XDF XILINX DEVELOPER FORUM

Revision Control Methodology

Presented By

Brian Lay
Product Marketing Manager
12/10/2018

XILINX

Agenda

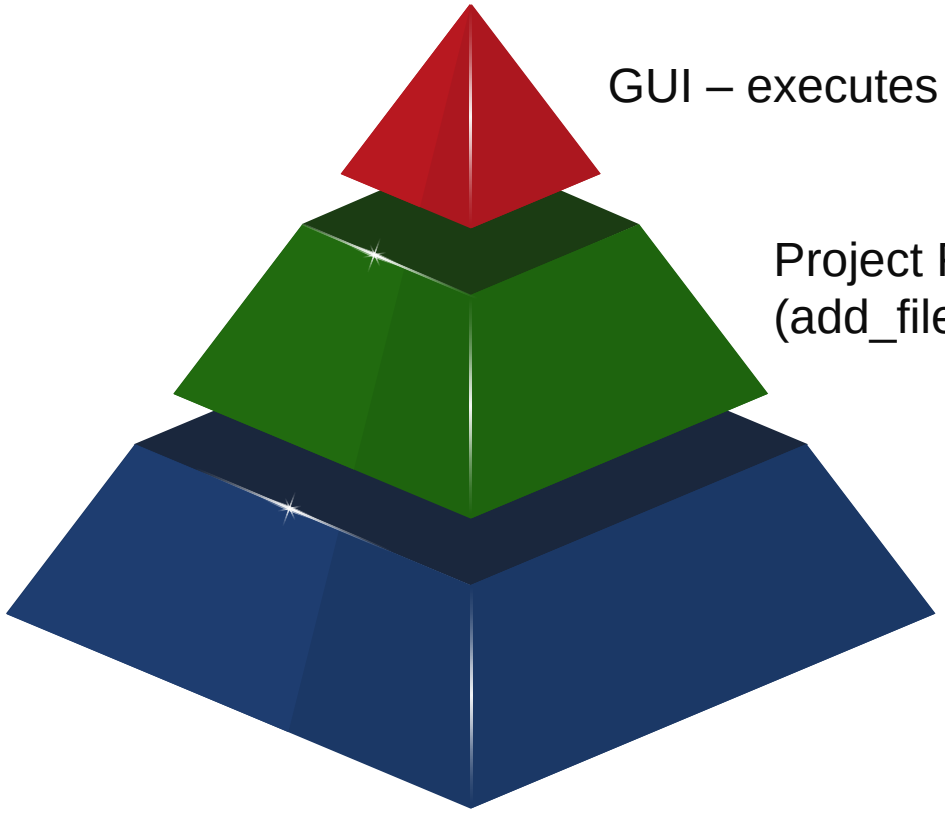
- **Motivation**
- **Recommendations for implementing a revision control strategy**
  - RTL projects
  - IP projects
  - BD projects
- **Future improvements**

Importance of Maintaining a Revision Control Strategy

- Reproduce previous results
- Board revisions
- Co-development
- Compliance
- Revenue

Implementing a Revision Control Strategy

Foundations of Vivado



GUI – executes project Tcl

Project Flow
(add_files, import_files, launch_runs)

Non-Project Flow
(read_verilog, read_vhdl, place_design)

