

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**

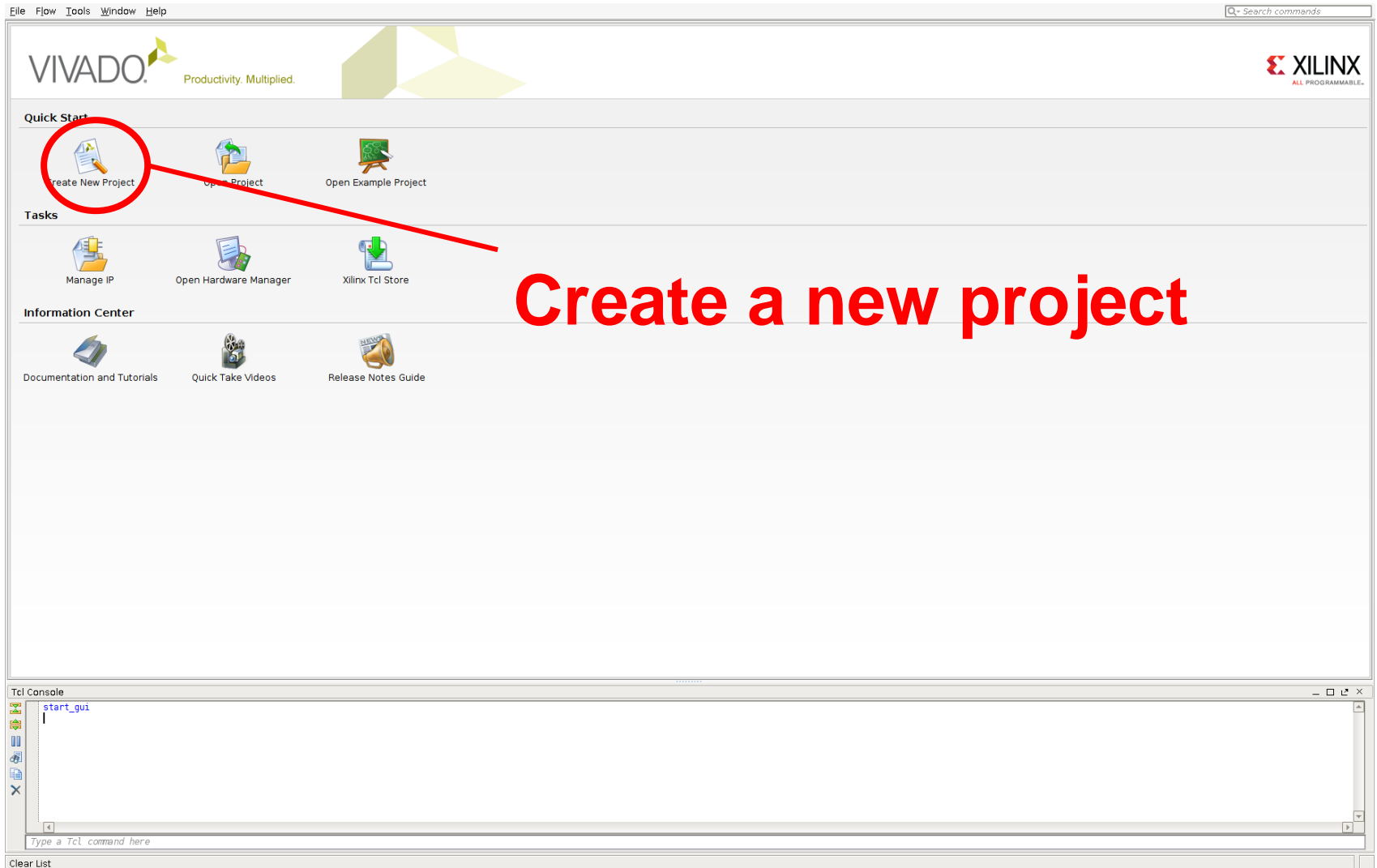
- **All done?**
 - Time to wrap your core

Packaging Cores

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

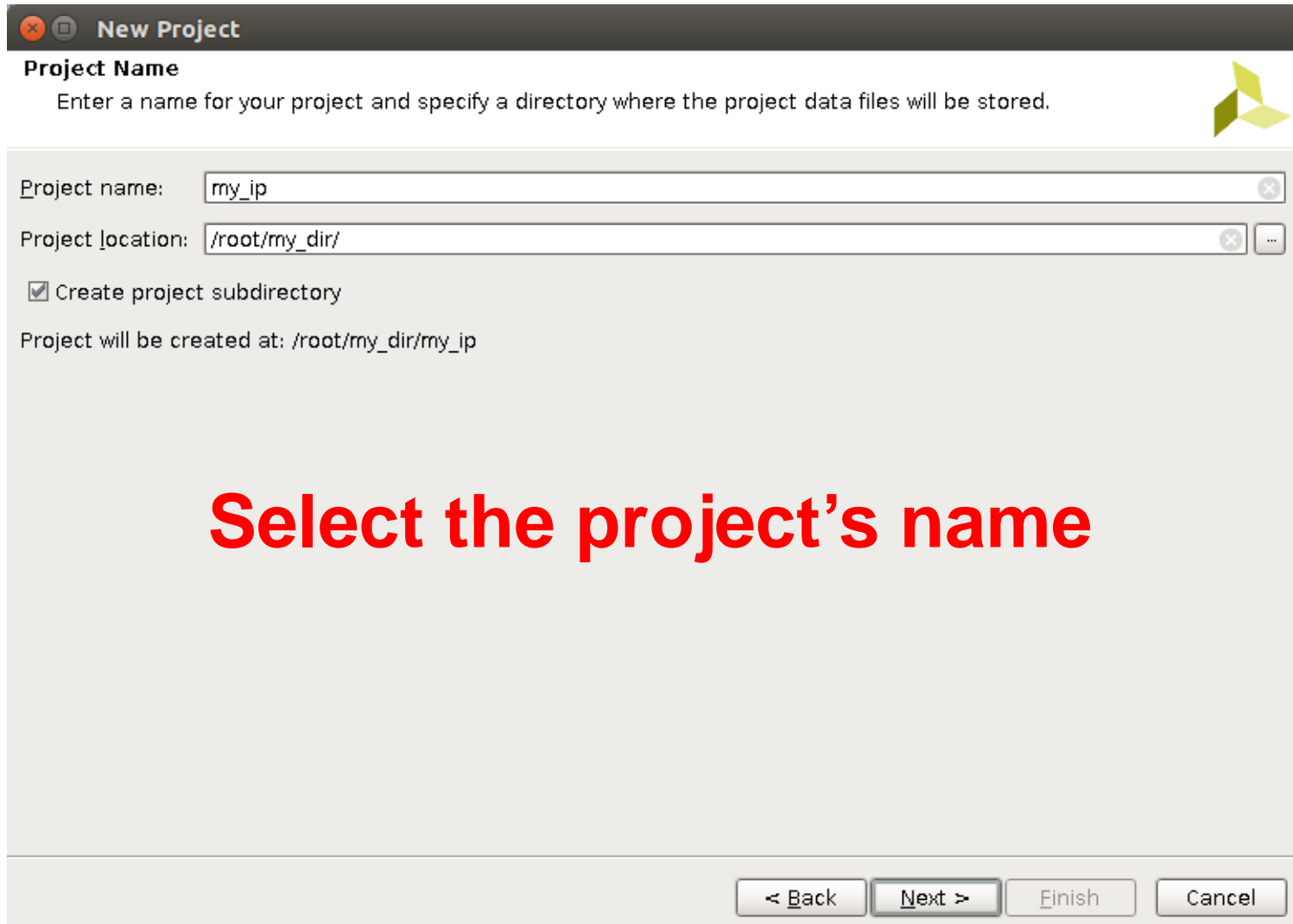
- **For best reuse across projects, we recommend using TCL scripts**
 - You can use the GUI and still export TCL
 - But they are not fully compatible

