Using Revision Control In Vivado

Tim Vanevenhoven

XILINX
ALL PROGRAMMABLE™

- Overview of revision control
- Recent improvements in Vivado Design Suite
- Recommendations for Vivado Design Suite
- Summary / Q&A

Why use revision control?

- Automates critical management of source files
  - Milestones are “backed up”
  - Can revert to previous milestone if necessary
  - Changes are logged
- Reduces compile times
  - Dependency tracking – only build when inputs change
  - Take advantage of parallelization – independent steps can run in parallel
- Most customers use some form of revision control

The UltraFast Design Methodology Guide (UG949) recommends using revision control with Vivado Design Suite

Which revision control systems do you use?

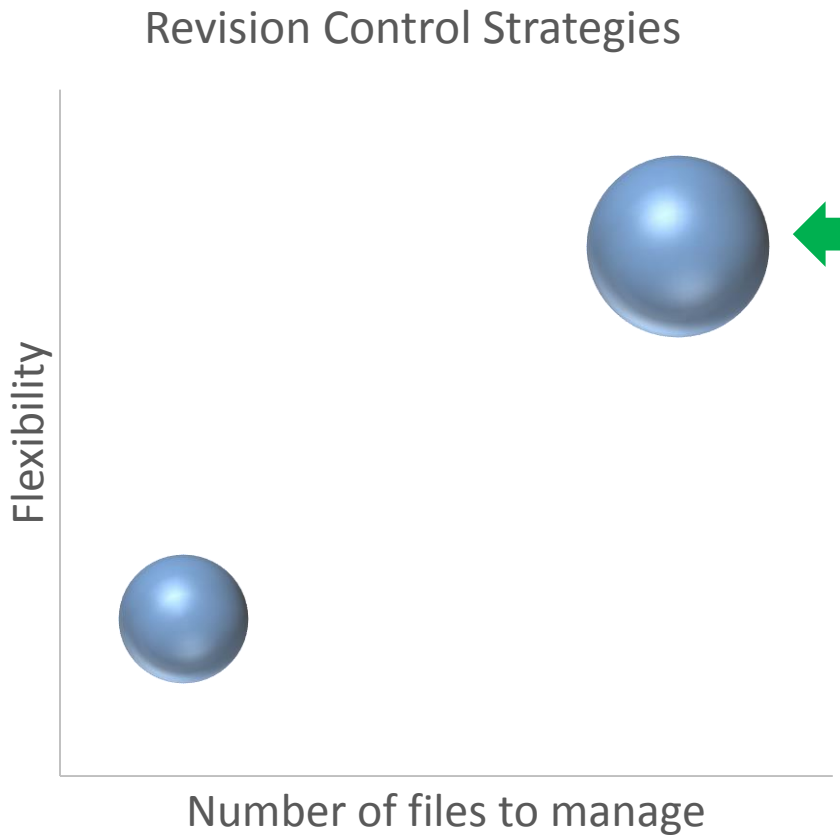
Vivado revision control philosophy

- Designed to be “friendly” to revision control
  - No direct integration with specific tools
  - Generally prefer ASCII-based internal files (XML project files)
  - Tolerate hidden “dot” files and read-only sources “locked” in revision control
  - Minimize file updates on opening projects and only write changes
- Work with different use models
  - GUI (IDE) vs non-GUI (TCL scripting)
  - Project vs non-project flows
  - Distributed vs centralized revision control repositories

Recent improvements in Vivado Design Suite

- Released with Vivado Design Suite 2015.1
  - Only the .bd file is required to recreate a block design
  - Compared to 2014.1 the average number of files per IP is about 1/3
  - Increased / focused testing around recommended methodology
- Enhancements to Chapter 2 in the UltraFast Design Methodology Guide released in June 2015
- Updated Revision Control Quick Take Video
- Coming in Vivado Design Suite 2015.3 – September 2015
  - Introduction of core container – one file per IP
  - Further reduction in the number of files per IP
  - New revision control tutorial available via DocNav