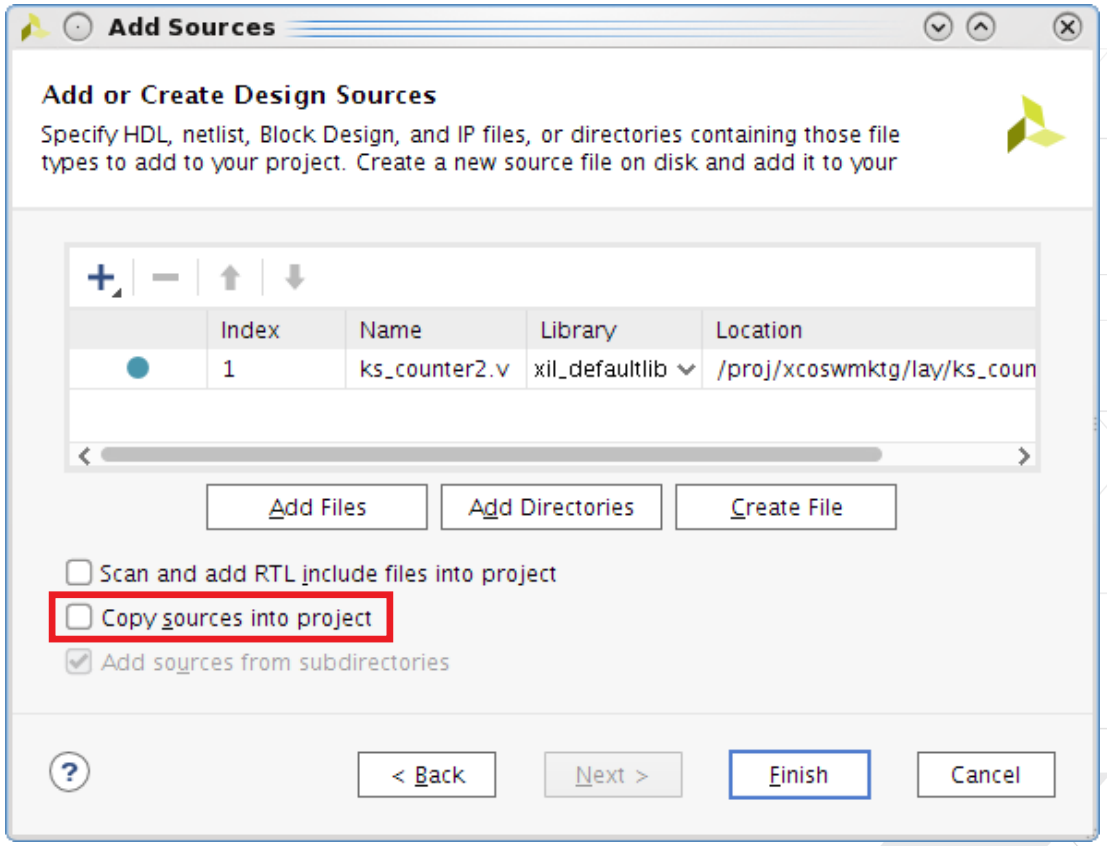
Focusing on project based scripted flow

Strategy for Successful Revision Control

- Use scripted flows for revision control
- Keep sources external to the project
- Revision control the source repository
- Generate a script to recreate the project
- Revision control the script
- Test your methodology

Adding Sources to Projects

- **Use add_files for all sources**
  - Keeps sources external to the project
  - Un-check “Copy sources into project”
- **Filesets contain all project sources**
  - get_filesets
  - Four “default” filesets
  - get_files –of [get_filesets]



RTL Example

./userdir
    ./my_repo/2018.2
    ./workspace

./my_repo/2018.2

bft.vhdl
FifoBuffer.v
async_fifo.v
bft_package.vhdl
core_transform.vhdl
round_1.vhdl
round_2.vhdl
round_3.vhdl
round_4.vhdl
bft_full.xdc
build.tcl

build.tcl

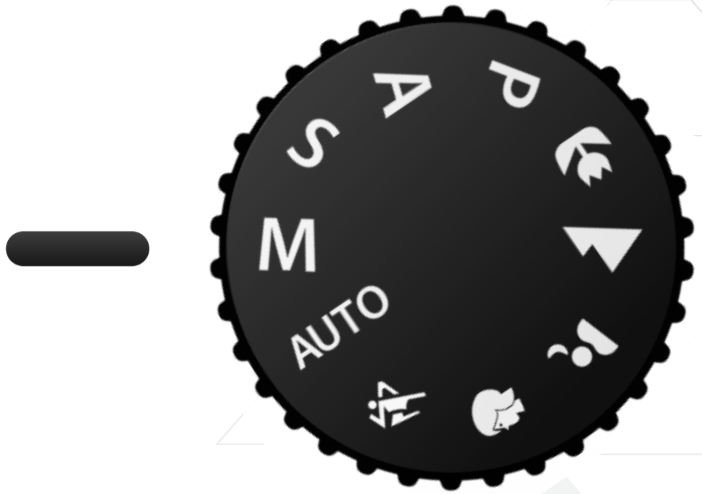
```
set dirname project_foo

create_project $dirname . -part xc7k70tfbg484-2
set source_repo ../my_repo/2018.2
add_files $source_repo/bft.vhdl
add_files $source_repo/FifoBuffer.v
add_files $source_repo/async_fifo.v
add_files $source_repo/bft_package.vhdl
add_files $source_repo/core_transform.vhdl
add_files $source_repo/round_1.vhdl
add_files $source_repo/round_2.vhdl
add_files $source_repo/round_3.vhdl
add_files $source_repo/round_4.vhdl
add_files -fileset constrs_1 $source_repo/bft_full.xdc

set_property library bftLib [get_files  $source_repo/round_1.vhdl]
set_property library bftLib [get_files  $source_repo/round_2.vhdl]
set_property library bftLib [get_files  $source_repo/round_3.vhdl]
set_property library bftLib [get_files  $source_repo/round_4.vhdl]
set_property library bftLib [get_files  $source_repo/core_transform.vhdl]
set_property library bftLib [get_files  $source_repo/bft_package.vhdl]

set_property top bft [current_fileset]
update_compile_order -fileset sources_1

launch_runs impl_1 -jobs 8
```

