# NetFPGA Summer Course


**Presented by:**

**Andrew W Moore, Noa Zilberman, Gianni Antichi**
**Stephen Ibanez, Marcin Wojcik, Jong Hun Han,**
**Salvator Galea, Murali Ramanujam, Jingyun Zhang,**
**Yuta Tokusashi**

**University of Cambridge**
**July 24 – July 28, 2017**

http://NetFPGA.org

# DESIGNING CORES

# Outline

- **What is a core?**
- **IP Core logic**
- **IP cores packaging**
  - Vivado
  - TCL
- **Instantiating IPs**
- **Using Subcores**
- **Compile**
- **Do’s and Don’ts**

# The role of Cores

- **A Core (also known as IP Core) is a stand alone module**
- **Can be reused**
  - Within a design
  - Between designs
- **Can be configured**
- **Can be written in different languages**
  - Verilog, VHDL, system Verilog, C ….

- **The module is “packaged” as a core**

# IP Core Logic

- **Design your module**
- **“Ignore” the top project**
- **Can be anything from one HDL file to a complex design**
- **Test you core in a simulation**
  - Write a core-specific test bench
  - Not a must
- **Set timing constraints**

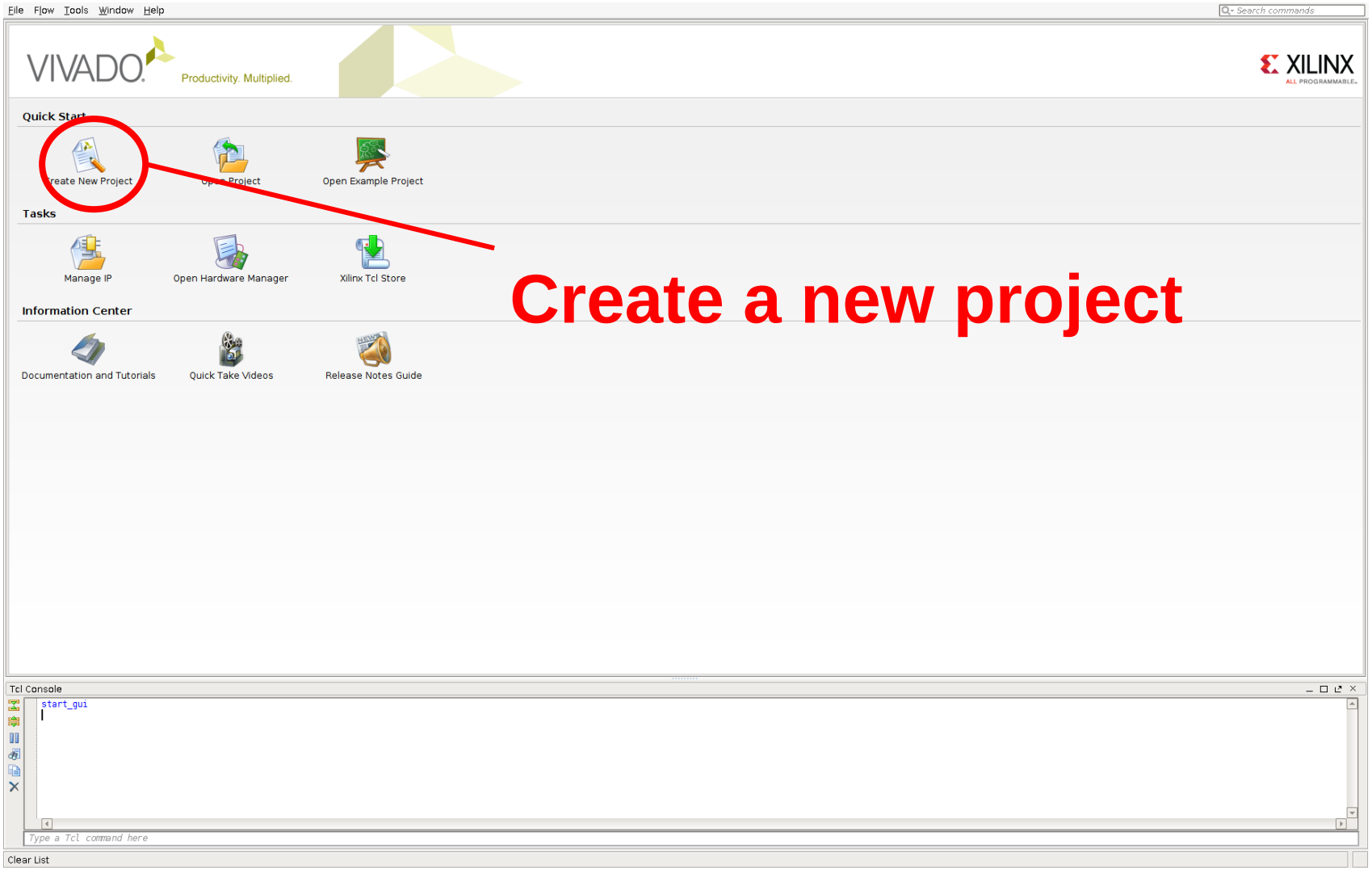
- **All done?**
  - Time to wrap your core

# Packaging Cores

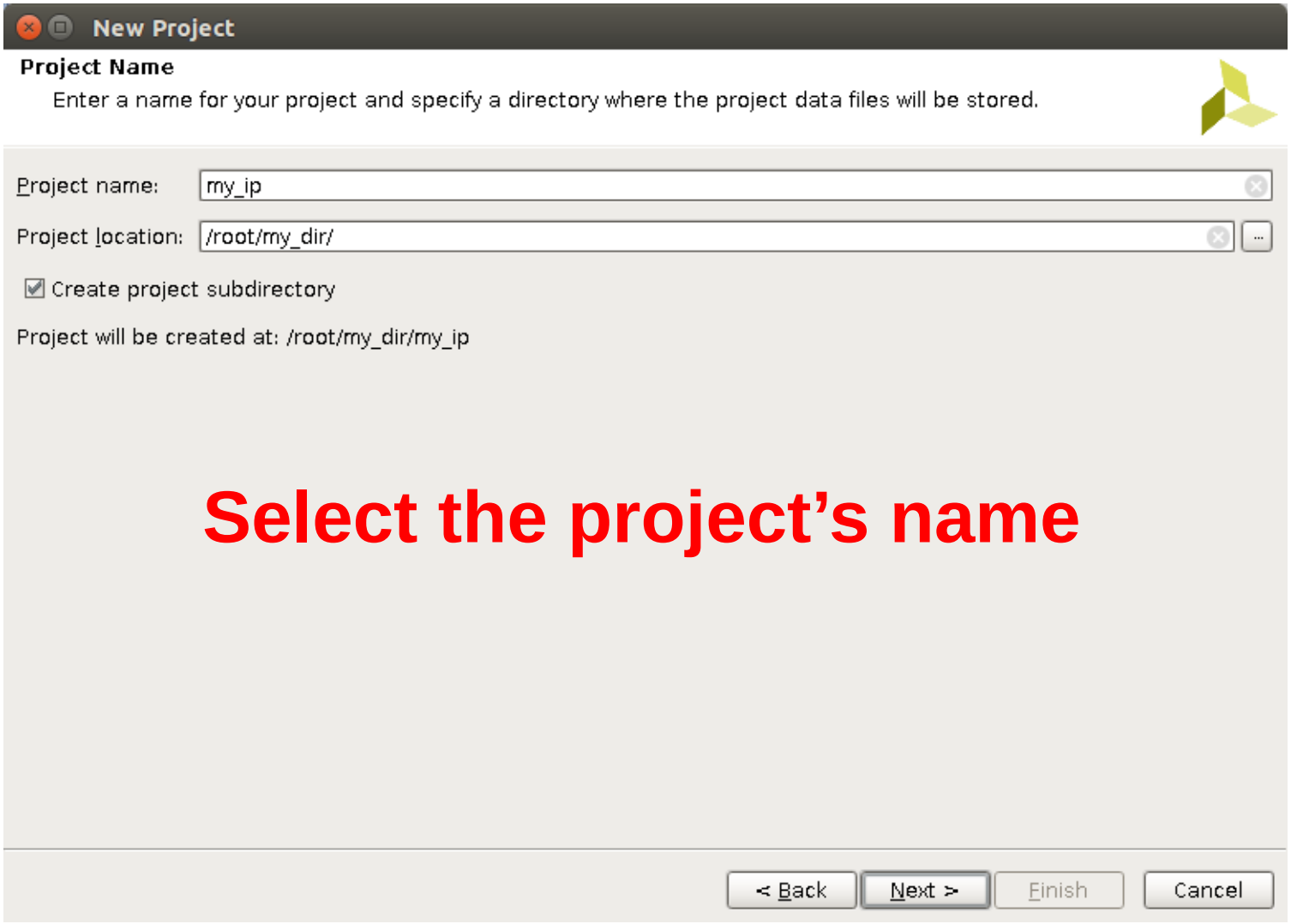
- **There are (at least) two ways to package a core:**
  - Through the Vivado GUI
  - Using TCL scripts
- **We will explore both**

- **For best reuse across projects, we recommend using TCL scripts**
  - You can use the GUI and still export TCL
  - But they are not fully compatible