Two documented revision control strategies



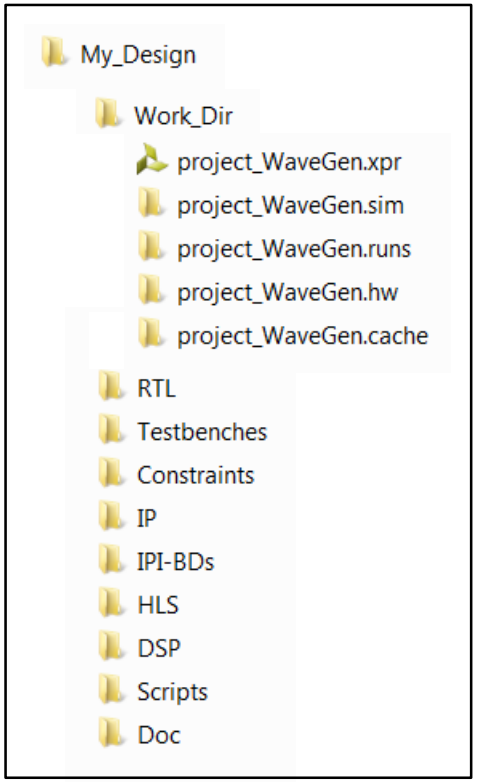
Xilinx recommendation

- Maximum Flexibility
  - Option to use IP in future releases of Vivado without upgrading
  - Shorter runtime to rebuild project

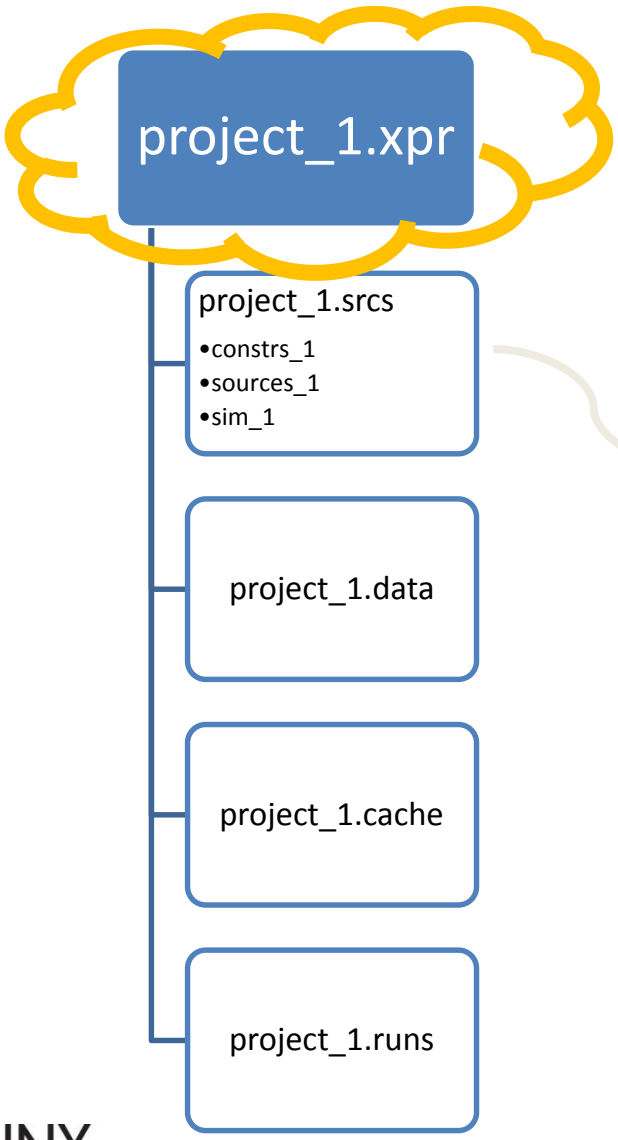
- Minimum Files
  - The least number of files to manage at the expense of flexibility and run time

Overview of recommended directory structure

- Manage different source types in separate remote directories
- Use a “work” directory to compile the design and to create Vivado projects
- Set up the Vivado project with remote sources from the various directories
  - Elect to not copy them into the project
- Either manage revisions from the remote sources area directly or use it as a local sandbox for the project