./workspace

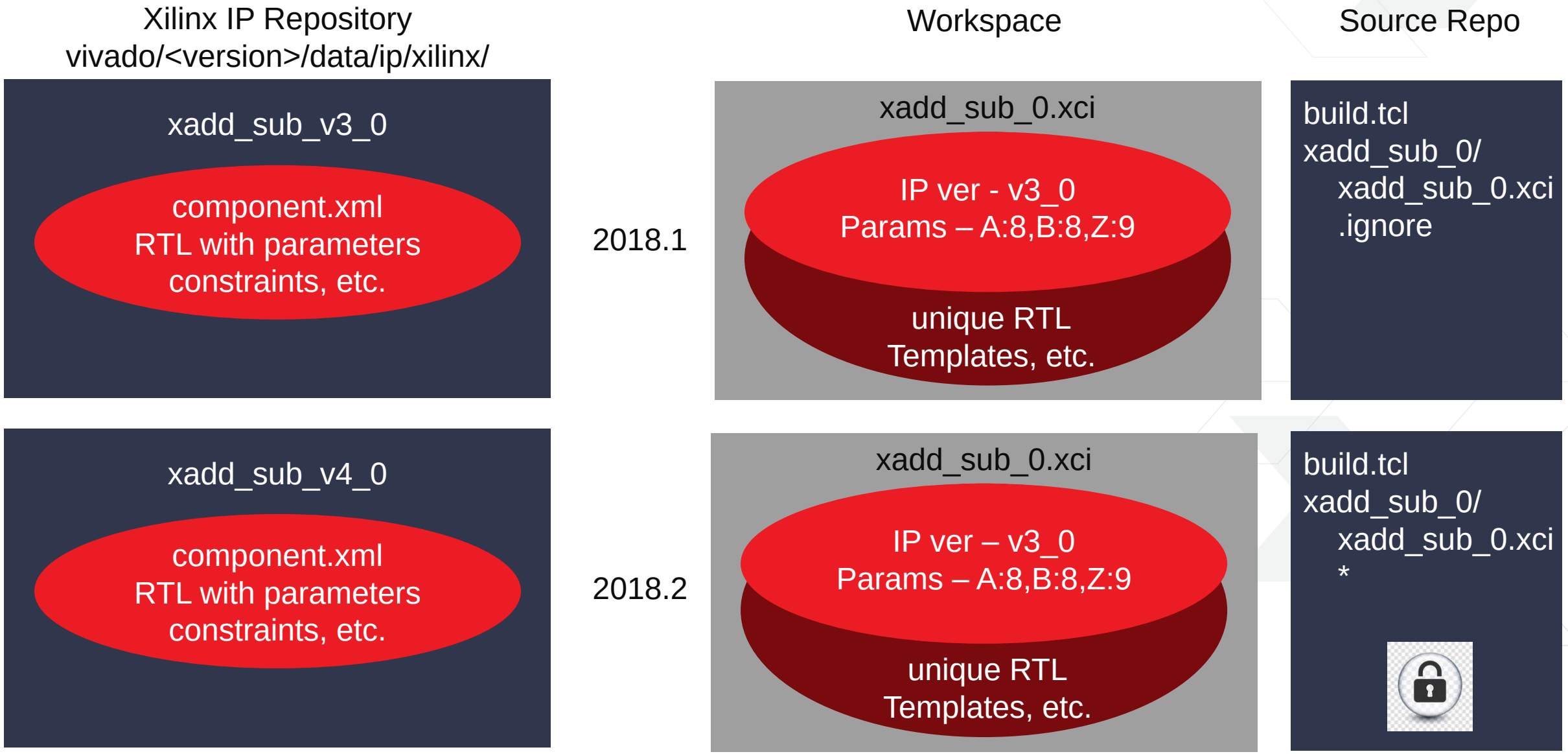
vivado –source ../my_repo/build.tcl

Creating a Project Script

- **Manually create the script**
  - Minimalist
  - Organized
  - Risk missing some settings
- **Automatically create the script using write_project_tcl**
  - Verbose
  - Complicated
  - Robust, should not miss any settings
- **Scripts must be maintained as projects evolve**