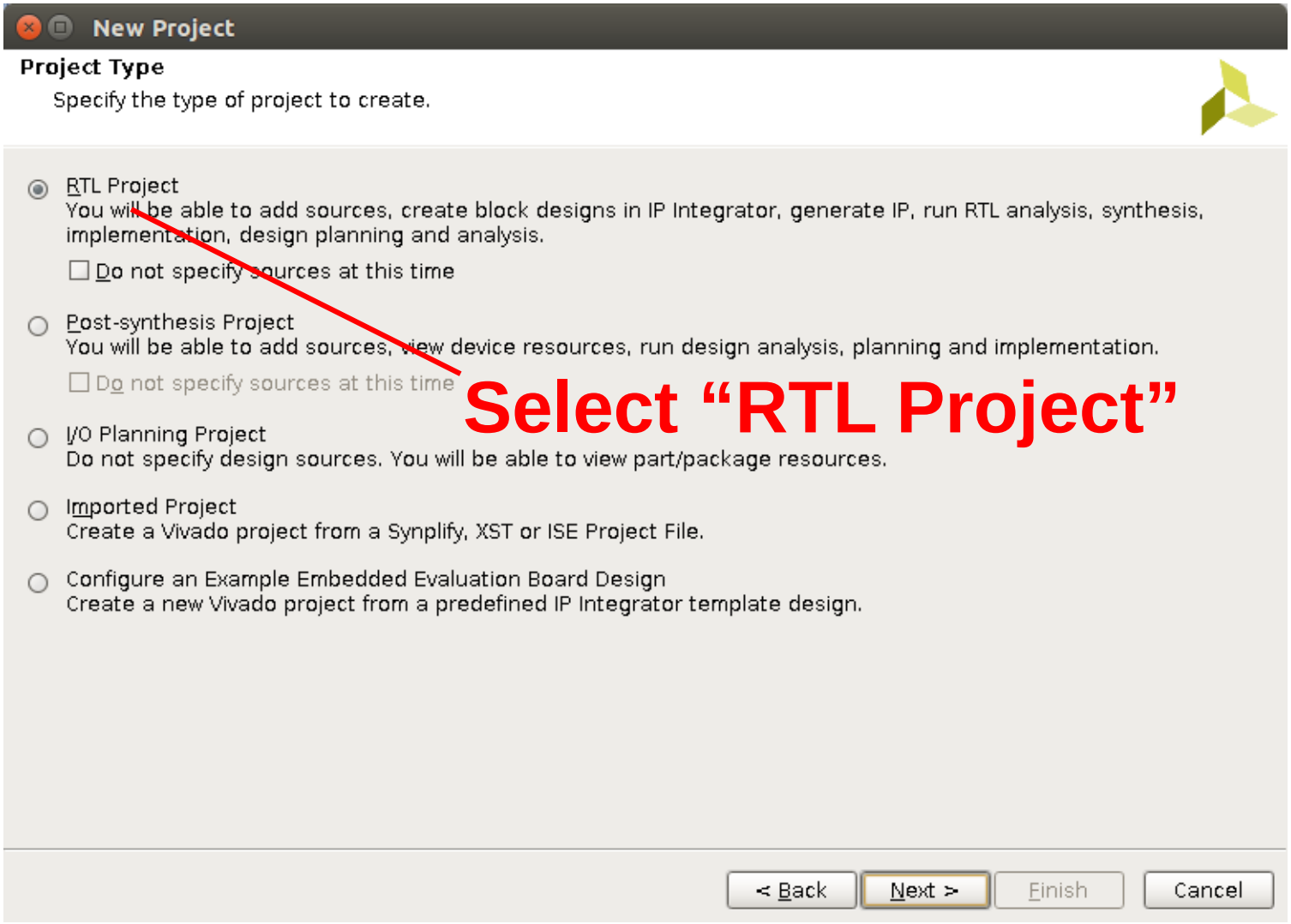
# Packaging a Core using Vivado



# Packaging a Core using Vivado (2)

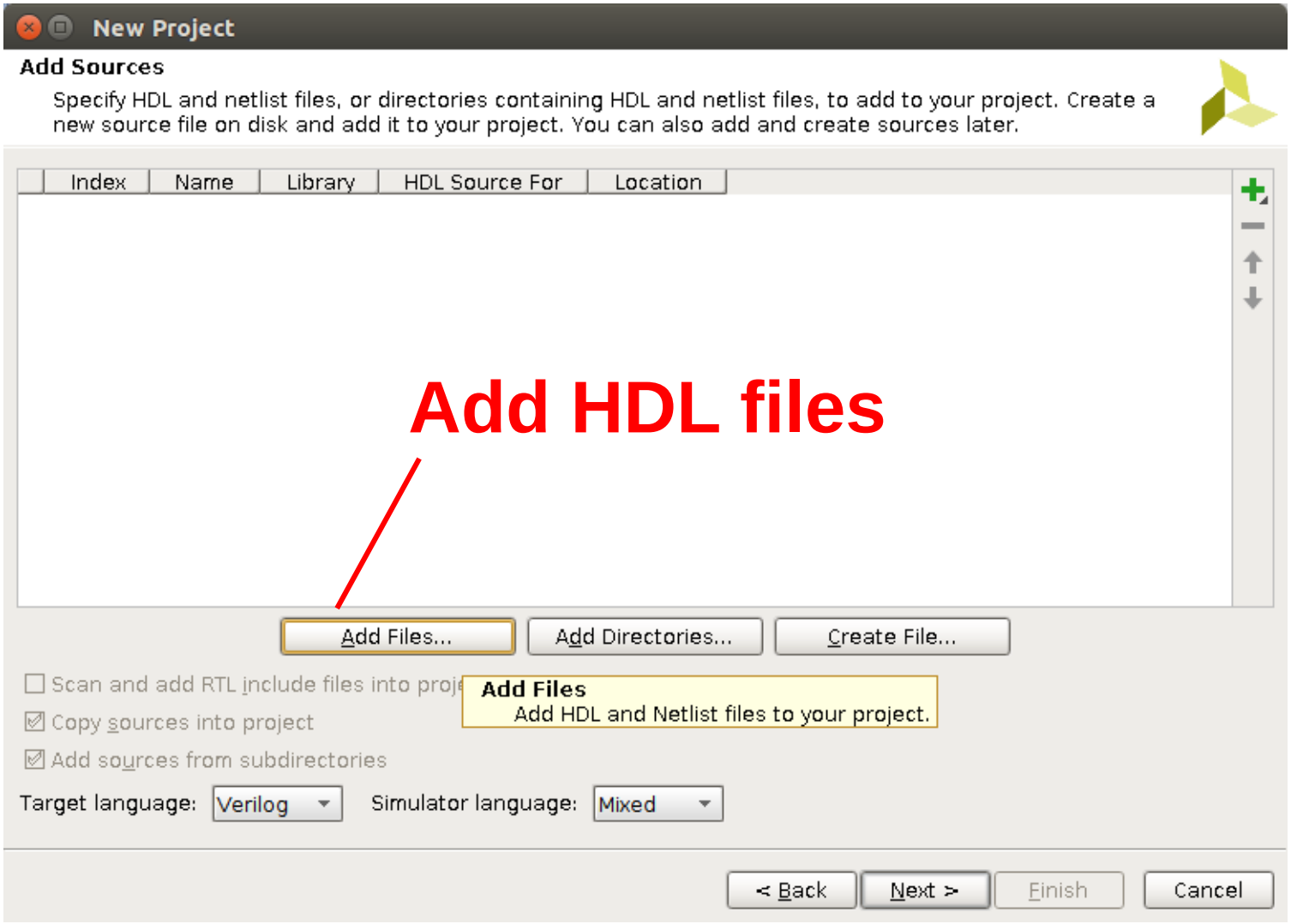


# Packaging a Core using Vivado (3)

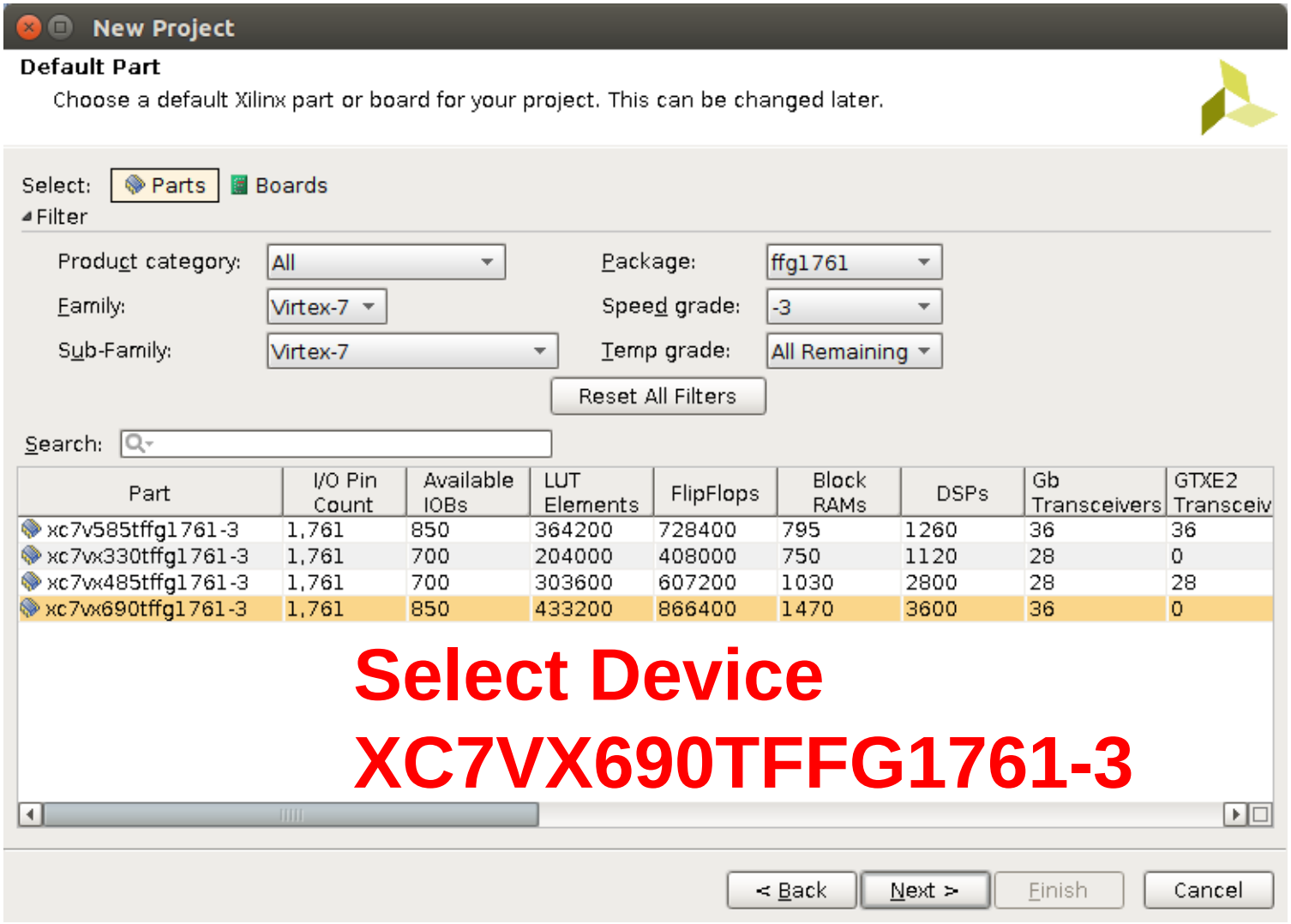



Select “RTL Project”