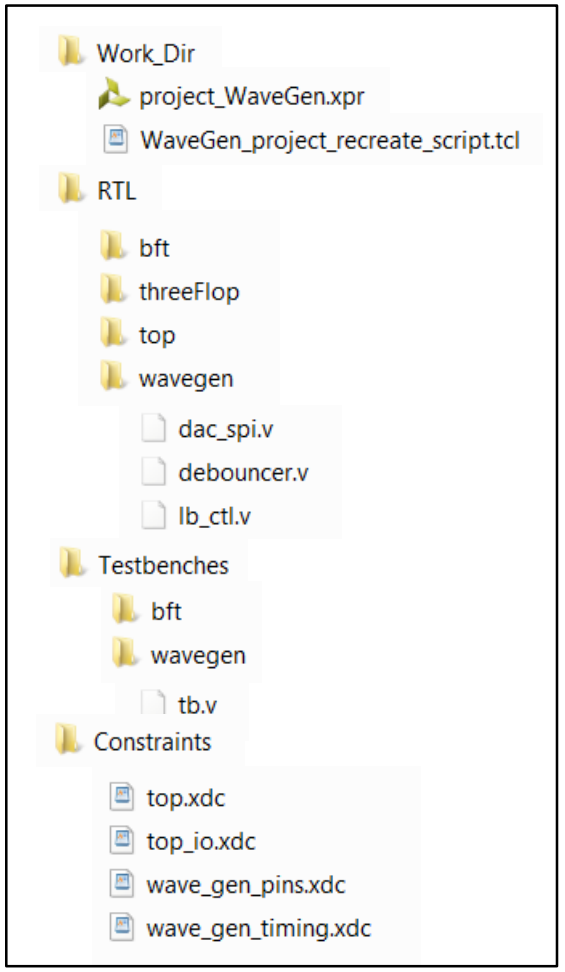
Project with sources under revision control



Remote Sources
HDL (Verilog/VHDL)
IPI BD (entire dir)
IP XCI (entire dir)
Sys Gen IP XCI (entire dir)
XDC Constraints
Tcl Scripts
Simulation Testbenches
COE/BMM/ELF

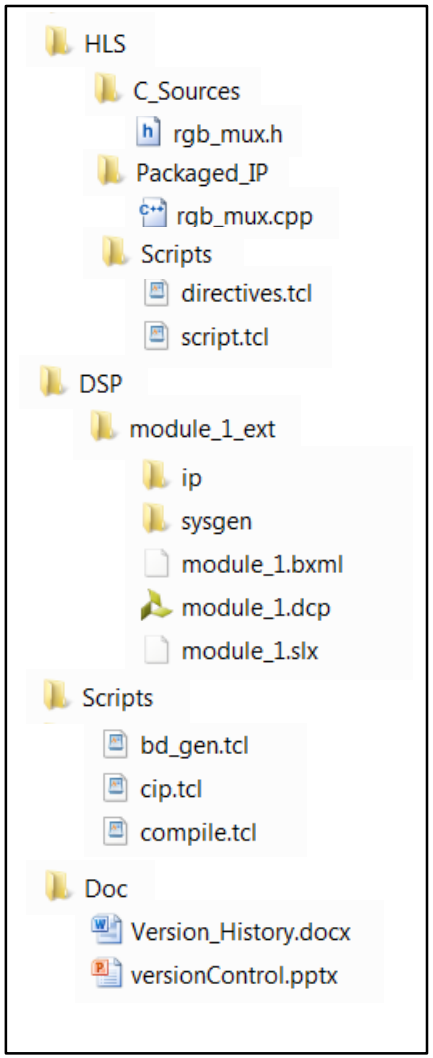
Maximum flexibility - Recommended files to manage

- If using projects, manage just the .xpr file or the Tcl recreate script
  - Use the write_project_tcl command to create a script to re-create the project
  - Do not check in the project sub-directories
- Use your own judgment on directory structures for RTL, XDC, etc.
- For IP and IP Integrator sources, manage the entire directory tree
  - Generated sources can be used in future Vivado releases