Understanding Xilinx IP



Xilinx IP Revision Control Options

IP Files to Revision Control	Size	Compile time	Re-customizable[1]	Forced to upgrade[2]
XCI	S	Slow[3]	Y	Y
TCL (write_ip_tcl)	S	Slow[3]	Y	Y
Whole IP directory	L	Fast	Y	N / locked
XCIX	M	Fast	Y	N / locked
DCP	S	Fast	N	NA

- **Two real options**
  - XCI or XCIX
- **Recommendation**
  - Start with only the XCI file
  - On upgrade switch to XCIX for IP with changes that are too disruptive

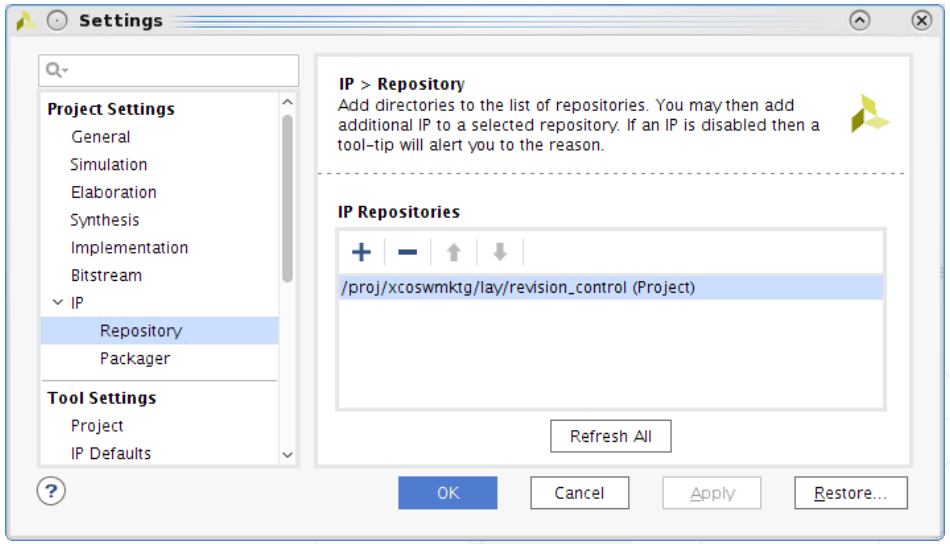
[1] In the existing version of Vivado that generated the original XCI
[2] Rebuild project using the existing version of Vivado and open project with latest version
[3] With Out-of-context synthesis and IP caching enabled, compile time differences may be negligible