# Packaging a Core using Vivado (4)

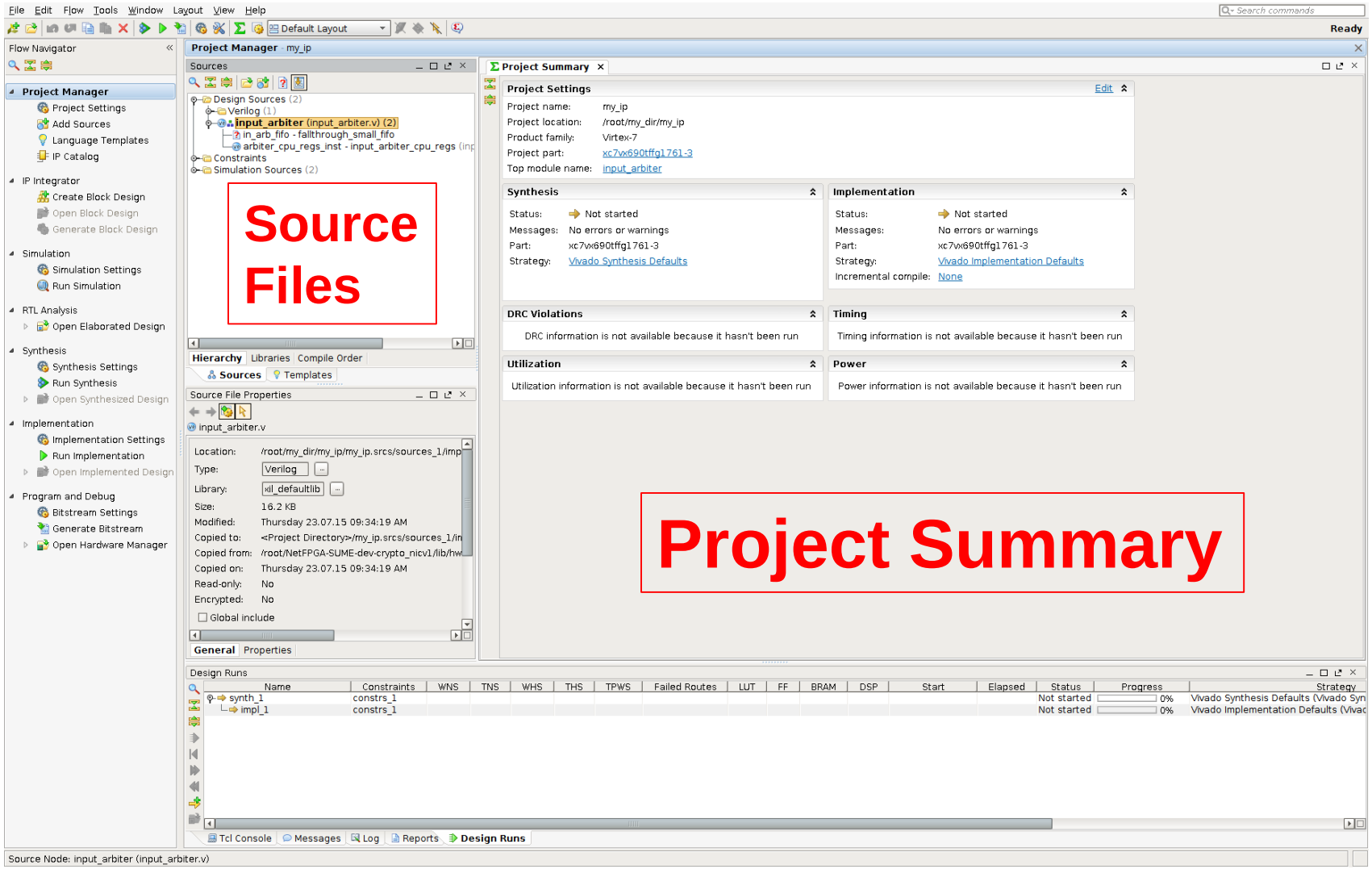


# Packaging a Core using Vivado (5)



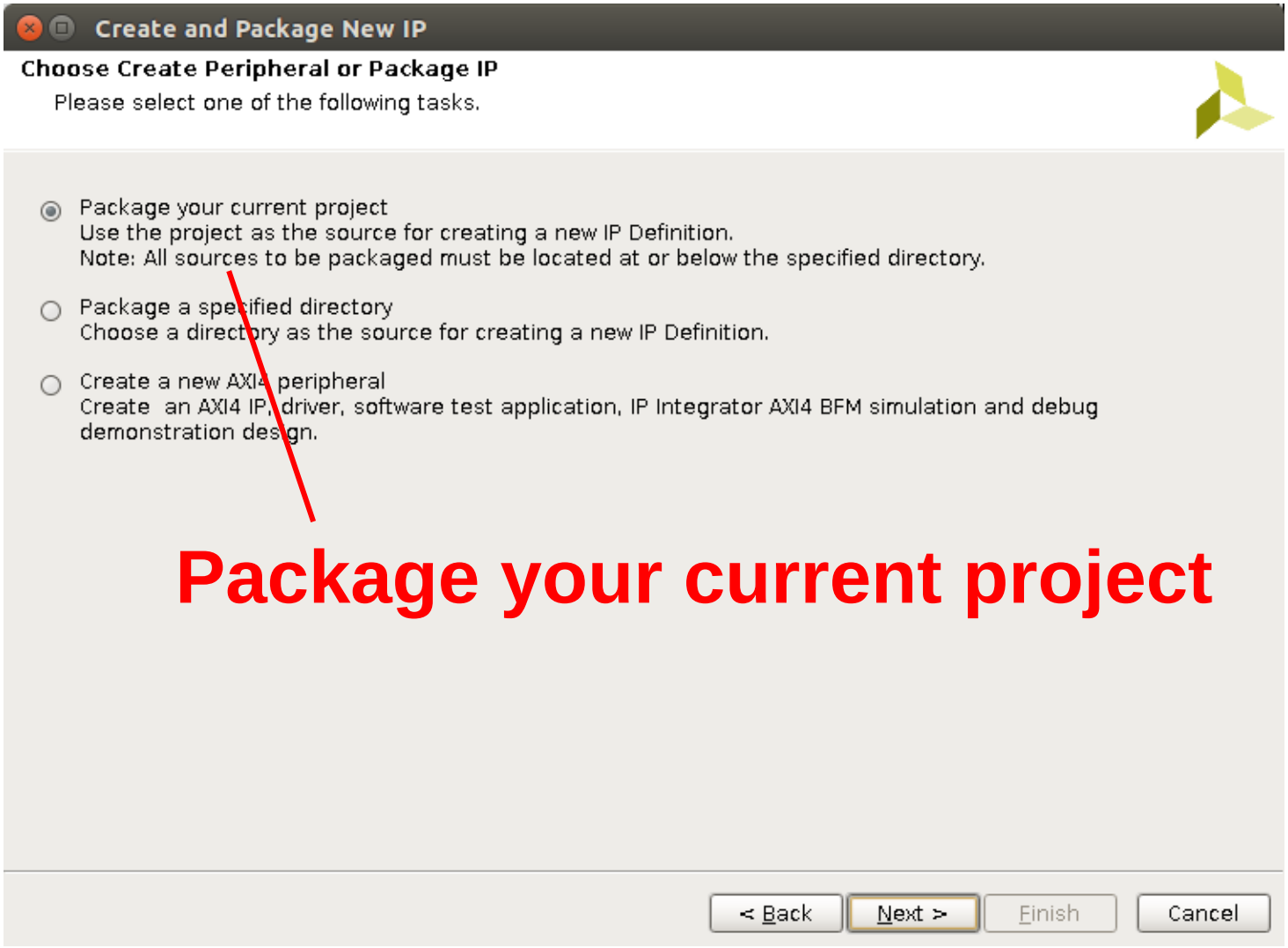
# Packaging a Core using Vivado (6)

# Packaging a Core using Vivado (7)

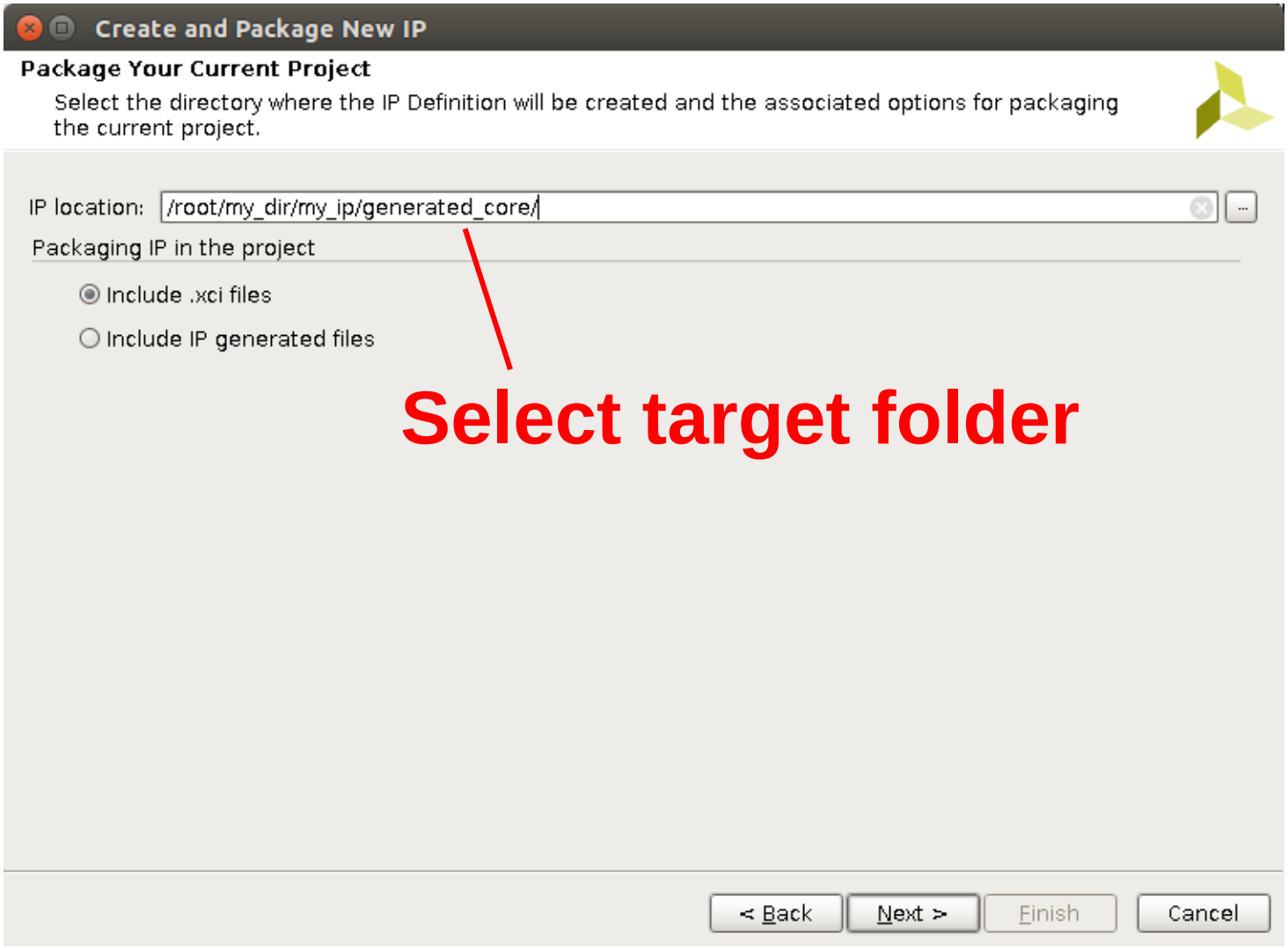


# Packaging a Core using Vivado (8)

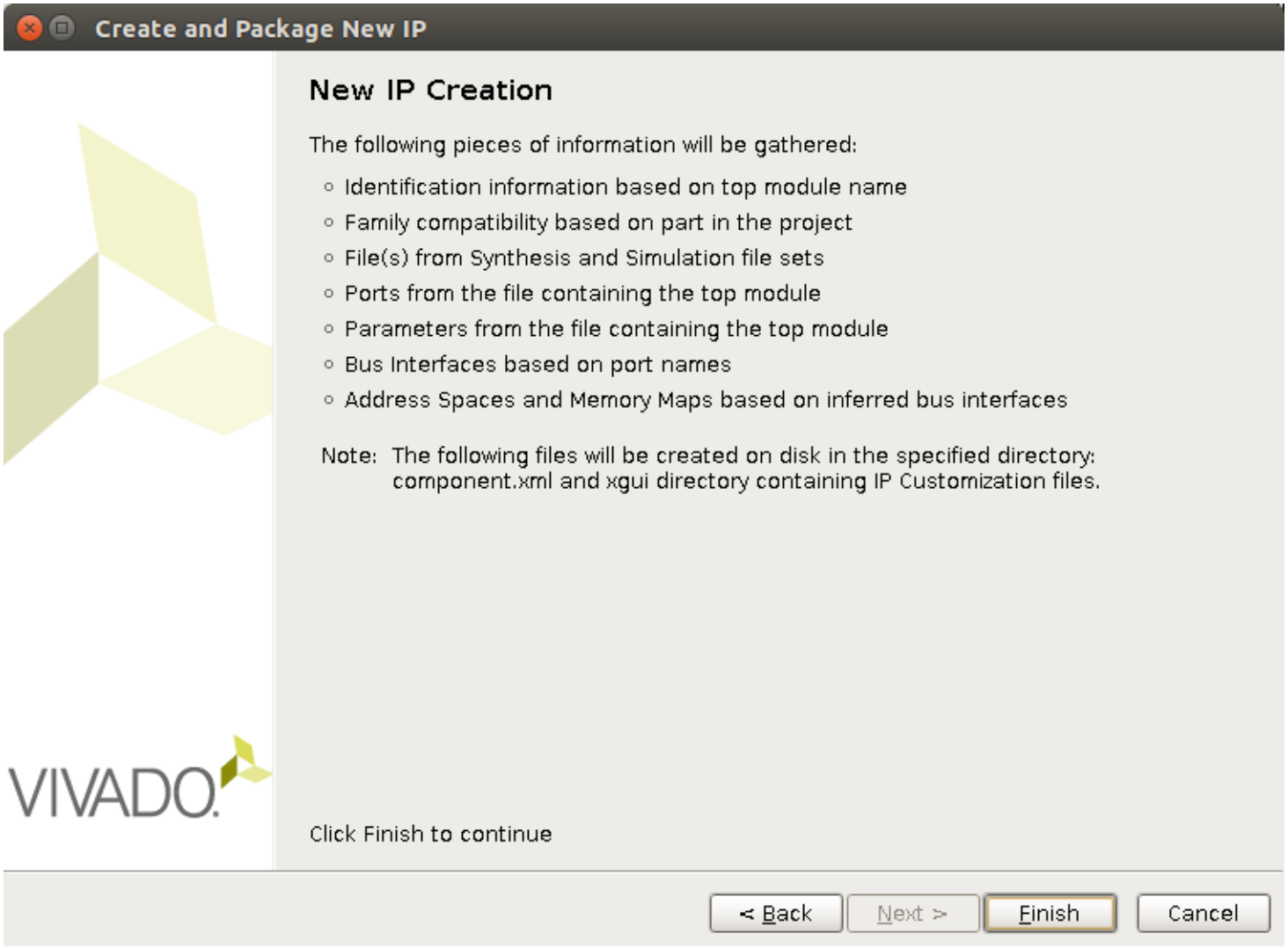
# Packaging a Core using Vivado (9)



