Create a portable Tcl for different Vivado versions

To create a portable Tcl script for Vivado block design recreation—usable across different Vivado versions—follow these best practices and recommendations:

Key Principles for Portability

- Focus on **script-based project creation** rather than GUI-based export artifacts, since Vivado's `.xpr` project files and IP core metadata are version-specific and often incompatible between releases.[1] [2]
- Always **version-control HDL sources, constraints, and all custom IP cores** separately from Vivado-generated folders and archives. Do not commit auto-generated or cache files in your repository.[3]

Recommended Script Structure

1. **Minimize Version-Specific Parameters**
   - Edit the generated Tcl to remove lines that forcibly specify a Vivado version, part number, or IP core revision.
   - Avoid referencing tool-specific or deprecated features.
2. **Use Standard Tcl Commands for Project Creation**
   - Use commands like `create_project`, `add_files`, `read_verilog`, `read_vhdl`, `add_files <constraints>`, and not tool-generated metadata.[2]
   - Separate project top-level creation (sources, constraints) from block design instantiation.
3. **Create Block Design with Exported Tcl**
   - When using `write_bd_tcl`, save your block design script (e.g., `design_1.tcl`) and source it in your main script:

     ```
     source $origin_dir/src/bd/design_1.tcl
     ```

   - This maintains the design hierarchy and logic while avoiding version-locked artifacts.[3]
4. **Enable IP Core Upgrades**
   - Add the following after your IP instantiation and block design creation:

     ```
     upgrade_ip [get_ips]
     ```

- This command helps update IP cores automatically to match the running Vivado version, if needed.[2]

5. **Avoid Hard-Coded Paths and Use Relative Locations**
   - Set project and file paths based on the script location:

```
set origin_dir [file dirname [info script]]
create_project myproject $origin_dir/myproject
```

6. **Document Customizations or Version-Specific Features**
   - If your design is dependent on features present only in certain Vivado versions, add comments and checks using Tcl logic to warn users.

Limitations & Migration Notes

- Vivado IP cores are versioned and may require **manual intervention** during upgrade (as new Vivado releases lock old IPs and force updates).[2]
- Always **test your Tcl script on multiple Vivado releases** if portability across versions is a requirement.
- When migrating, generate a new block design Tcl in the newer Vivado version if you encounter compatibility or locked IP core issues.

Example Skeleton

```
# Portable Vivado project creation
set origin_dir [file dirname [info script]]
create_project myproject $origin_dir/myproject -part <device_part>
add_files $origin_dir/src/hdl/*.v
add_files $origin_dir/src/constraints/*.xdc

# Block design recreation
source $origin_dir/src/bd/design_1.tcl
make_wrapper -files [get_files design_1.bd] -top
add_files -norecurse $origin_dir/myproject/design_1_wrapper.v

# Upgrade IPs for new Vivado releases
upgrade_ip [get_ips]

# Synthesis and implementation
launch_runs synth_1 impl_1 -jobs 4
```

By following this approach, your Tcl script should remain as portable as possible between Vivado versions, provided you handle IP core migration per release and maintain source compatibility.[1] [3] [2]

⁂

1. https://docs.amd.com/r/en-US/ug994-vivado-ip-subsystems/Using-Tcl-Scripts-to-Create-Projects-and-Block-Designs

2. https://xillybus.com/tutorials/vivado-version-control-packaging
3. https://www.fpgadeveloper.com/2014/08/version-control-for-vivado-projects.html/
4. https://grittyengineer.com/creating-vivado-ip-the-smart-tcl-way/
5. https://docs.amd.com/r/en-US/ug895-vivado-system-level-design-entry/Working-with-Tcl
6. https://www.xilinx.com/support/documents/sw_manuals/xilinx2022_1/ug835-vivado-tcl-commands.pdf
7. https://www.xilinx.com/support/documents/sw_manuals/xilinx2022_2/ug904-vivado-implementation.pdf
8. https://www.reddit.com/r/FPGA/comments/11jaz0d/different_versions_of_vivado/
9. https://stackoverflow.com/questions/57640831/how-to-modify-vivado-2018-3-generated-tcl-script-for-version-control
10. https://www.xilinx.com/support/documents/sw_manuals/xilinx2022_1/ug911-vivado-migration.pdf
11. https://xilinx.github.io/vck190-base-trd/2020.2/html/build-vivado.html
12. https://www.youtube.com/watch?v=NLZEdx7zIoA
13. https://static.eetrend.com/files/2021-09/wen_zhang_/100553629-220271-ug904-vivado-implementation.pdf
14. https://www.fpgarelated.com/thread/14179/vivado-tcl
15. https://www.reddit.com/r/FPGA/comments/4yyo78/vivado_development_best_practices/
16. https://docs.amd.com/r/en-US/ug994-vivado-ip-subsystems/Exporting-a-Block-Design-to-a-Tcl-Script-in-the-IDE
17. https://www.xilinx.com/support/documents/sw_manuals/xilinx2022_2/ug894-vivado-tcl-scripting.pdf
18. https://forum.digilent.com/topic/25590-vivado-tcl-build-scripts/
19. https://adaptivesupport.amd.com/s/question/0D52E00006iHk37SAC/vivado-and-version-control?language=en_US
20. https://docs.amd.com/r/en-US/ug894-vivado-tcl-scripting