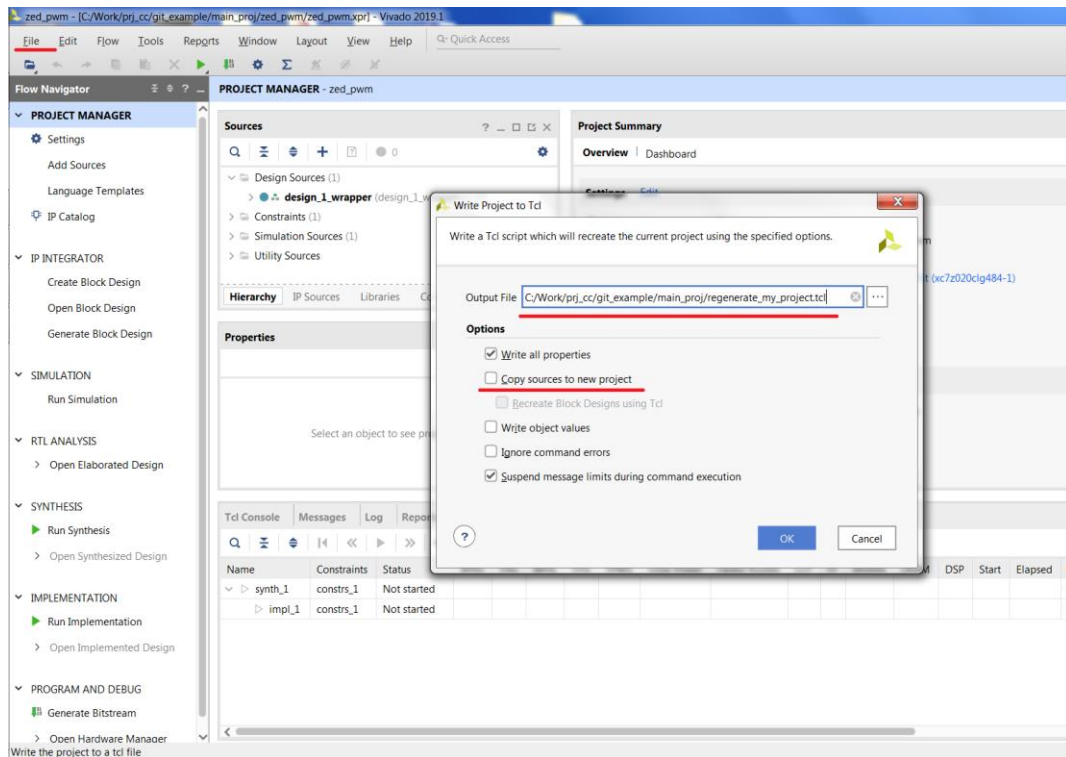
Установка и инициализация GIT под Windows

1. Скачиваем дистрибутив отсюда: <https://git-scm.com/download/win> и устанавливаем его
2. Далее будем работать в оболочке git bash
3. Конфигурируем:
 - `git config --global user.name "John Doe"`
 - `git config --global user.email johndoe@example.com`
 - Задаем режим сохранения концов строк
 - Задаем редактор
 - Задаем origin
4. Создаем основной репозиторий `git init --bare --shared` (как правило, на сервере, но можно и на локальном компьютере)

Пример совместной работы Vivado с GIT (6)

Генерация скрипта восстановления проекта

1. Вызовите File/Write TCL...
2. **Отключите** опцию “Copy sources to new project”
3. Выберите каталог для сохранения скрипта на один уровень выше каталога проекта



Пример совместной работы Vivado с GIT (7)

Первый commit

- Переходим в самый верхний каталог проекта и даем команду: `$git init`
- Настраиваем в этом каталоге файл `.gitignore` (см. пример)
- В верхнем каталоге (там должны быть файлы `.git` и `.gitinit`) даем команды:
`git add .gitignore`
`git add ip_repo/*`
`git add main_proj/*`
`git commit -m "first commit"`
- Загружаем коммит на сервер:
`git push origin master`

(или в репозиторий на локальном компьютере:

`git push --set-upstream /c/Work/prj_cc/git_repo/pwmprij master)`

Полезные команды для контроля:

`git status`

`gitk --all`

Пример совместной работы Vivado с GIT (8)

Развертывание проекта

- Переходим в каталог, где будем развертывать проект, например C:/Work/prj_cc/git_cloned
- Даем команду `git clone <адрес репозитория> .`
(Наш репозиторий находится в каталоге /c/Work/prj_cc/git_repo/pmwprj)