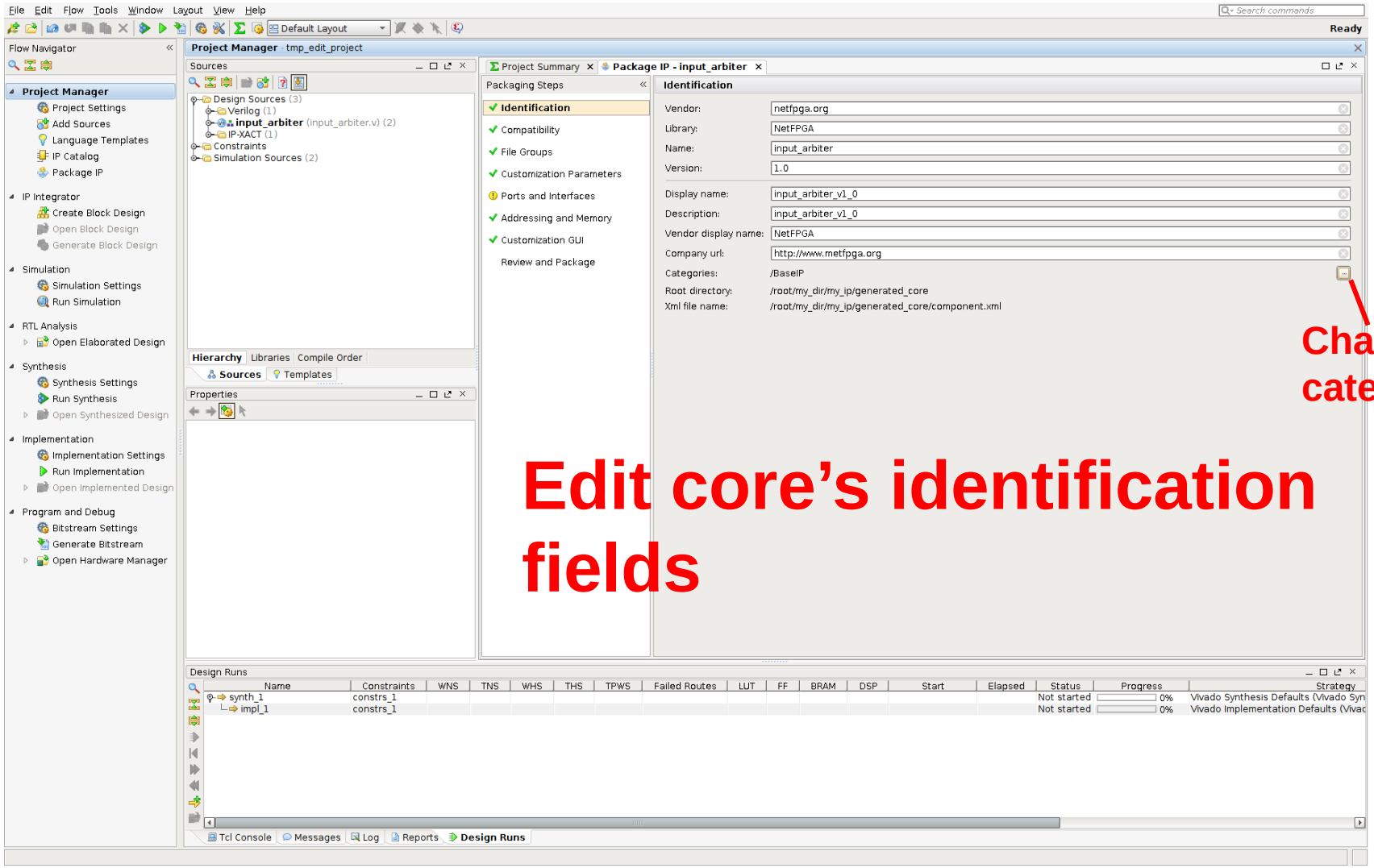
# Packaging a Core using Vivado (10)