Packaging a Core using Vivado



Create a new project

Packaging a Core using Vivado (2)



New Project

Project Name
Enter a name for your project and specify a directory where the project data files will be stored.

Project name:

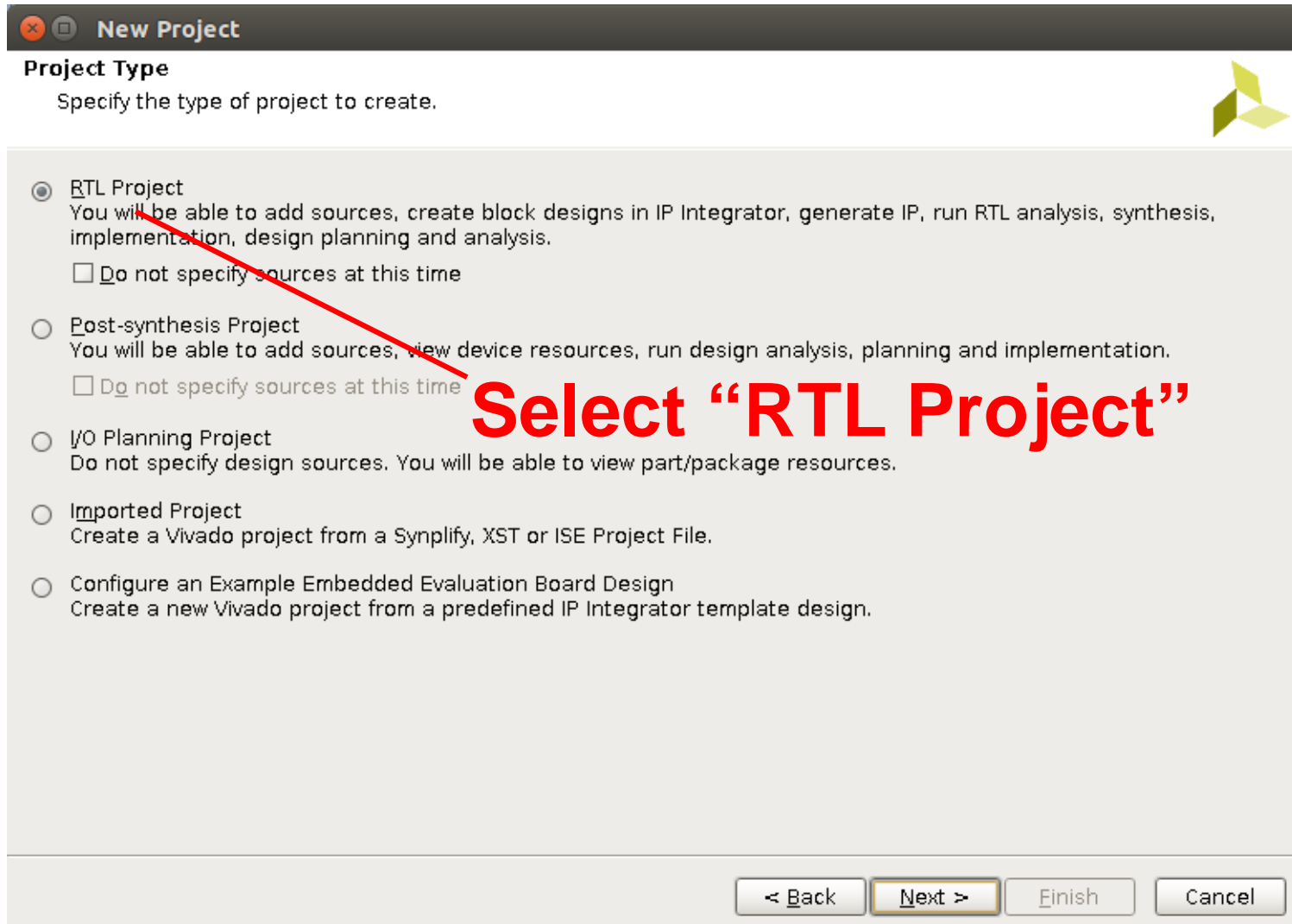
Project location:

Create project subdirectory

Project will be created at: /root/my_dir/my_ip

Select the project's name

Packaging a Core using Vivado (3)



New Project

Project Type
Specify the type of project to create.

- RTL Project**
You will be able to add sources, create block designs in IP Integrator, generate IP, run RTL analysis, synthesis, implementation, design planning and analysis.
 Do not specify sources at this time
- Post-synthesis Project**
You will be able to add sources, view device resources, run design analysis, planning and implementation.
 Do not specify sources at this time
- I/O Planning Project**
Do not specify design sources. You will be able to view part/package resources.
- Imported Project**
Create a Vivado project from a Synplify, XST or ISE Project File.
- Configure an Example Embedded Evaluation Board Design**
Create a new Vivado project from a predefined IP Integrator template design.

Select "RTL Project"

◀ Back **Next >** Finish Cancel

Packaging a Core using Vivado (4)

New Project

Add Sources

Specify HDL and netlist files, or directories containing HDL and netlist files, to add to your project. Create a new source file on disk and add it to your project. You can also add and create sources later.

Index	Name	Library	HDL Source For	Location
-------	------	---------	----------------	----------

Add HDL files

Add Files... **Add Directories...** **Create File...**

Scan and add RTL include files into project

Copy sources into project

Add sources from subdirectories

Target language: **Verilog** Simulator language: **Mixed**

Add Files
Add HDL and Netlist files to your project.

< Back **Next >** **Finish** **Cancel**

Packaging a Core using Vivado (5)

New Project

Default Part
Choose a default Xilinx part or board for your project. This can be changed later.