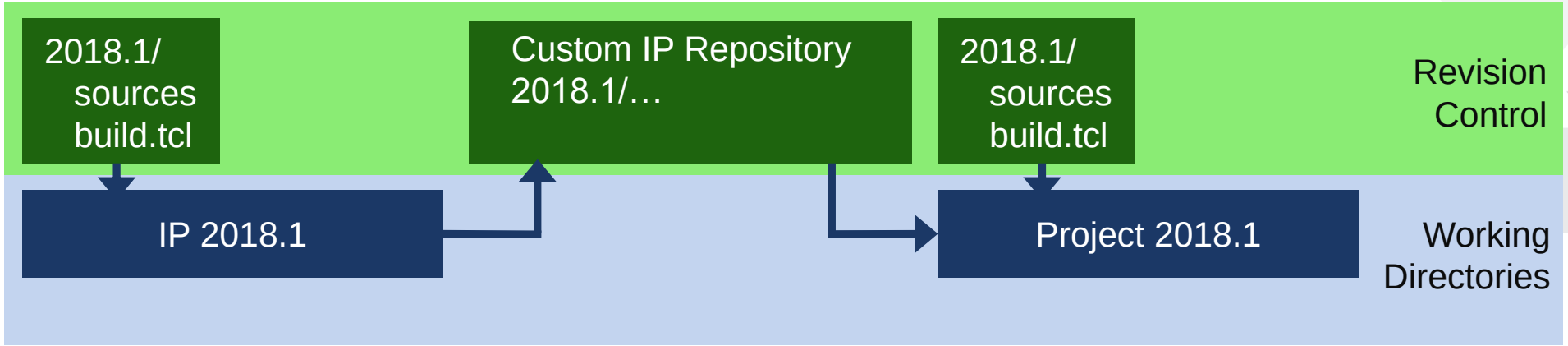
Avoid Using the DCP

- **A DCP generated out-of-context (OOC) is unconstrained**
- **IP are synthesized OOC**
- **Scoped timing constraints are used during the OOC synthesis run**
- **Timing constraints are discarded prior to writing the DCP**
- **Using the XCI or XCIX files ensure a fully constrained design**

Managing Custom IP Using Repositories

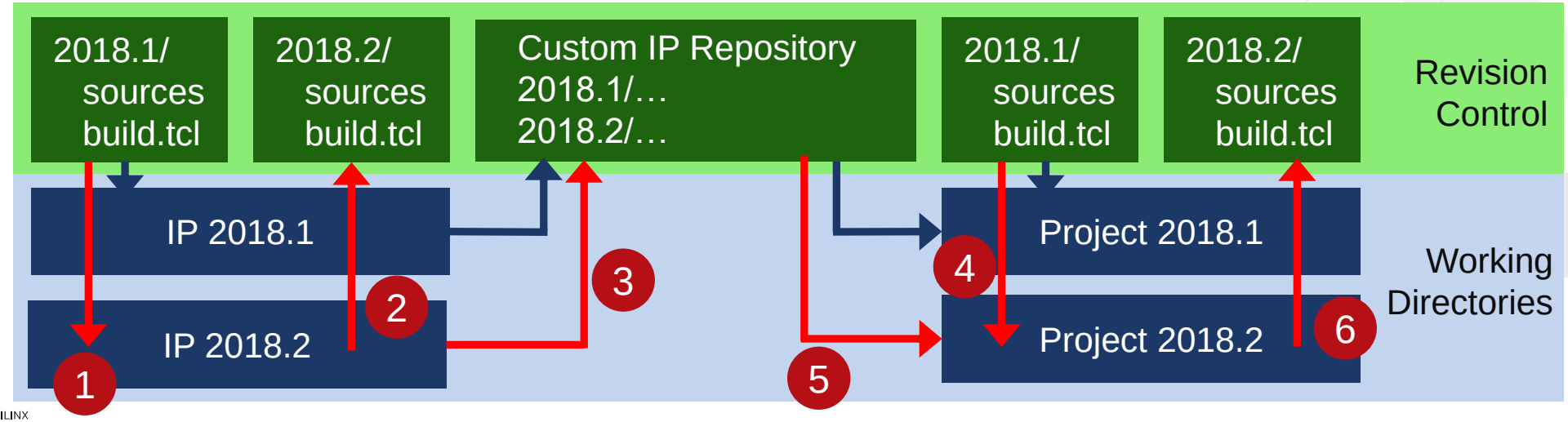
- **Develop IP in a revision controlled working directory**
- **Package the IP into a custom IP repository**
  - Follow Xilinx IP repository directory structure as a reference
  - **<vivado_version>**/data/ip/<company>/<IPname_version>
- **Working project directories**
  - Set IP repository path
  - Add XCI
  - Add both to the build script





Upgrading Custom IP (using Xilinx IP) Repositories

1. Create new IP directory from previous sources
2. Upgrade project and check in new IP files
3. Re-package all IP into new custom <vivado_version> repository
4. Rebuild project directory from previous sources
5. Update the IP repository
6. Upgrade project and check in new project files to repository



Propagation of Parameters in Block Designs

Xilinx IP Repository
vivado/<version>/data/ip/xilinx/

v_hdmi_tx_ss_0_v2

component.xml
RTL with parameters
constraints, etc.

2018.1

Workspace

design_1.bd

IP_ver = _v2
Non-default IP params
XCI name for each instance

Propagate parameters
Identify conflicts
Update XCI / BD

unique RTL for BD/IP, etc.