# Packaging a Core using Vivado (11)

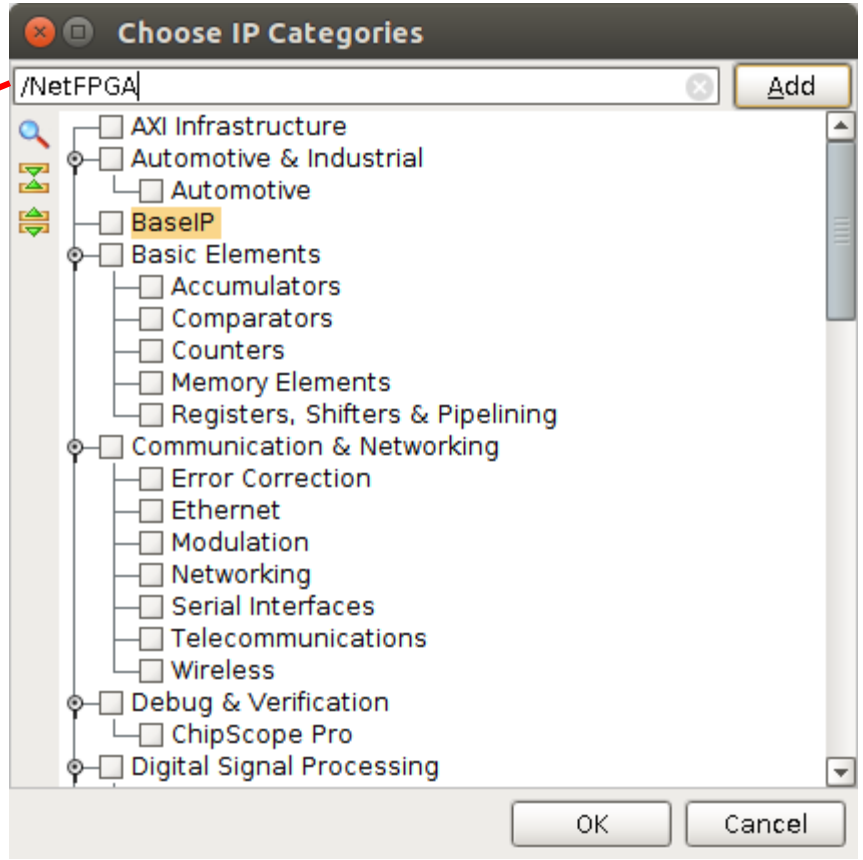


# Packaging a Core using Vivado (12)

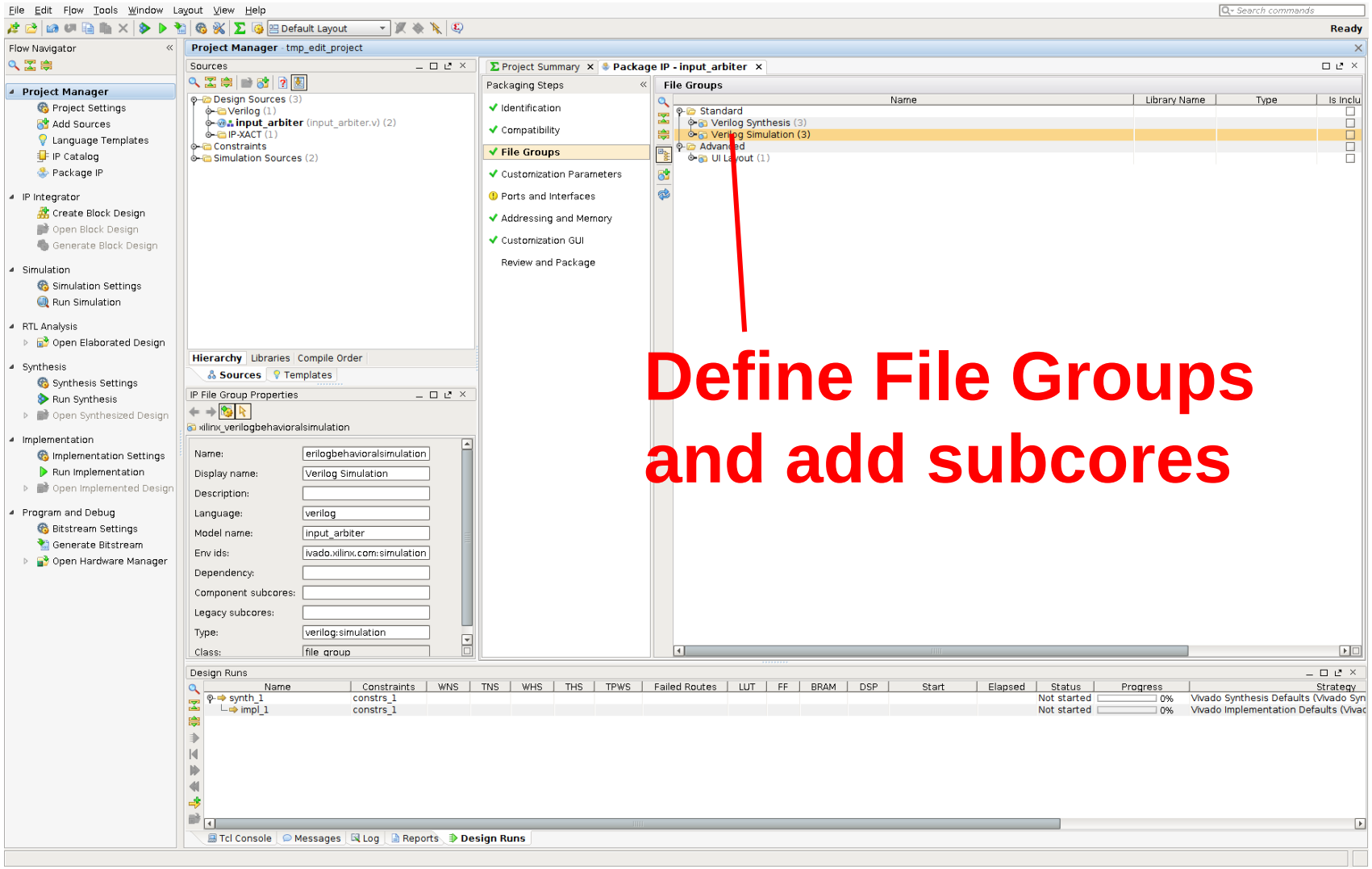


# Packaging a Core using Vivado (13)

**Select category or create a new one**

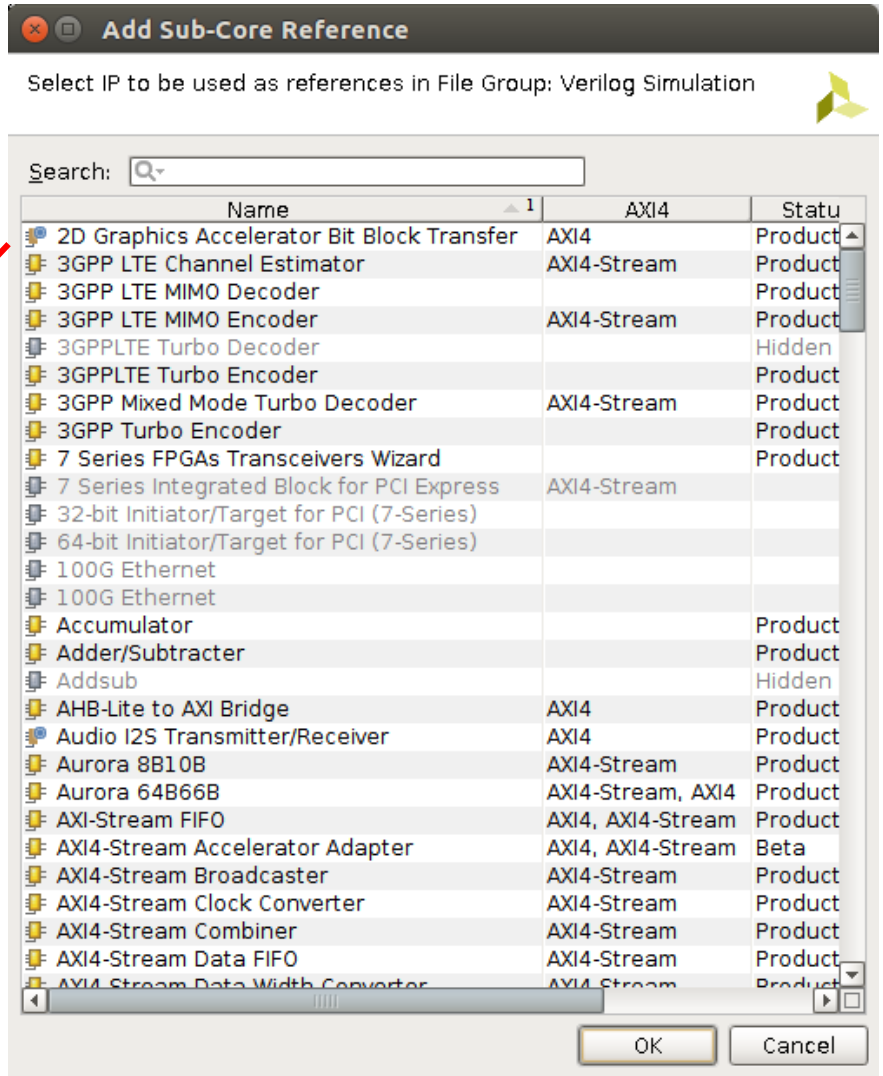


# Packaging a Core using Vivado (14)



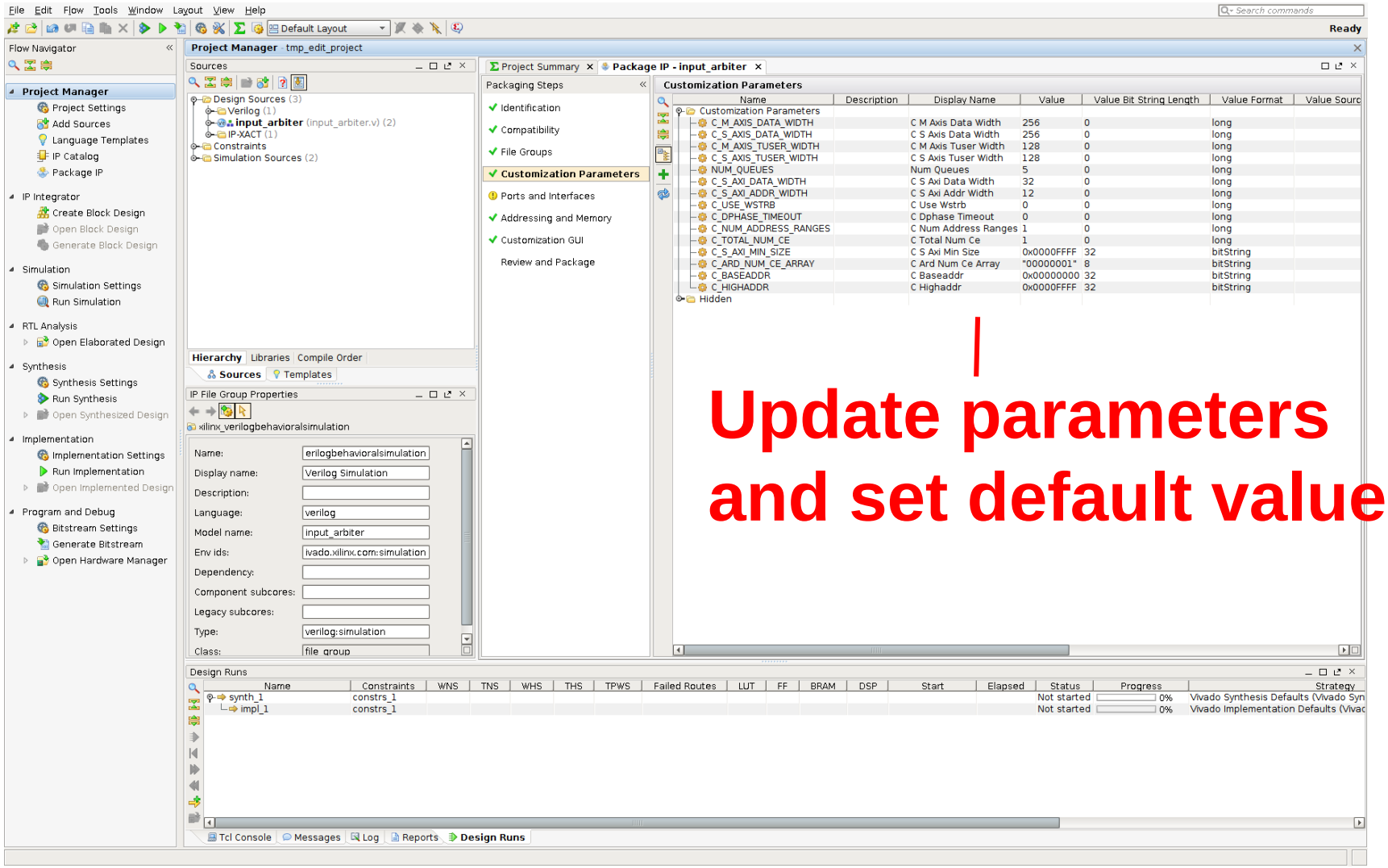
# Packaging a Core using Vivado (15)