Select: Parts Boards

Filter

Product category: All Package: ffg1761
Family: Virtex-7 Speed grade: -3
Sub-Family: Virtex-7 Temp grade: All Remaining

Reset All Filters

Search:

Part	I/O Pin Count	Available IOBs	LUT Elements	FlipFlops	Block RAMs	DSPs	Gb Transceivers	GTXE2 Transceiv
xc7v585tffg1761-3	1,761	850	364200	728400	795	1260	36	36
xc7vx330tffg1761-3	1,761	700	204000	408000	750	1120	28	0
xc7vx485tffg1761-3	1,761	700	303600	607200	1030	2800	28	28
xc7vx690tffg1761-3	1,761	850	433200	866400	1470	3600	36	0

Select Device
XC7VX690TFFG1761-3

◀ Back Next ▶ Finish Cancel

Packaging a Core using Vivado (6)



New Project

New Project Summary

- ⓘ A new RTL project named 'my_ip' will be created.
- ⓘ 3 source files will be added.
- ⚠ No Configurable IP files will be added. Use Add Sources to add them later.
- ⚠ No constraints files will be added. Use Add Sources to add them later.
- ⓘ The default part and product family for the new project:
Default Part: xc7vx690tffg1761-3
Product: Virtex-7
Family: Virtex-7
Package: ffg1761
Speed Grade: -3

VIVADO

Summary

To create the project, click Finish

⏪ Back Next ⏩ **Finish** Cancel

Packaging a Core using Vivado (7)

The screenshot displays the Vivado IDE interface for a project named 'my_ip'. The Project Manager on the left shows a hierarchy of sources, with 'input_arbiter (input_arbiter.v)' highlighted and a red box labeled 'Source Files' overlaid on it. The Project Summary panel on the right provides details for the project settings, synthesis, implementation, DRC violations, utilization, timing, and power. A red box labeled 'Project Summary' is overlaid on this panel. The Design Runs table at the bottom shows the status of synthesis and implementation runs.

Name	Constraints	WNS	TNS	WHS	THS	TPWS	Failed Routes	LUT	FF	BRAM	DSP	Start	Elapsed	Status	Progress	Strategy
synth_1	constrs_1											Not started		Not started	0%	Vivado Synthesis Defaults (Vivado Syn
impl_1	constrs_1											Not started		Not started	0%	Vivado Implementation Defaults (Vivac

Packaging a Core using Vivado (8)



Create and Package New IP

Create and Package IP

This wizard can be used to accomplish two tasks:

Package a new IP for the Vivado IP Catalog
This wizard will guide you through the process of creating a new Vivado IP using source files and information from your current project or specified directory.

Create a new AXI4 Peripheral
This wizard will guide you through the process of creating a new AXI4 peripheral which includes HDL, driver, software test application, IP Integrator BFM simulation and debug demonstration design.

Tools → Create and package IP

VIVADO.

Click Next to continue

< Back Next > Finish Cancel

Packaging a Core using Vivado (9)

Create and Package New IP

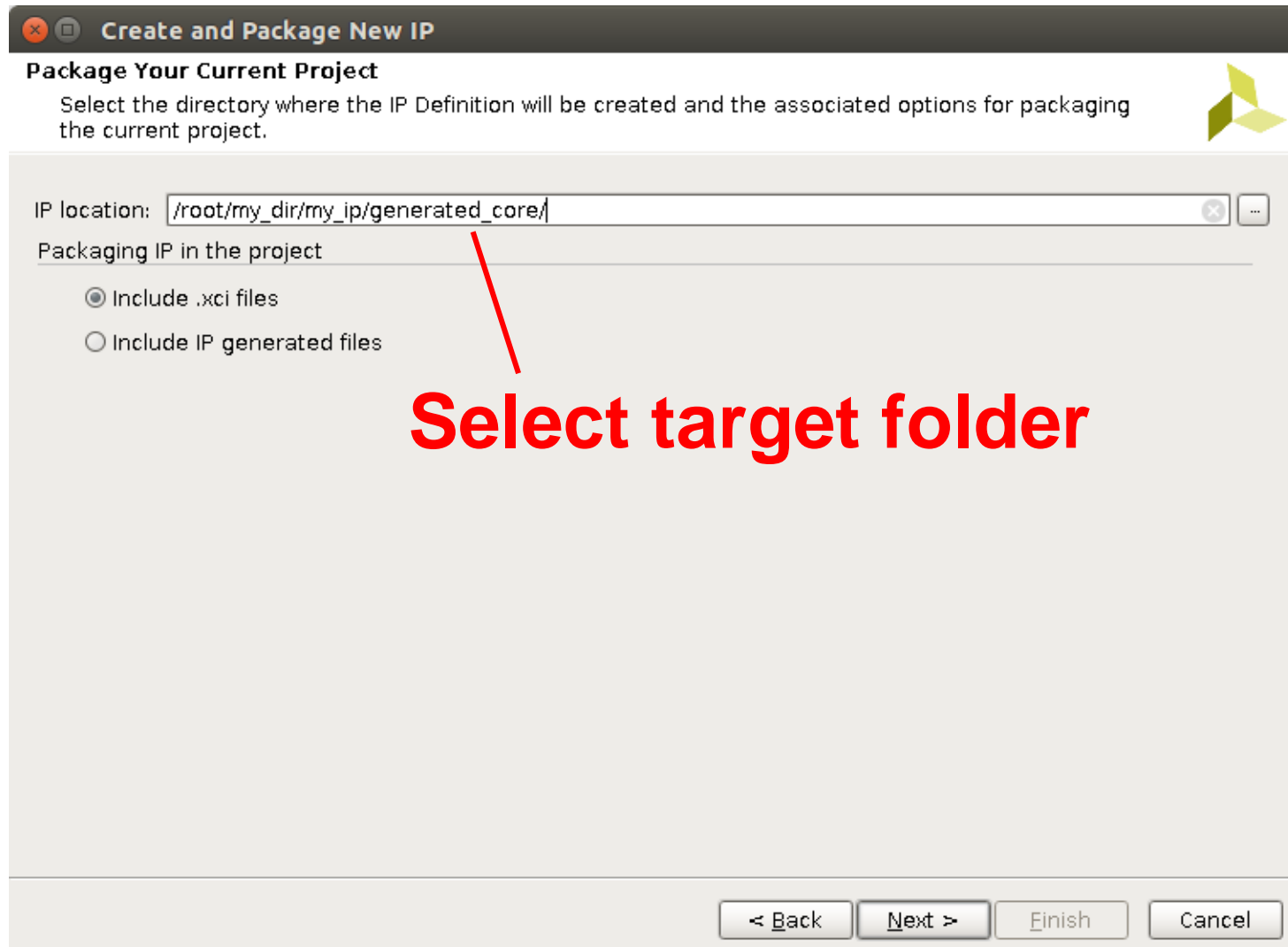
Choose Create Peripheral or Package IP
Please select one of the following tasks.

- Package your current project**
Use the project as the source for creating a new IP Definition.
Note: All sources to be packaged must be located at or below the specified directory.
- Package a specified directory**
Choose a directory as the source for creating a new IP Definition.
- Create a new AXI4 peripheral**
Create an AXI4 IP, driver, software test application, IP Integrator AXI4 BFM simulation and debug demonstration design.