Source Repo

build.tcl
design_1/
  design_1.bd
  .ignore

v_hdmi_tx_ss_0_v3

component.xml
RTL with parameters
constraints, etc.

2018.2

design_1.bd

IP_ver = _v2
Non-default IP params
XCI name for each instance

_v2

build.tcl
design_1/
  design_1.bd
  ip/
    v_hdmi…_v2/*
  .ignore

Upgrade to V3

Preserving Block Designs

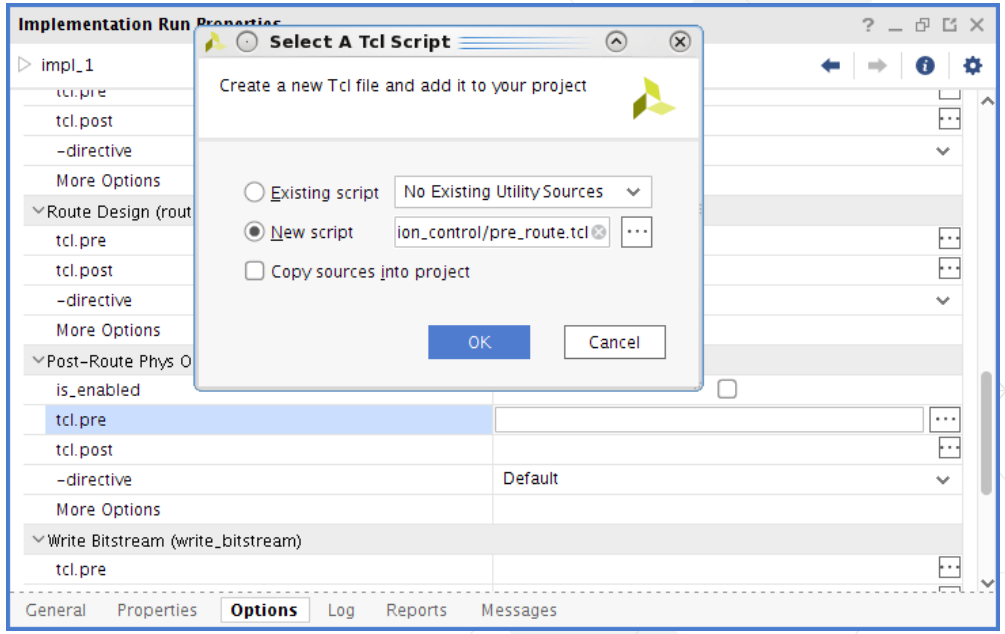
BD file to revision control	Size	Compile time	Preserve Layout	Forced to upgrade
BD	S	Slow[1]	N[2]	Y[3]
TCL (write_bd_tcl)	S	Slow[1]	Y[4]	Y
Whole BD directory	L	Fast	Y	N / locked

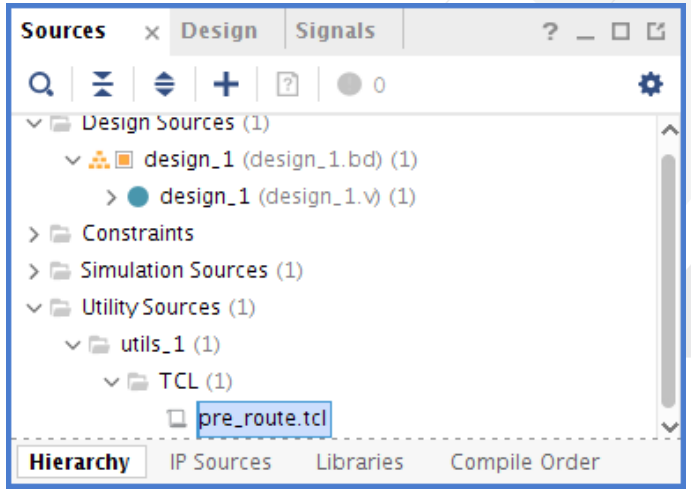
- **Recommendation**
  - Use write_bd_tcl to preserve entire BD including layout
  - If you want selective IP upgrade, then move towards BD

[1] With Out-of-context synthesis and IP caching enabled, compile time differences may be negligible
[2] Can be preserved by checking the BD/ui directory
[3] Not with selective IP upgrade. Generated output products of IP still need to be preserved
[4] Use –include_layout flag