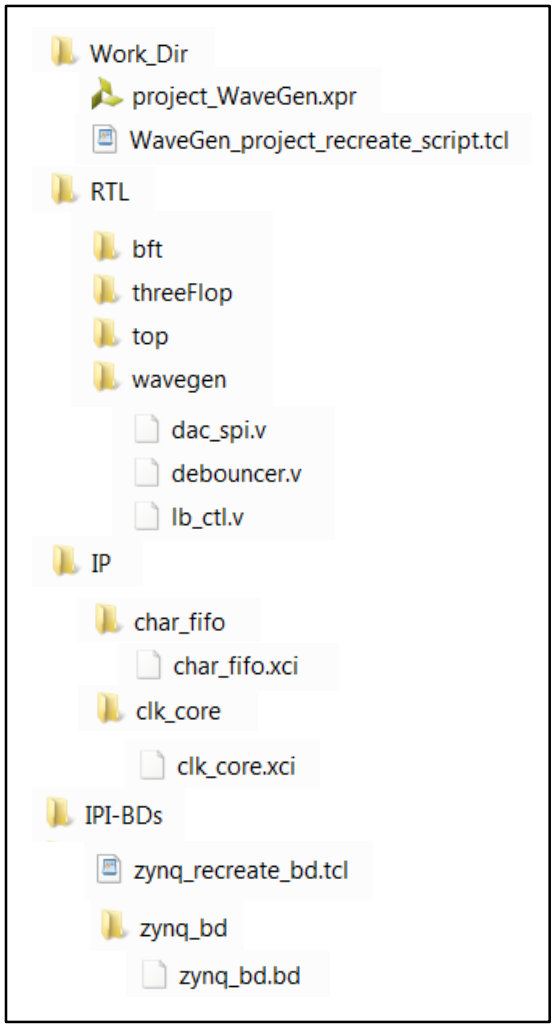
Maximum flexibility - Recommended files to manage

- Manage Vivado HLS source files, scripts, example projects, and packaged IP
- Manage the entire System Generator directory for DSP sources
- Manage Scripts and Docs as desired
- For SDK, manage the .hdf file



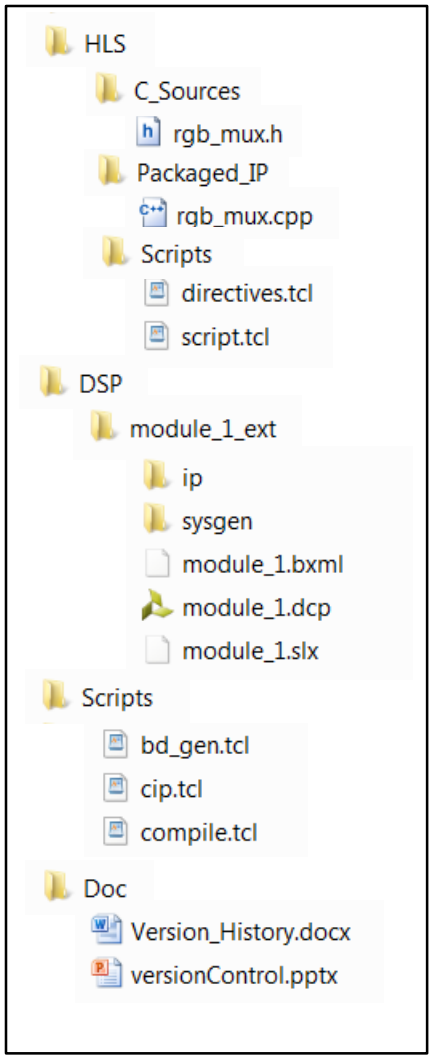
Minimum files - Recommended files to manage