Package your current project

< Back Next > Finish Cancel

Packaging a Core using Vivado (10)



Create and Package New IP

Package Your Current Project

Select the directory where the IP Definition will be created and the associated options for packaging the current project.

IP location:

Packaging IP in the project

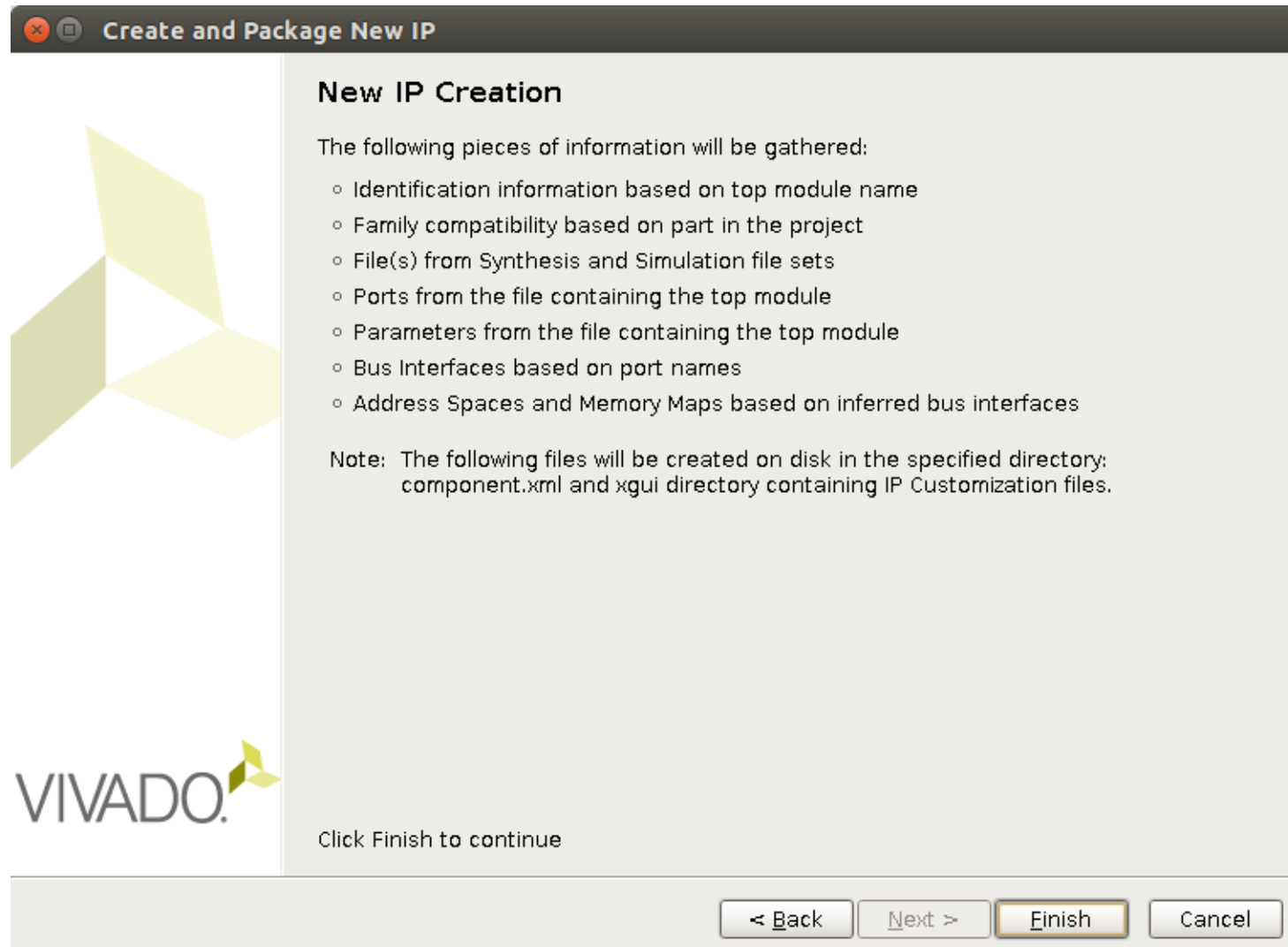
Include .xci files

Include IP generated files

Select target folder

< Back Next > Finish Cancel

Packaging a Core using Vivado (11)



Packaging a Core using Vivado (12)

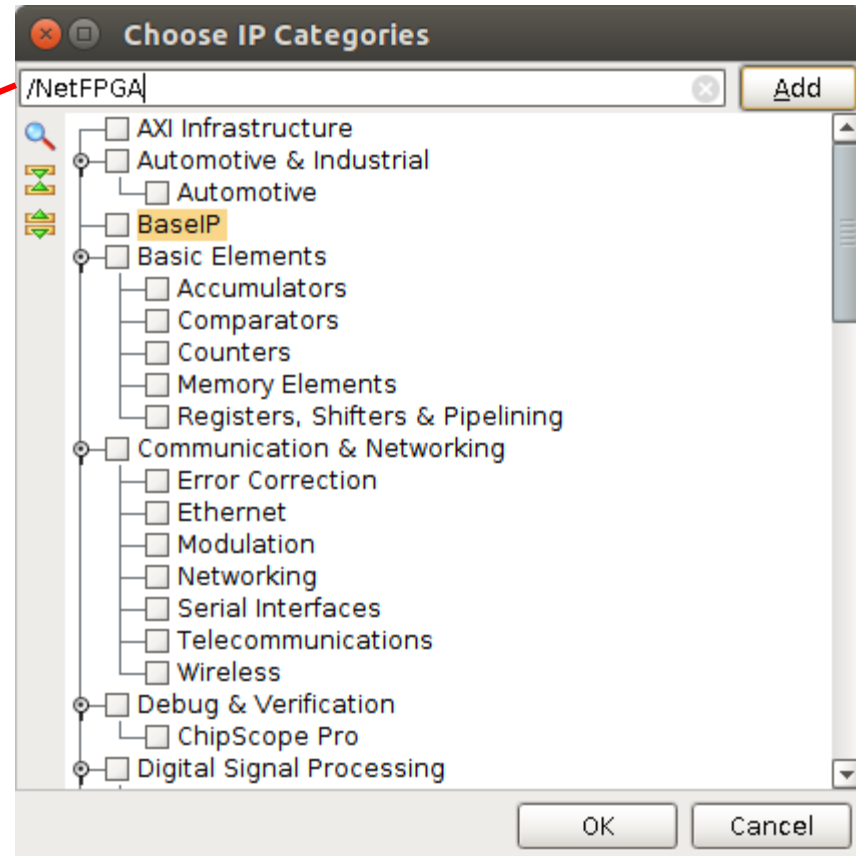
Change category

Edit core's identification fields

Name	Constraints	WNS	TNS	WHS	THS	TPWS	Failed Routes	LUT	FF	BRAM	DSP	Start	Elapsed	Status	Progress	Strategy
synth_1	constrs_1													Not started	0%	Vivado Synthesis Defaults (Vivado Syn
impl_1	constrs_1													Not started	0%	Vivado Implementation Defaults (Vivac

Packaging a Core using Vivado (13)

**Select category
or create a new
one**



Packaging a Core using Vivado (14)

The screenshot shows the Vivado IDE interface during the packaging of an IP core. The 'Project Manager' window displays the project structure, including 'input_arbiter (input_arbiter.v)'. The 'Packaging Steps' window shows the progress of various steps, with 'File Groups' currently selected and highlighted in yellow. A red arrow points from the 'File Groups' step to the 'File Groups' table. The table has columns for Name, Library Name, Type, and Is Includ. The 'File Groups' table is currently empty.