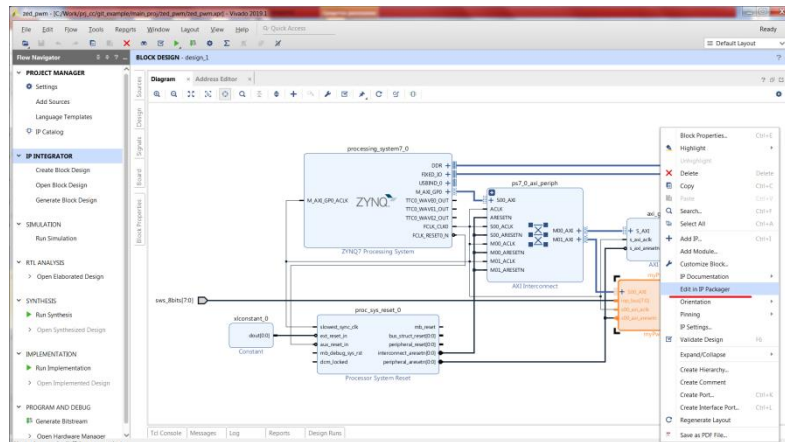
Пример совместной работы Vivado с GIT (9)

Построение проекта

- Переходим в каталог основного проекта (main_proj)
- Запускаем командный файл восстановления проекта: regenerate_my_proj.cmd
- Возникает новый каталог проекта zed_pwm
- Переходим в него, вызываем Vivado и генерируем битстрим

Пример совместной работы Vivado с GIT (10)

Модификация IP-ядер



Edit in IP Packager

Choose a project name and location for editing.

Project name:

Project location: ...

Edit IP project will be created at: ...p/myPwM_v1_0...



Пример совместной работы Vivado с GIT (11)

Модификация IP-ядер (2)

- Модифицируем сорцы и переупаковываем ядро
- Апгрэйдим ip-ядро в основном проекте
- Коммитим все измененные сорцы

При этом сам проект IP-ядра сохранять не нужно!

Пример совместной работы Vivado с GIT (12)

Сравнение результатов:

Zip – архив (через File/Project/Archive):

– 5.3MB

Репозиторий git с IP (после первого коммита):

– Около 300 КБ

Замечание:

Для работы требуются одинаковые версии Vivado!

Иначе нужно будет сохранять кэш IP-ядер проекта

Приложение: Основные команды GIT

Команда	Описание
git init	Инициализация локального репозитория
git init <prj_nm> --bare [-shared]	Инициализация удаленного репозитория
git status	Проверка статуса файлов
git add	Добавление файлов в stage
git commit -m "..."	Коммит в локальный репозиторий
git reset --hard	Восстановление из локального репозитория
git remote add <name> <addr_or_url>	Указание расположения удаленного репозитория
git push [--set-upstream] origin master	Заливка в удаленный репозиторий
git clone <addr_or_url> .	Первое скачивание из удаленного репозитория
git pull	Последующие скачивания из удаленного репозитория
git checkout ...	Скачивание из локального репозитория, переход на новую ветку и т.д. ...

Спасибо за внимание !

Ваши вопросы



Наши ответы



**МАКРО
ГРУПП**

Официальный дилер Xilinx

Контакты

Тел.: 8 (800) 333-06-05

email: SALES@MACROGROUP.RU

Продукция Xilinx и техподдержка: fpga@macrogroup.ru

Олег Болихов – руководитель направления “Цифровая электроника”

Дмитрий Хорьков – руководитель направления Xilinx

Владимир Викулин, Дмитрий Шадрин – техподдержка Xilinx

Xilinx – полезные ссылки

- ◆ Сайт Xilinx: <https://www.xilinx.com/>
- ◆ Сайт Developer: <https://developer.xilinx.com/>
- ◆ Форум: <https://forums.xilinx.com/>
- ◆ Обучение: <https://xilinxprod-catalog.netexam.com/>
- ◆ Репозиторий: <https://github.com/Xilinx>
- ◆ Отладочные платы и платформы:
<https://www.xilinx.com/products/boards-and-kits/see-all-evaluation-boards.html>

Xilinx – материалы с сайта

Справочные и методологические материалы на сайте Xilinx

- ◆ DocNav – все в одном месте на вашем компьютере
- ◆ Selection guides – руководства по выбору
- ◆ User guides – руководства по применению
- ◆ UFSM – методология проектирования
- ◆ AR – ответы на вопросы

Xilinx – как получить техподдержку

Техподдержка

- ◆ Сначала посмотреть на форуме Xilinx
- ◆ Поискать на ресурсах Xilinx:
<https://www.xilinx.com/support.html>
- ◆ Обратиться в Macro: fpga@macrogroup.ru
- ◆ Открыть service request:
https://service.xilinx.com/sservice_prod/start.swe

Xilinx – на чем разрабатывать и отлаживать свои проекты

Отладочные платы и платформы

Подберем отладку под ваши задачи

- ◆ <https://www.macrogroup.ru/catalog/partgroup/378>
- ◆ <https://www.xilinx.com/products/boards-and-kits.html>