Select subcore

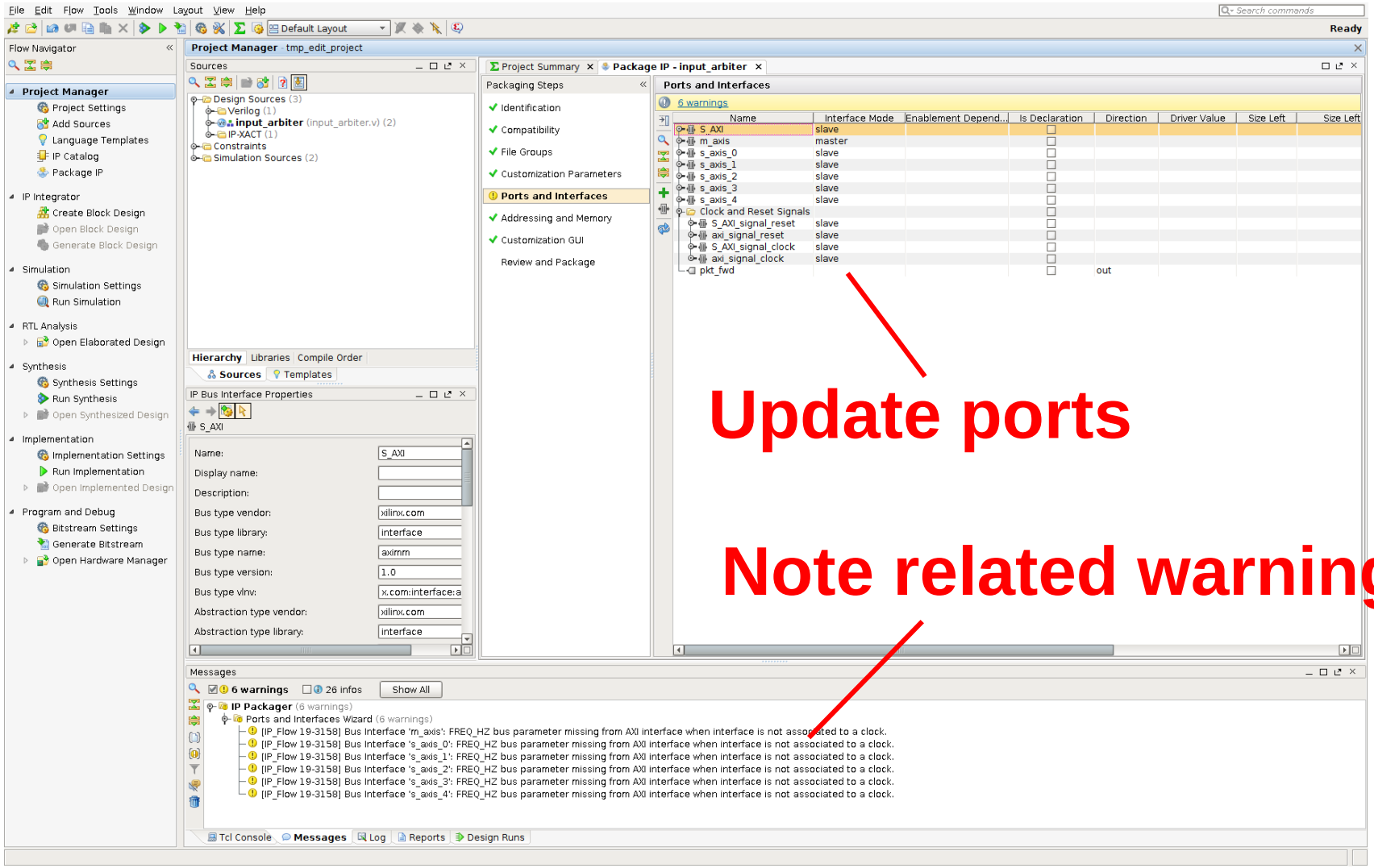



**A subcore is an IP instantiated within the core**

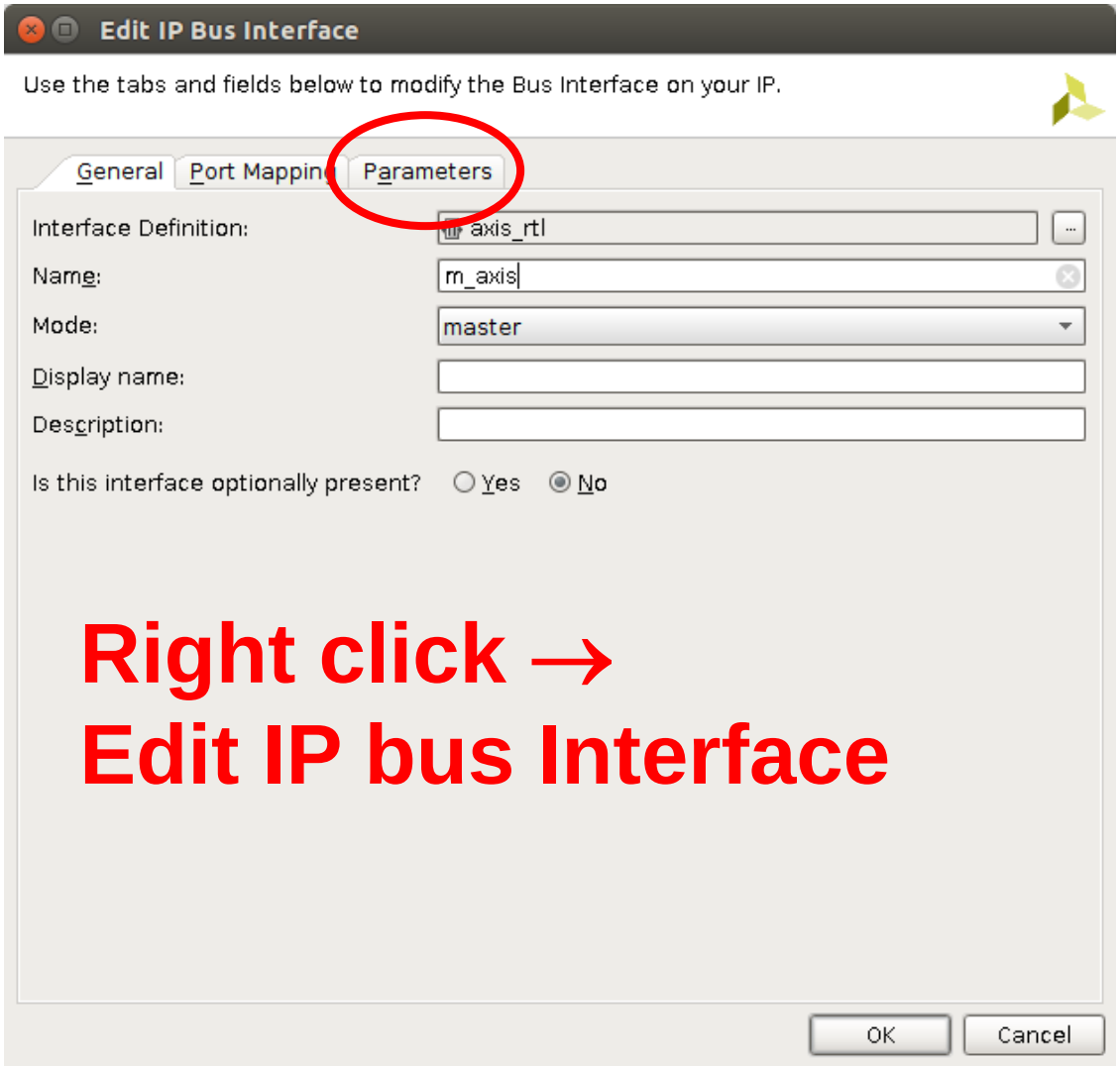
# Packaging a Core using Vivado (16)



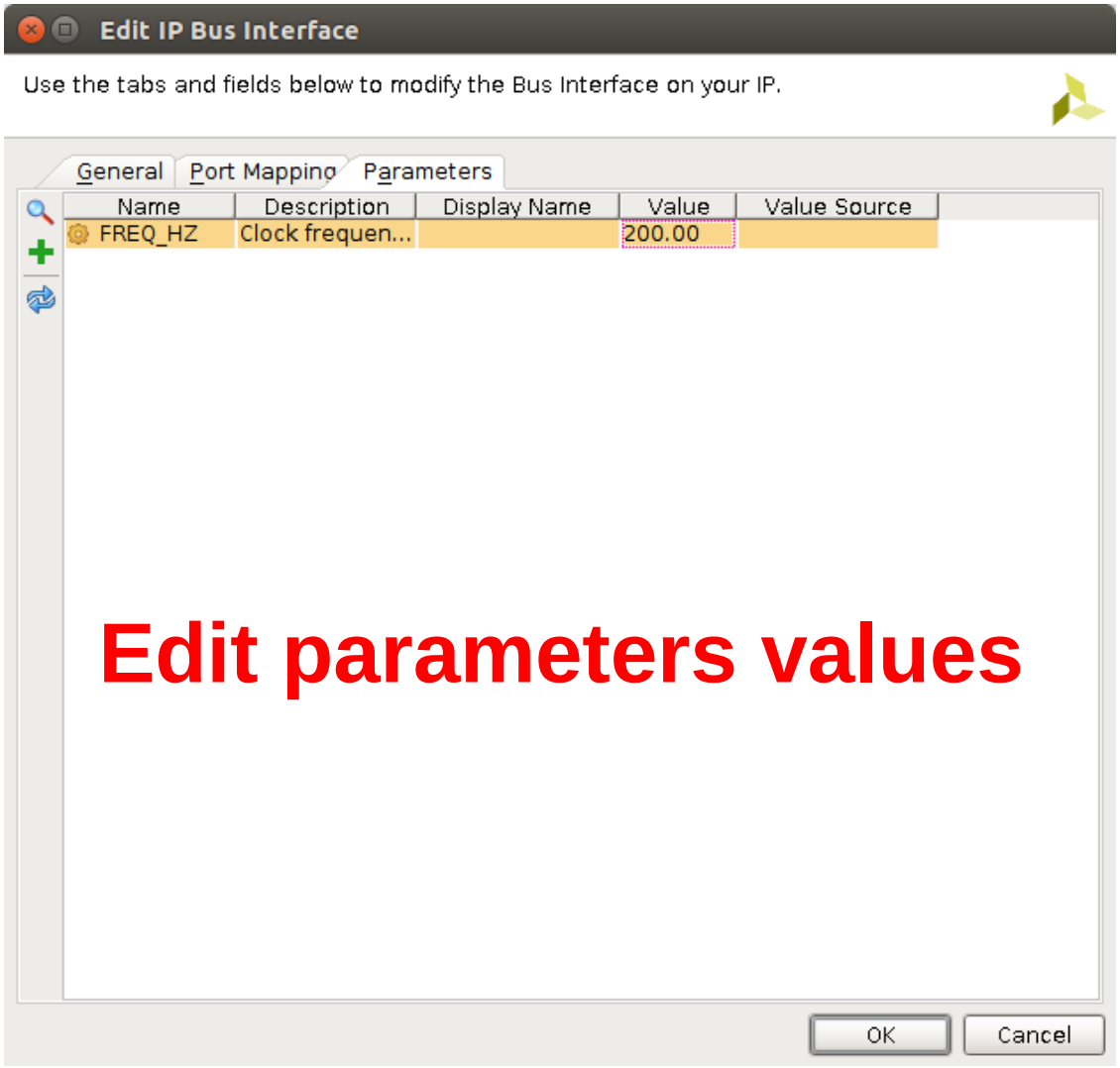
# Packaging a Core using Vivado (17)



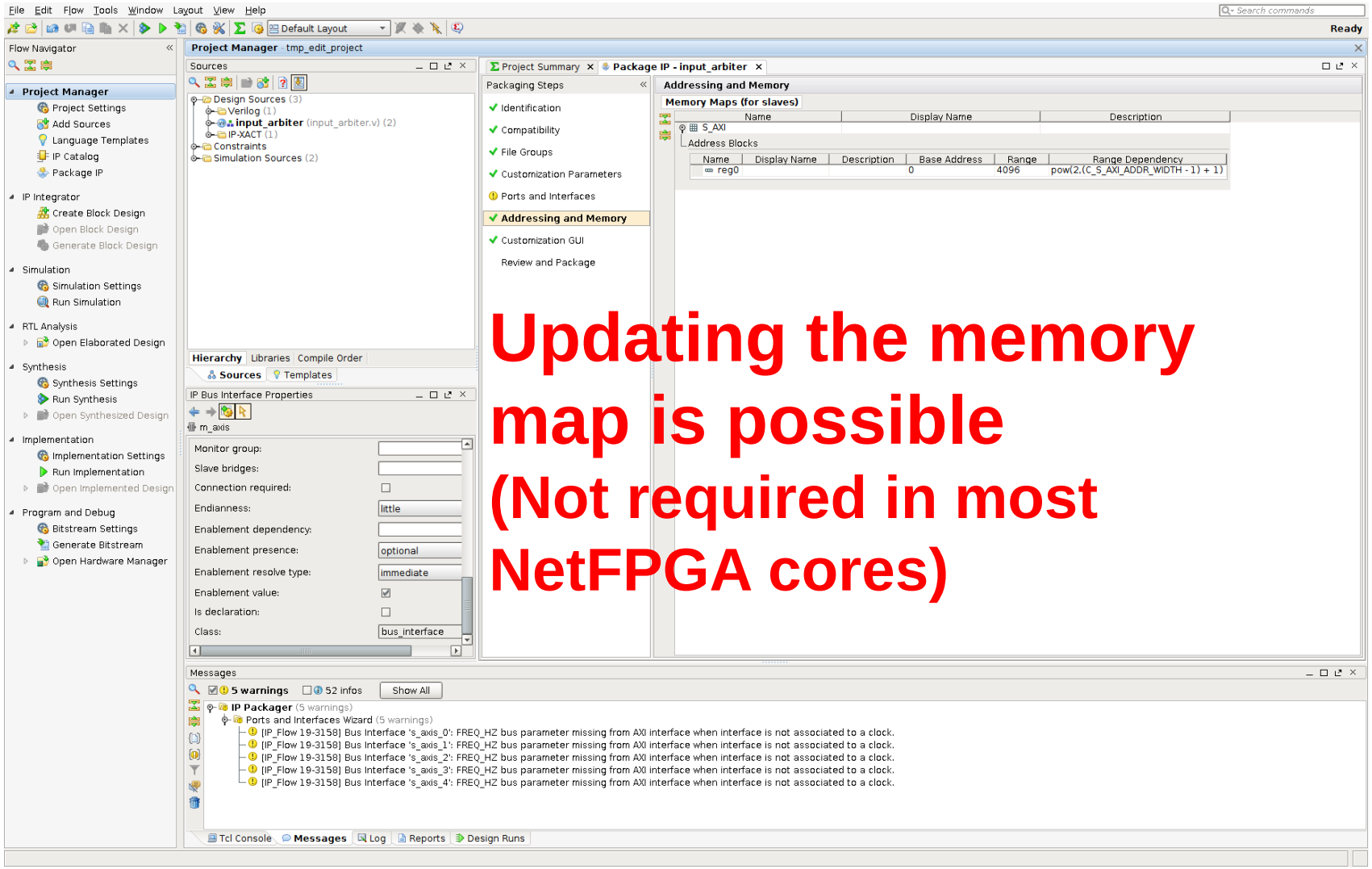
# Packaging a Core using Vivado (18)



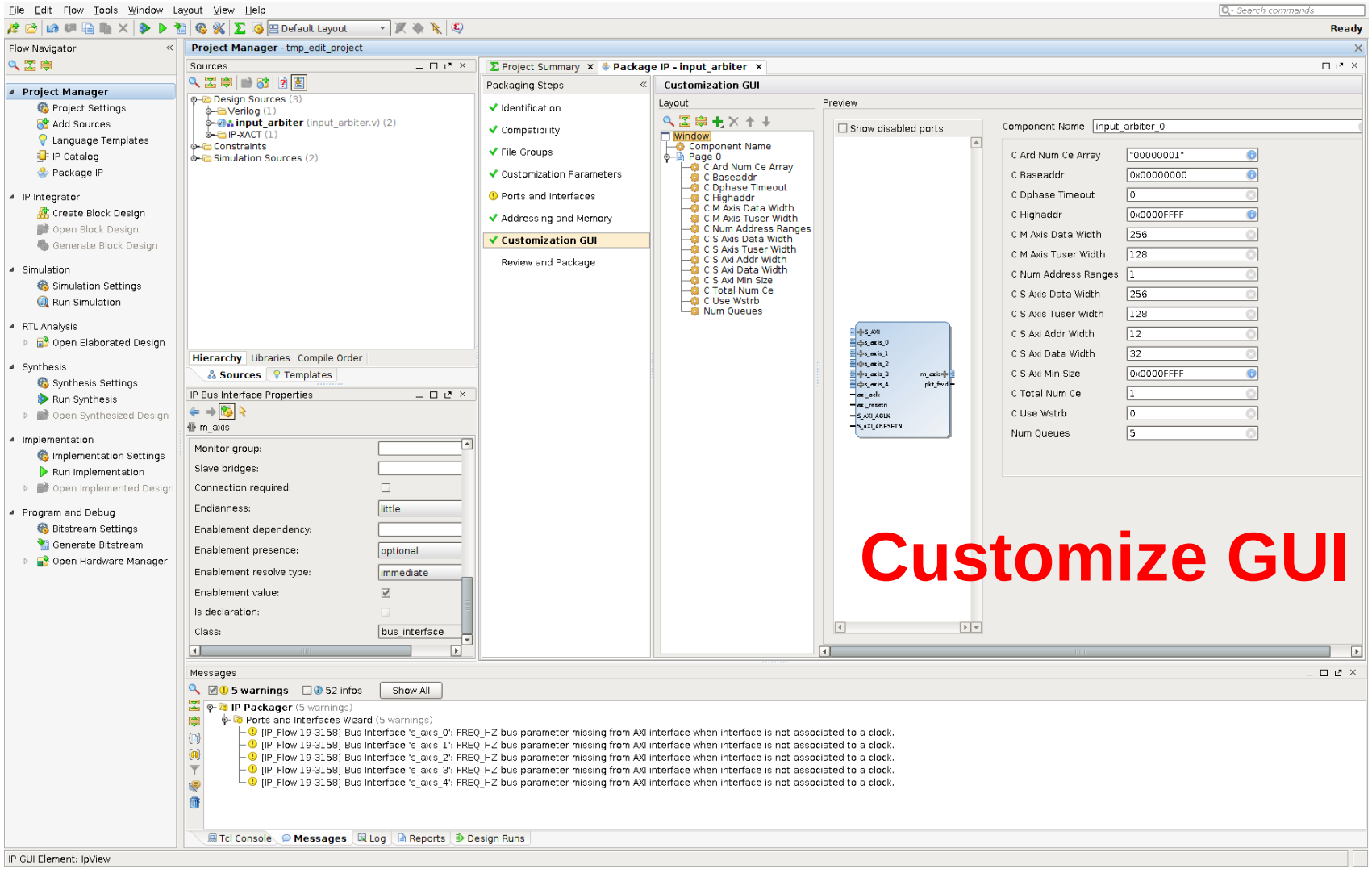
# Packaging a Core using Vivado (19)