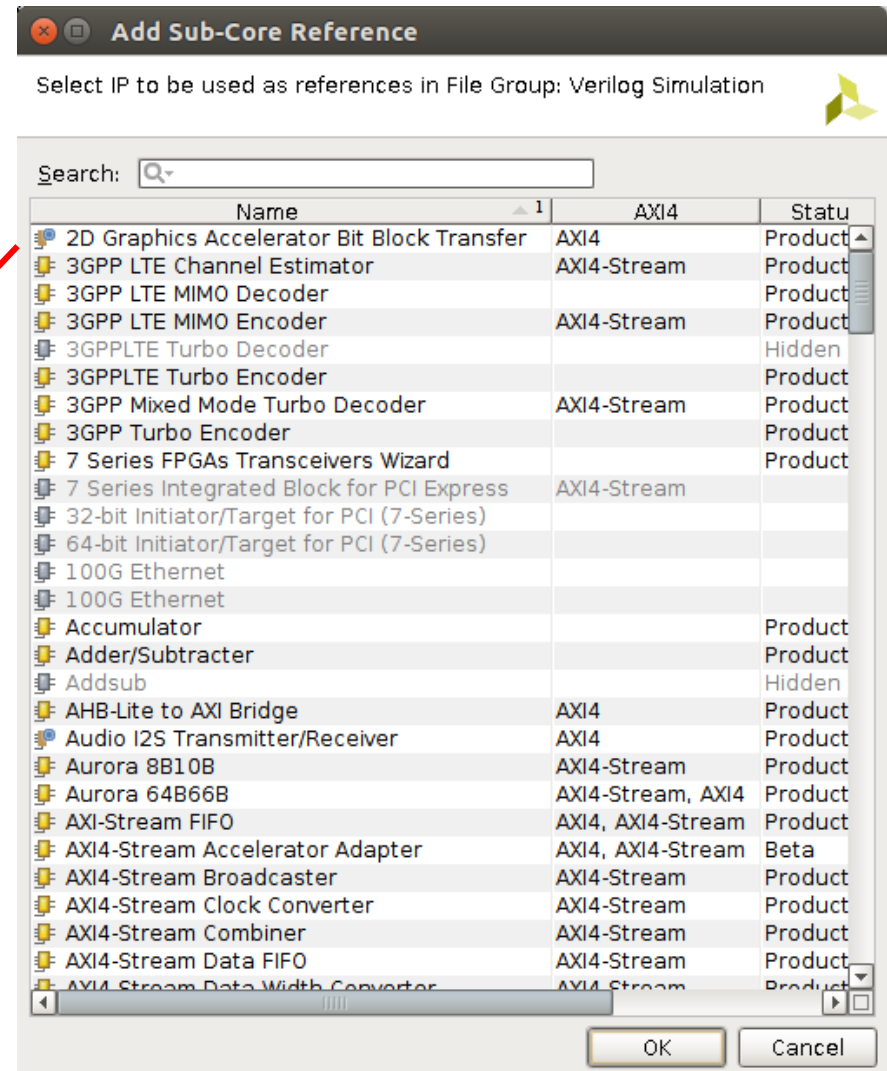
Define File Groups and add subcores

Name	Constraints	WNS	TNS	WHS	THS	TPWS	Failed Routes	LUT	FF	BRAM	DSP	Start	Elapsed	Status	Progress	Strategy
synth_1	constrs_1													Not started	0%	Vivado Synthesis Defaults (Vivado Syn
impl_1	constrs_1													Not started	0%	Vivado Implementation Defaults (Vivac

Packaging a Core using Vivado (15)

Select subcore

A subcore is an IP instantiated within the core



Packaging a Core using Vivado (16)

The screenshot shows the Vivado IDE interface with the 'Customization Parameters' window open. The window displays a table of parameters for the 'input_arbiter' IP core. The parameters are listed in the following table:

Name	Description	Display Name	Value	Value Bit String Length	Value Format	Value Source
Customization Parameters						
C_M_AXIS_DATA_WIDTH	C M Axis Data Width	C M Axis Data Width	256	0	long	
C_S_AXIS_DATA_WIDTH	C S Axis Data Width	C S Axis Data Width	256	0	long	
C_M_AXIS_TUSER_WIDTH	C M Axis Tuser Width	C M Axis Tuser Width	128	0	long	
C_S_AXIS_TUSER_WIDTH	C S Axis Tuser Width	C S Axis Tuser Width	128	0	long	
NUM_QUEUES	Num Queues	Num Queues	5	0	long	
C_S_AXI_DATA_WIDTH	C S Axii Data Width	C S Axii Data Width	32	0	long	
C_S_AXI_ADDR_WIDTH	C S Axii Addr Width	C S Axii Addr Width	12	0	long	
C_USE_WSTRB	C Use Wstrb	C Use Wstrb	0	0	long	
C_DPHASE_TIMEOUT	C Dphase Timeout	C Dphase Timeout	0	0	long	
C_NUM_ADDRESS_RANGES	C Num Address Ranges	C Num Address Ranges	1	0	long	
C_TOTAL_NUM_CE	C Total Num Ce	C Total Num Ce	1	0	long	
C_S_AXI_MIN_SIZE	C S Axii Min Size	C S Axii Min Size	0x0000FFFF	32	bitString	
C_ARC_NUM_CE_ARRAY	C Arc Num Ce Array	C Arc Num Ce Array	"00000001"	8	bitString	
C_BASEADDR	C Baseaddr	C Baseaddr	0x00000000	32	bitString	
C_HIGHADDR	C Highaddr	C Highaddr	0x0000FFFF	32	bitString	
Hidden						

A large red text overlay is positioned over the right side of the screenshot, reading: **Update parameters and set default values**.