- If using projects, manage just the .xpr file or the Tcl recreate script
  - Use the write_project_tcl command to create a script to re-create the project
  - Do not check in the project sub-directories
- For IP, check in the .xci file only
  - The .xci file can be used recreate the IP output products
  - IP can only be recreated using the version of Vivado the .xci was created with
- For IP Integrator, check in the .bd file or the Tcl recreate script
  - Use the write_bd_tcl command to create a script for the block design



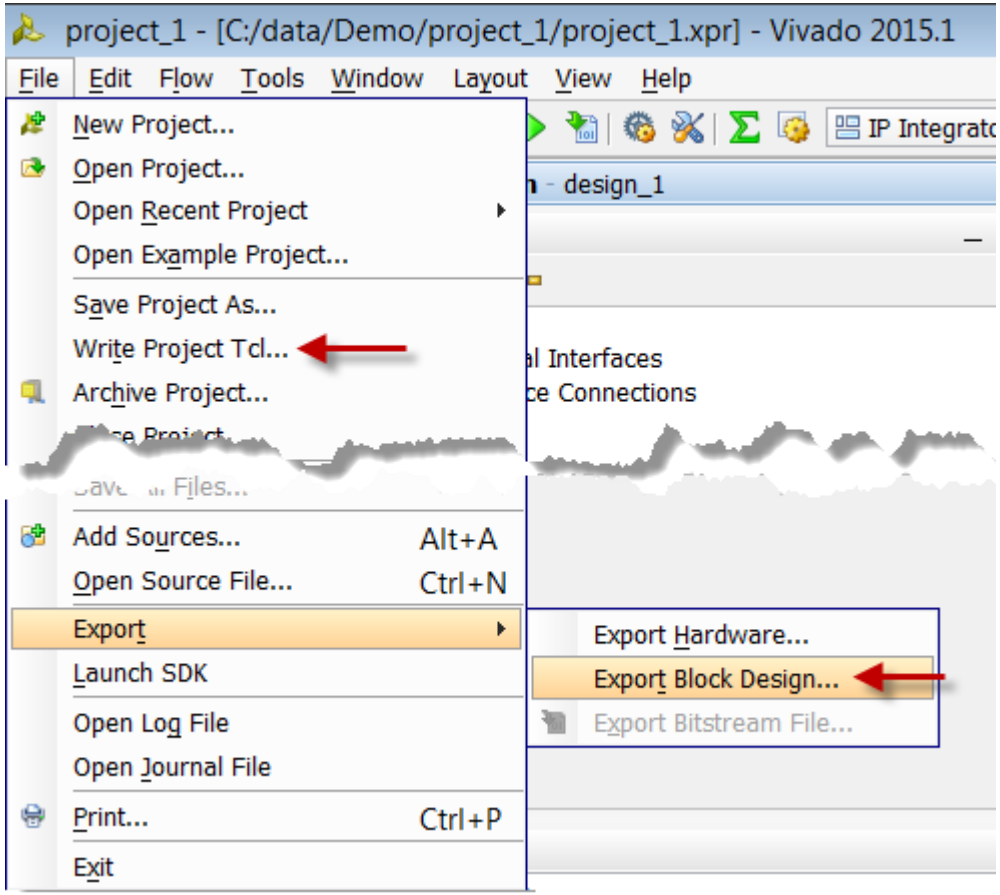
Minimum files – Recommended files to manage

- Manage Vivado HLS source files, scripts, example projects, and packaged IP
- Manage the entire System Generator directory for DSP sources
- Manage Scripts and Docs as desired
- For SDK, manage the .hdf file



Best practices for revision control

- Create a Tcl script to recreate a project
  - Write_project_tcl creates a template that you can use/modify
- Generate a Tcl script to recreate an IP Integrator block design
  - Write_bd_tcl creates a Tcl script to recreate just the BD
  - Highly version dependent – like the IP
- Script check-in / check-out flow



Summary

- There are two documented approaches to decide which files to manage
  - Maximum flexibility / least rebuild time **Xilinx recommendation**
  - Minimum number managed files
- A key is to use remote sources and not to manage the project directories
- Updated documentation available now
  - Chapter 2 in UG949
  - Revision Control Quick Take Video
  - Revision Control Tutorial