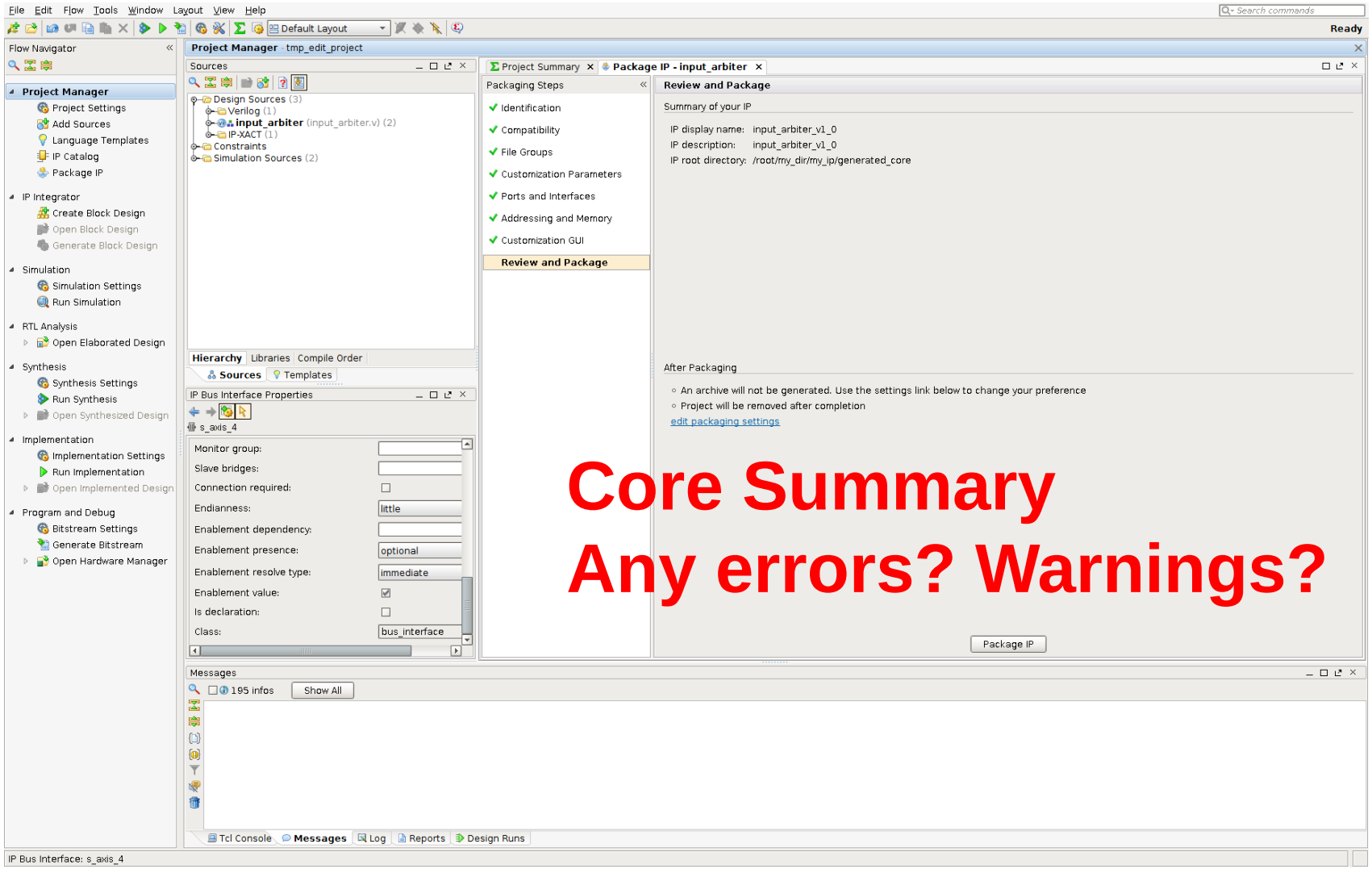
# Packaging a Core using Vivado (20)



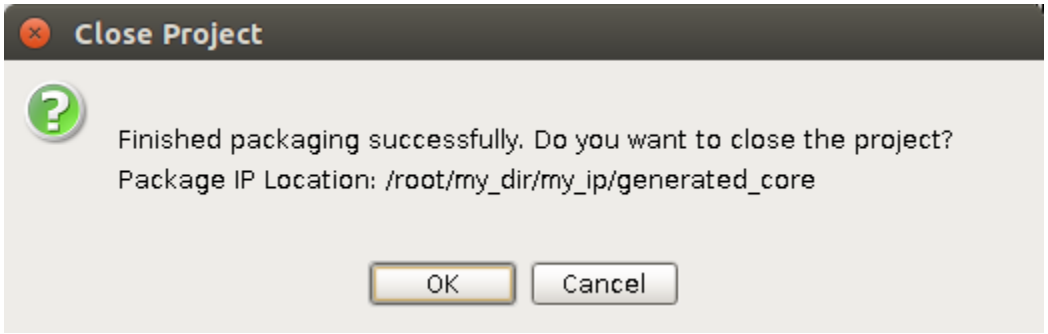
# Packaging a Core using Vivado (21)



# Packaging a Core using Vivado (22)



# Packaging a Core using Vivado (23)




**Job done!**

# Packaging a core using TCL

- **Start from a template of an existing core**
- **Place all your HDL files under <core_name>/hdl**
- **Edit <core_name>.tcl**
- **Update Makefile with the name of the core**
- **Run make**

- **You may want to add your core to $SUME_FOLDER/Makefile as well**
  - Note that the order of generation matters

# <core_name>.tcl

- **TCL file structure:**
  - Project Defines
  - Creating the project
  - Adding the HDL files
  - Packaging the project
  - Adding core information & parameters
  - Validation
  - Completing the project

# <core_name>.tcl

- **TCL file structure:**
  - Project Defines

```
set design <core_name>
set top <core_name>
set device xc7vx690t-3-ffg1761
set proj_dir ./ip_proj
set ip_version 1.00
set lib_name NetFPGA
```

**Recommend to keep identical**

**Recommend not to change**

# <core_name>.tcl

- **TCL file structure:**
  - Creating the project

```
create_project -name ${design} -force –dir "./${proj_dir}"
        -part ${device} -ip
set_property source_mgmt_mode All [current_project]
set_property top ${top} [current_fileset]
set_property ip_repo_paths
$::env(SUME_FOLDER)/lib/hw/
        [current_fileset]
puts "Creating <core name> IP"
```

# <core_name>.tcl

- **TCL file structure:**
  - Adding the HDL files

```
read_verilog "./hdl/<some file>.v"
read_verilog "./hdl/<core_name>_cpu_regs_defines.v"
read_verilog "./hdl/<core_name>_cpu_regs.v"
read_verilog "./hdl/<core_name>.v"
update_compile_order -fileset sources_1
update_compile_order -fileset sim_1
```

# <core_name>.tcl

- **TCL file structure:**
  - Adding core information & parameters

package_project

# <core_name>.tcl

- **TCL file structure:**
  - Packaging the project – core information

```
set_property name ${design} [ipx::current_core]
set_property library ${lib_name} [ipx::current_core]
set_property vendor_display_name {NetFPGA} [ipx::current_core]
set_property company_url {www.netfpga.org} [ipx::current_core]
set_property vendor {NetFPGA} [ipx::current_core]
....
set_property version ${ip_version} [ipx::current_core]
update_ip_catalog -rebuild
```

# <core_name>.tcl

- **TCL file structure:**
  - Packaging the project – parameters

```
ipx::infer_user_parameters [ipx::current_core]

ipx::add_user_parameter {PARAM_NAME} [ipx::current_core]
set_property value_resolve_type {user} [ipx::get_user_parameter
                PARAM_NAME [ipx::current_core]]
set_property display_name {PARAM_NAME}
                [ipx::get_user_parameter PARAM_NAME
                [ipx::current_core]]
set_property value {<some value>} [ipx::get_user_parameter
                PARAM_NAME [ipx::current_core]]
set_property value_format {long} [ipx::get_user_parameter
                PARAM_NAME [ipx::current_core]]
```

# <core_name>.tcl

- **TCL file structure:**
  - Packaging the project – bus parameters

```
ipx::add_bus_parameter FREQ_HZ [ipx::get_bus_interfaces
m_axis –
                                of_objects [ipx::current_core]]
ipx::add_bus_parameter FREQ_HZ [ipx::get_bus_interfaces s_axis
–
                                of_objects [ipx::current_core]]
```

# <core_name>.tcl

- **TCL file structure:**
  - Validation

```
ipx::check_integrity [ipx::current_core]
```