Packaging a Core using Vivado (17)

Update ports

Note related warnings

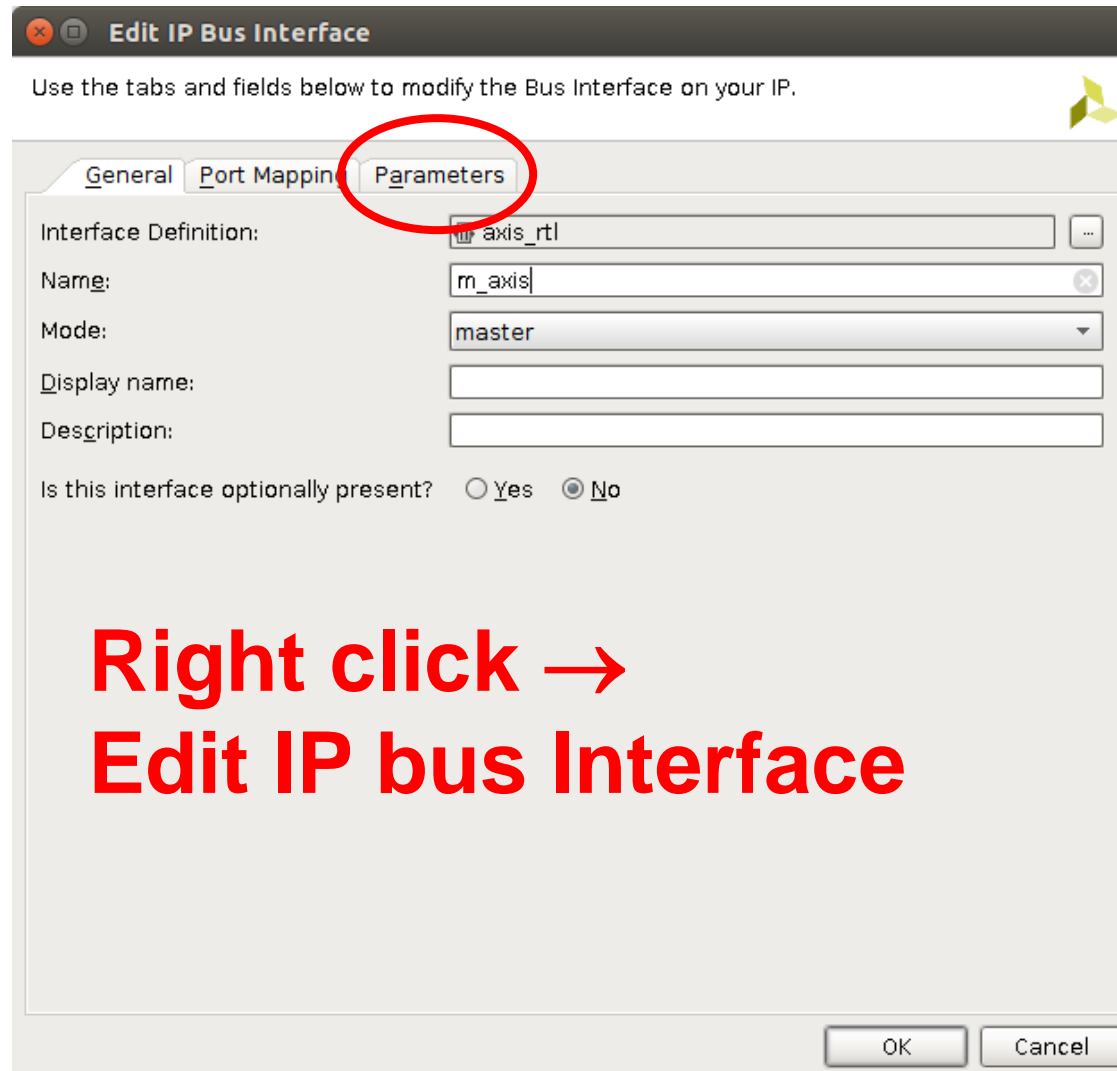
Name	Interface Mode	Enableness	Depend...	Is Declaration	Direction	Driver Value	Size Left	Size Left
S_AXI	slave			<input type="checkbox"/>				
m_axis	master			<input type="checkbox"/>				
s_axis_0	slave			<input type="checkbox"/>				
s_axis_1	slave			<input type="checkbox"/>				
s_axis_2	slave			<input type="checkbox"/>				
s_axis_3	slave			<input type="checkbox"/>				
s_axis_4	slave			<input type="checkbox"/>				
Clock and Reset Signals								
S_AXI_signal_reset	slave			<input type="checkbox"/>				
axi_signal_reset	slave			<input type="checkbox"/>				
S_AXI_signal_clock	slave			<input type="checkbox"/>				
axi_signal_clock	slave			<input type="checkbox"/>				
pkt_fwd				<input type="checkbox"/>	out			

Messages
6 warnings (26 infos) Show All

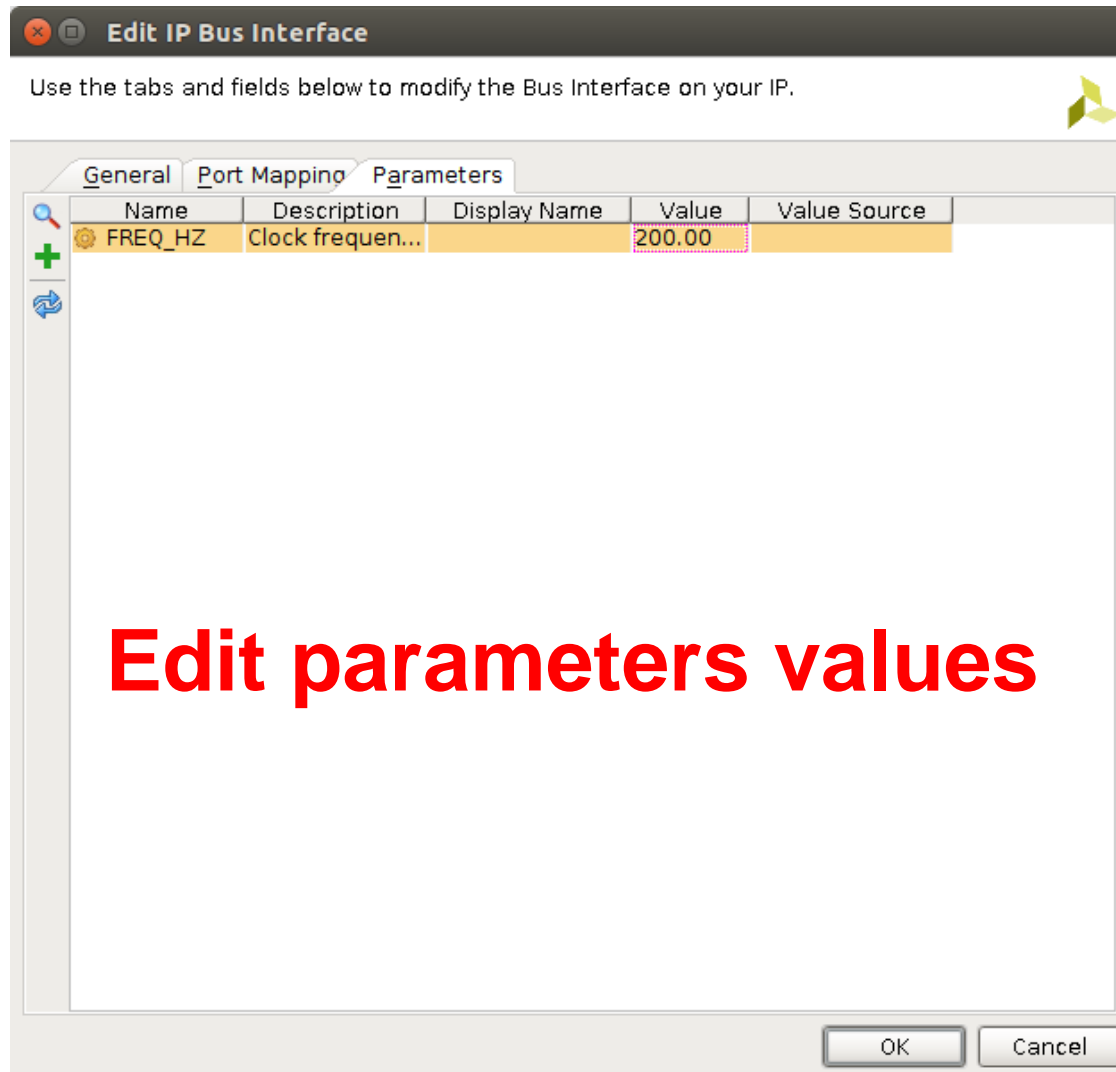
IP Packager (6 warnings)