Additional Files to Revision Control

- **Other source files**
  - XDC
  - Simulation test benches
  - Sysgen IP
  - HLS IP
  - Pre/post tcl scripts
  - Incremental compile DCPs
  - ELF files
- **Util_1 source set introduced in 2018.1**
  - Files are now managed by the project
  - Included in a project archive



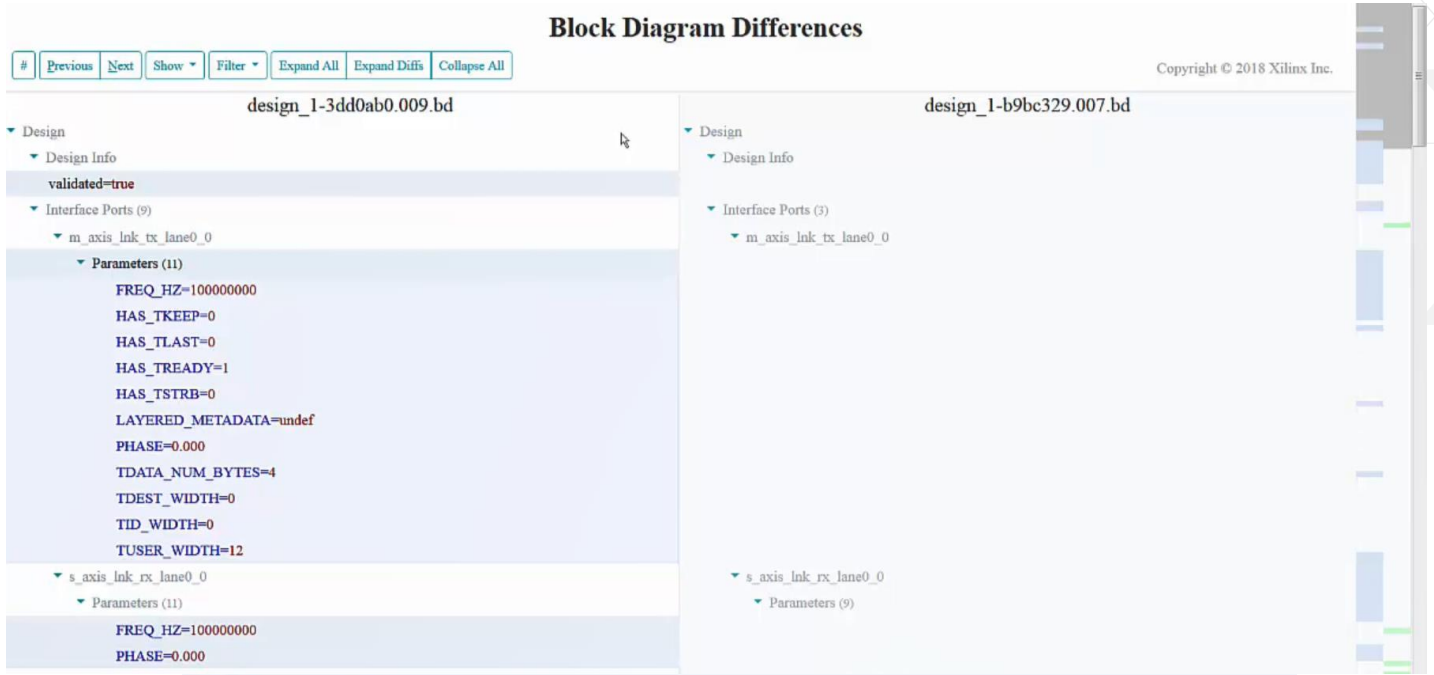


Output Files to Consider for Revision Control

- Simulation scripts for 3rd party simulators (export_simulation)
- Hardware definition files for SDK (export_hardware)
- DSAs for export to SDx (write_dsa)
- Bitstreams (generate_bitstream)
- Hardware debug (.ltx, .lpr, debug dashboards)
- Intermediate run results (runs / checkpoints)

Future Revision Control Improvements

- **Auto create .ignore files**
- **Separate output products from the sources**
- **Make the BD the one true source for a design**
- **BD Differences**
  - Compare two BD to understand the differences between the designs



Summary

- **Vivado provides the frameworks to develop your revision control strategy**
- **Six general steps**
  - Use scripted flows for revision control
  - Keep source files in a repository
  - Revision control the repository
  - Create a Tcl script to recreate the project
  - Revision control the script
  - Test your scripts

XDF
XILINX
DEVELOPER
FORUM