**Read the output and look for reported issues**

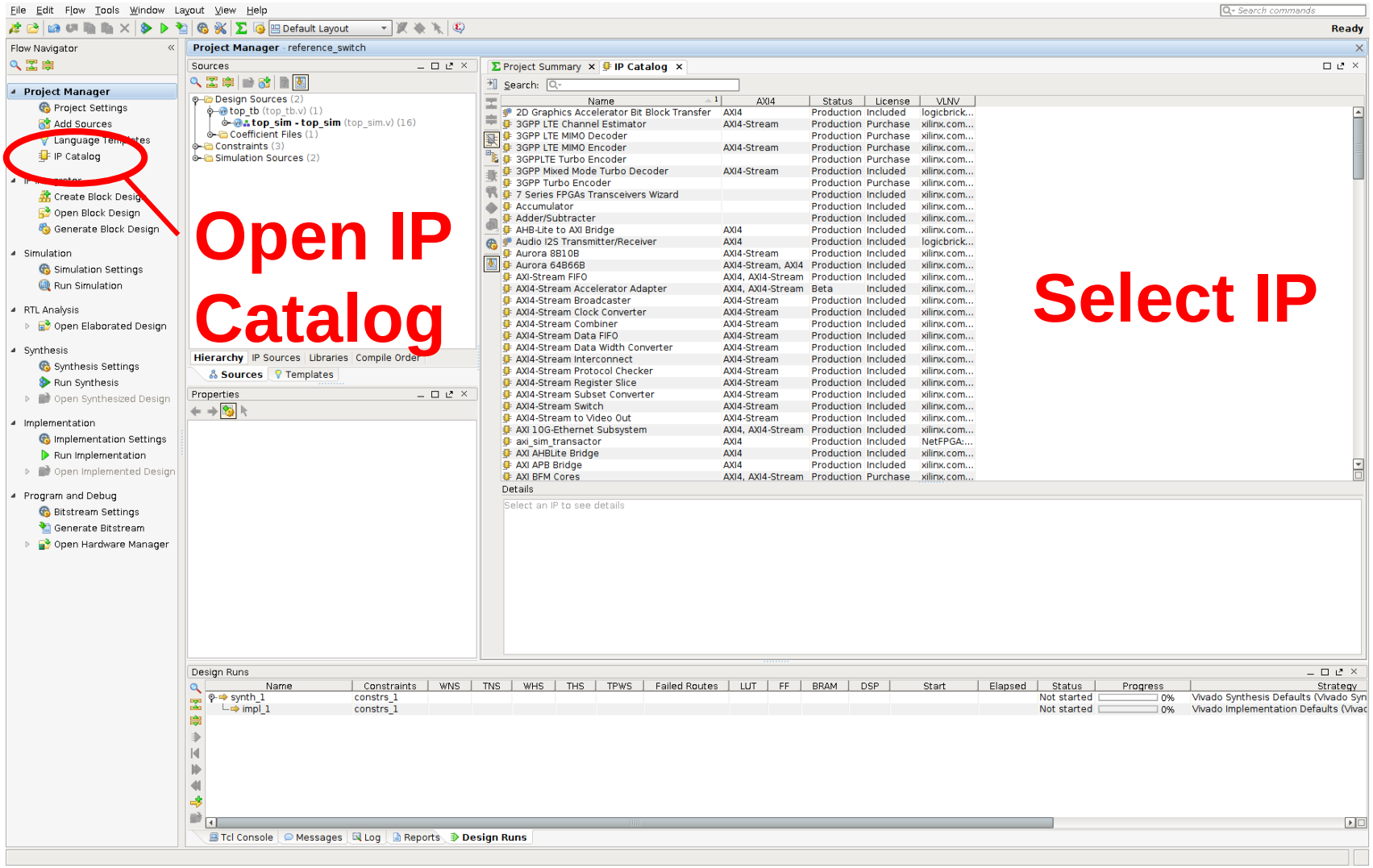
# <core_name>.tcl

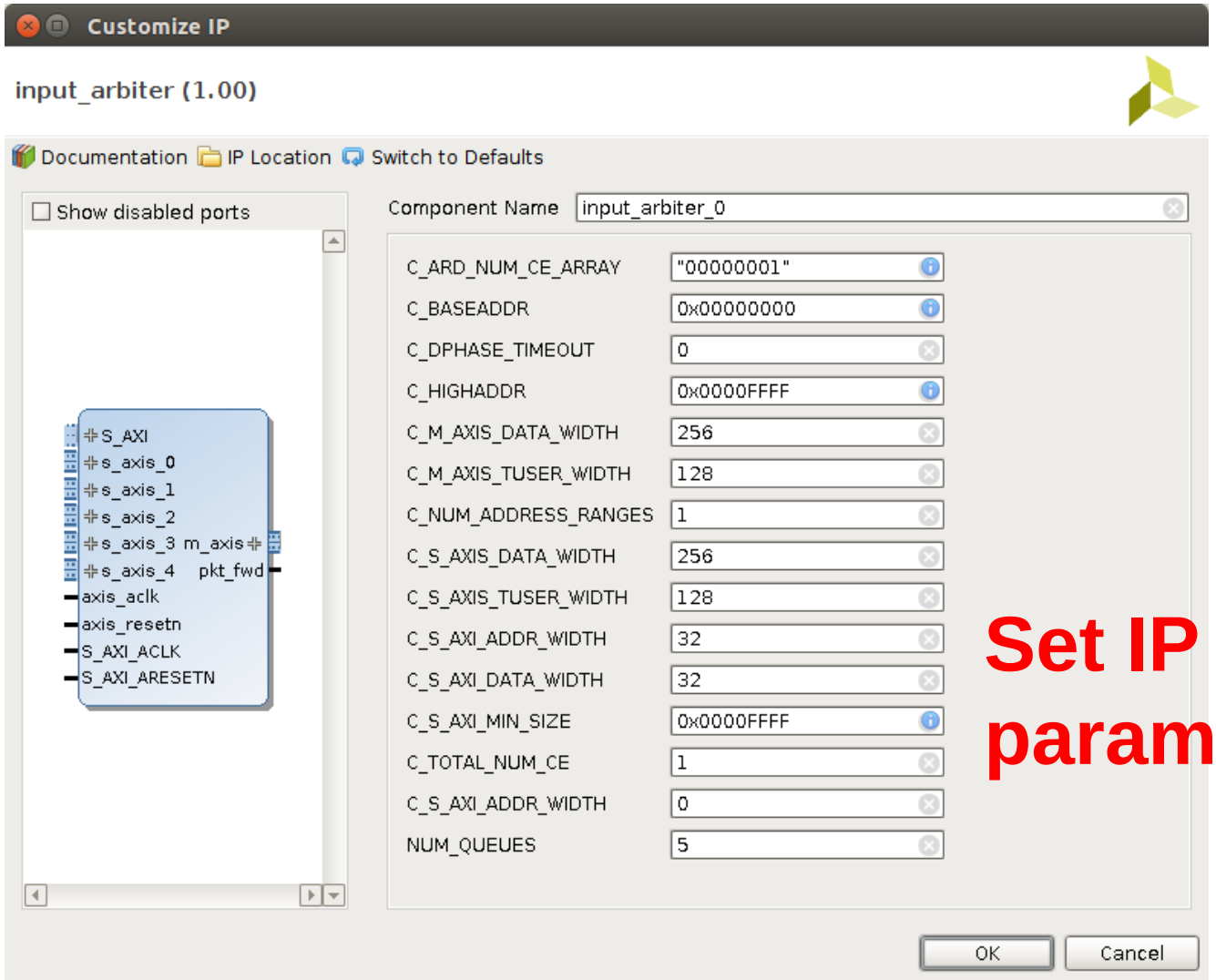
- **TCL file structure:**
  - Completing the project

```
ipx::save_core [ipx::current_core]
update_ip_catalog
close_project
```

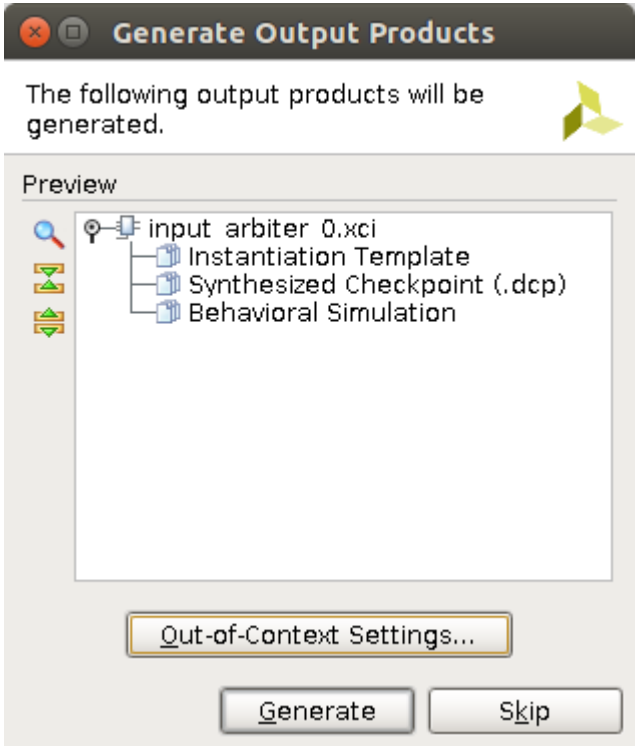
**Update the IP catalog to see the new core in the repo**

# Using an IP






**Set IP name and parameters values**




**Generate IP outputs (e.g. template, simulation)**

**Can take some time to generate**

# Add IP in TCL

- **From within a project:**

```
create_ip -name <core_name> -vendor <vendor_name>
          -library <lib_name> -module_name <ip_name>
set_property generate_synth_checkpoint false
          [get_files <ip_name>.xci]
reset_target all [get_ips <ip_name>]
```

**Example:**

```
create_ip -name output_port_lookup -vendor NetFPGA
          -library NetFPGA -module_name output_port_lookup_ip
set_property generate_synth_checkpoint false [get_files
          output_port_lookup_ip.xci]
reset_target all [get_ips output_port_lookup_ip]
```

# Using Subcores

- **What happens if you use an IP core within your core?**
- **How do you call it?**
- **How do you pass parameters to it?**
- **What happens if the same core is instantiated in multiple different cores, with different settings?**
  - An IP can be created only once (using the same name)
  - A *created IP* can have only a single set of values for its parameters

# Using Subcores

- **Solution: Subcores**
- **Indicate that an IP core instantiates other IP cores**
- **Can propagate parameters values in HDL**

```
ipx::add_subcore <vendor>: <library>:<name>: <version>
            [ipx::get_file_groups xilinx_verilogsynthesis -of_objects
            [ipx::current_core]]
ipx::add_subcore <vendor>: <library>:<name>: <version>
            [ipx::get_file_groups xilinx_verilogbehavioralsimulation –
             of_objects [ipx::current_core]]
```

**Example:**

```
ipx::add_subcore NetFPGA:NetFPGA:fallthrough_small_fifo:1.00
            [ipx::get_file_groups xilinx_verilogsynthesis -of_objects
            [ipx::current_core]]
```

# Compile

- **TCL only**
- **Run:**

vivado -mode batch -source <core_name>.tcl

# Do's and Don'ts

- **Don't create the same IP multiple times**
  - Save synthesis time!
- **Don't "create IP" within IPs**
- **Use add_subcores**
- **Make sure all parameters are available to the user**
- **Validate your design**
- **Provide useful information in your core identification**
- **Update core versions!**

# Conclusion

# Section IX: Conclusion

# Acknowledgments (I)


***NetFPGA Team at University of Cambridge (Past and Present):***

Andrew Moore, David Miller, Muhammad Shahbaz, Martin Zadnik, Matthew Grosvenor, Yury Audzevich, Neelakandan Manihatty-Bojan, Georgina Kalogeridou, Jong Hun Han, Noa Zilberman, Gianni Antichi, Charalampos Rotsos, Hwanju Kim, Marco Forconesi, Jinyun Zhang, Bjoern Zeeb, Robert Watson, Salvator Galea, Marcin Wojcik, Diana Andreea Popescu, Murali Ramanujam

***NetFPGA Team at Stanford University (Past and Present):***

Nick McKeown, Glen Gibb, Jad Naous, David Erickson, G. Adam Covington, John W. Lockwood, Jianying Luo, Brandon Heller, Paul Hartke, Neda Beheshti, Sara Bolouki, James Zeng, Jonathan Ellithorpe, Sachidanandan Sambandan, Eric Lo, Stephen Gabriel Ibanez

***All Community members (including but not limited to):***

Paul Rodman, Kumar Sanghvi, Wojciech A. Koszek, Yahsar Ganjali, Martin Labrecque, Jeff Shafer, Eric Keller, Tatsuya Yabe, Bilal Anwer, Yashar Ganjali, Martin Labrecque, Lisa Donatini, Sergio Lopez-Buedo , Andreas Fiessler, Robert Soule, Pietro Bressana, Yuta Tokusashi

Steve Wang, Erik Cengar, Michael Alexander, Sam Bobrowicz, Garrett Aufdemberg, Patrick Kane, Tom Weldon

Patrick Lysaght, Kees Vissers, Michaela Blott, Shep Siegel, Cathal McCabe

# Acknowledgements (II)