- Ports and Interfaces Wizard (6 warnings)
- [IP_Flow 19-3158] Bus Interface 'm_axis': FREQ_HZ bus parameter missing from AXI interface when interface is not associated to a clock.
- [IP_Flow 19-3158] Bus Interface 's_axis_0': FREQ_HZ bus parameter missing from AXI interface when interface is not associated to a clock.
- [IP_Flow 19-3158] Bus Interface 's_axis_1': FREQ_HZ bus parameter missing from AXI interface when interface is not associated to a clock.
- [IP_Flow 19-3158] Bus Interface 's_axis_2': FREQ_HZ bus parameter missing from AXI interface when interface is not associated to a clock.
- [IP_Flow 19-3158] Bus Interface 's_axis_3': FREQ_HZ bus parameter missing from AXI interface when interface is not associated to a clock.
- [IP_Flow 19-3158] Bus Interface 's_axis_4': FREQ_HZ bus parameter missing from AXI interface when interface is not associated to a clock.

Packaging a Core using Vivado (18)



Packaging a Core using Vivado (19)



Packaging a Core using Vivado (20)

Updating the memory map is possible (Not required in most NetFPGA cores)

Name	Display Name	Description	Base Address	Range	Range Dependency
reg0			0	4096	pow(2,(C_S_AXI_ADDR_WIDTH - 1) + 1)

Messages: 5 warnings, 52 Infos. Show All

- IP Packager (5 warnings)
- Ports and Interfaces Wizard (5 warnings)
- [IP_Flow 19-3158] Bus Interface 's_axis_0': FREQUENCY_HZ bus parameter missing from AXI interface when interface is not associated to a clock.
- [IP_Flow 19-3158] Bus Interface 's_axis_1': FREQUENCY_HZ bus parameter missing from AXI interface when interface is not associated to a clock.
- [IP_Flow 19-3158] Bus Interface 's_axis_2': FREQUENCY_HZ bus parameter missing from AXI interface when interface is not associated to a clock.
- [IP_Flow 19-3158] Bus Interface 's_axis_3': FREQUENCY_HZ bus parameter missing from AXI interface when interface is not associated to a clock.
- [IP_Flow 19-3158] Bus Interface 's_axis_4': FREQUENCY_HZ bus parameter missing from AXI interface when interface is not associated to a clock.

Packaging a Core using Vivado (21)

The screenshot displays the Vivado IP Integrator interface for packaging an IP core. The main window is titled "Package IP - input_arbiter" and shows the "Customization GUI" for the "input_arbiter" component. The interface is divided into several panes:

- Project Manager:** Shows the project structure with sources like Verilog, IP-XACT, and Simulation Sources.
- Packaging Steps:** A checklist of steps including Identification, Compatibility, File Groups, Customization Parameters, Ports and interfaces, Addressing and Memory, and Customization GUI (which is currently selected).
- Hierarchy:** Shows the component hierarchy with "input_arbiter" as the main component.
- IP Bus Interface Properties:** A panel for configuring the bus interface, including fields for Monitor group, Slave bridges, Connection required, Endianness, Enablement dependency, Enablement presence, Enablement resolve type, Enablement value, Is declaration, and Class (set to "bus_interface").
- Customization GUI:** The main configuration area, split into "Layout" and "Preview". The "Layout" pane shows a tree view of the component's parameters. The "Preview" pane shows a block diagram of the component and a table of configuration parameters.
- Messages:** A panel at the bottom showing warnings from the IP Packager, specifically related to missing FREQ_HZ bus parameters for the AXI interfaces.

The "Preview" pane contains the following configuration parameters for the "input_arbiter_0" component:

Parameter	Value
C Ard Num Ce Array	"00000001"
C Baseaddr	0x00000000
C Dphase Timeout	0
C Highaddr	0x0000FFFF
C M Axis Data Width	256
C M Axis Tuser Width	128
C Num Address Ranges	1
C S Axis Data Width	256
C S Axis Tuser Width	128
C S Axi Addr Width	12
C S Axi Data Width	32
C S Axi Min Size	0x0000FFFF
C Total Num Ce	1
C Use Wstrb	0
Num Queues	5

A large red text overlay "Customize GUI" is positioned over the bottom right of the customization GUI pane.

Packaging a Core using Vivado (22)

File Edit Flow Tools Window Layout View Help

Flow Navigator

Project Manager tmp_edit_project

Sources

- Design Sources (3)
 - Verilog (1)
 - input_arbiter (input_arbiter.v) (2)
 - IP-XACT (1)
- Constraints
- Simulation Sources (2)

Hierarchy Libraries Compile Order

Sources Templates

IP Bus Interface Properties

s_axis_4

Monitor group: []

Slave bridges: []

Connection required:

Endianness: little

Enablement dependency: []

Enablement presence: optional

Enablement resolve type: immediate

Enablement value:

Is declaration:

Class: bus_interface

Packaging Steps

- Identification
- Compatibility
- File Groups
- Customization Parameters
- Ports and interfaces
- Addressing and Memory
- Customization GUI
- Review and Package**

Review and Package

Summary of your IP

IP display name: input_arbiter_v1_0
IP description: input_arbiter_v1_0
IP root directory: /root/my_dir/my_ip/generated_core

After Packaging

- An archive will not be generated. Use the settings link below to change your preference
- Project will be removed after completion

[edit packaging settings](#)

Package IP

Messages

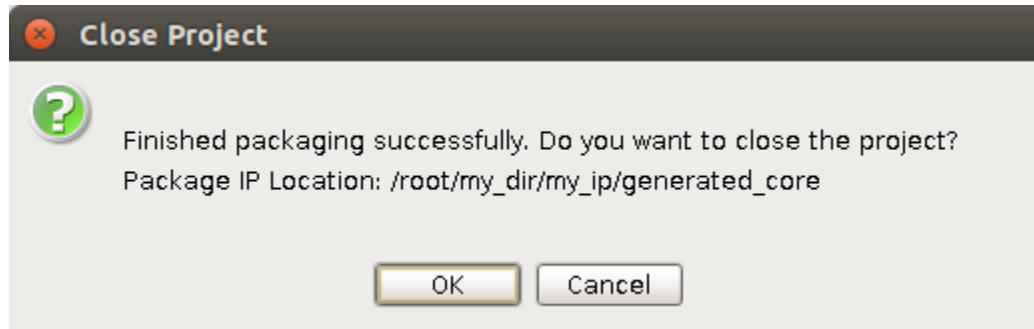
195 infos Show All

Tcl Console Messages Log Reports Design Runs

IP Bus Interface: s_axis_4

Core Summary
Any errors? Warnings?

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

Open IP Catalog

Select IP

Name	AXI4	Status	License	VLNV
2D Graphics Accelerator Bit Block Transfer	AXI4	Production	Included	logicbrick...
3GPP LTE Channel Estimator	AXI4-Stream	Production	Purchase	xilinx.com...
3GPP LTE MIMO Decoder	AXI4-Stream	Production	Purchase	xilinx.com...
3GPP LTE MIMO Encoder	AXI4-Stream	Production	Purchase	xilinx.com...
3GPP LTE Turbo Encoder	AXI4-Stream	Production	Purchase	xilinx.com...
3GPP LTE Turbo Decoder	AXI4-Stream	Production	Included	xilinx.com...
3GPP Turbo Encoder	AXI4-Stream	Production	Purchase	xilinx.com...
7 Series FPGAs Transceivers Wizard		Production	Included	xilinx.com...
Accumulator		Production	Included	xilinx.com...
Adder/Subtractor		Production	Included	xilinx.com...
AHB-Lite to AXI Bridge	AXI4	Production	Included	xilinx.com...
Audio I2S Transmitter/Receiver	AXI4	Production	Included	logicbrick...
Aurora 8B10B	AXI4-Stream	Production	Included	xilinx.com...
Aurora 64B66B	AXI4-Stream, AXI4	Production	Included	xilinx.com...
AXI-Stream FIFO	AXI4, AXI4-Stream	Production	Included	xilinx.com...
AXI4-Stream Accelerator Adapter	AXI4, AXI4-Stream	Beta	Included	xilinx.com...
AXI4-Stream Broadcaster	AXI4-Stream	Production	Included	xilinx.com...
AXI4-Stream Clock Converter	AXI4-Stream	Production	Included	xilinx.com...
AXI4-Stream Combiner	AXI4-Stream	Production	Included	xilinx.com...
AXI4-Stream Data FIFO	AXI4-Stream	Production	Included	xilinx.com...
AXI4-Stream Data Width Converter	AXI4-Stream	Production	Included	xilinx.com...
AXI4-Stream Interconnect	AXI4-Stream	Production	Included	xilinx.com...
AXI4-Stream Protocol Checker	AXI4-Stream	Production	Included	xilinx.com...
AXI4-Stream Register Slice	AXI4-Stream	Production	Included	xilinx.com...
AXI4-Stream Subset Converter	AXI4-Stream	Production	Included	xilinx.com...
AXI4-Stream Switch	AXI4-Stream	Production	Included	xilinx.com...
AXI4-Stream to Video Out	AXI4-Stream	Production	Included	xilinx.com...
AXI 10G-Ethernet Subsystem	AXI4, AXI4-Stream	Production	Included	xilinx.com...
axi_sim_transactor	AXI4	Production	Included	NetFPGA...
AXI AHB-Lite Bridge	AXI4	Production	Included	xilinx.com...
AXI APB Bridge	AXI4	Production	Included	xilinx.com...
AXI BFM Cores	AXI4, AXI4-Stream	Production	Purchase	xilinx.com...

Name	Constraints	WNS	TNS	WHS	THS	TPWS	Failed Routes	LUT	FF	BRAM	DSP	Start	Elapsed	Status	Progress	Strategy
synth_1	constrs_1													Not started	0%	Vivado Synthesis Defaults (Vivado Syn
impl_1	constrs_1													Not started	0%	Vivado Implementation Defaults (Vivac

Customize IP

input_arbiter (1.00)

Documentation IP Location Switch to Defaults

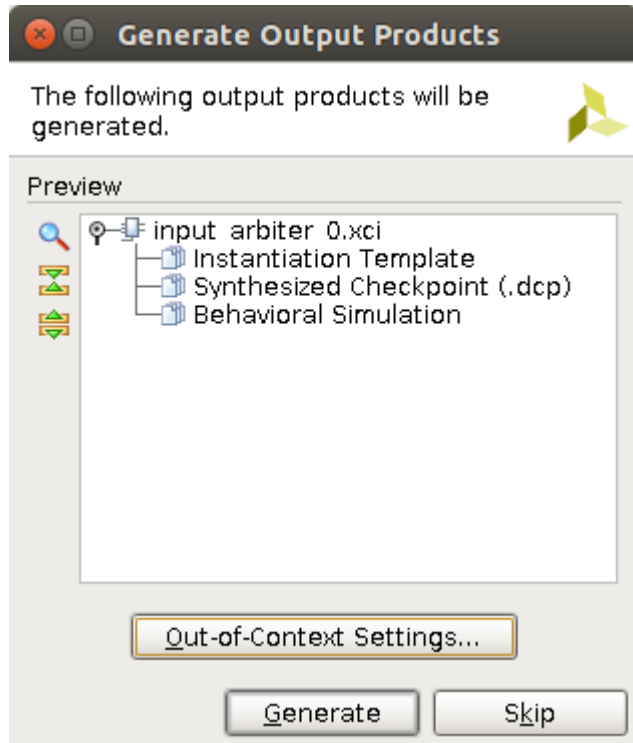
Show disabled ports

Component Name: input_arbiter_0

C_ARD_NUM_CE_ARRAY	"00000001"
C_BASEADDR	0x00000000
C_DPHASE_TIMEOUT	0
C_HIGHADDR	0x0000FFFF
C_M_AXIS_DATA_WIDTH	256
C_M_AXIS_TUSER_WIDTH	128
C_NUM_ADDRESS_RANGES	1
C_S_AXIS_DATA_WIDTH	256
C_S_AXIS_TUSER_WIDTH	128
C_S_AXI_ADDR_WIDTH	32
C_S_AXI_DATA_WIDTH	32
C_S_AXI_MIN_SIZE	0x0000FFFF
C_TOTAL_NUM_CE	1
C_S_AXI_ADDR_WIDTH	0
NUM_QUEUES	5

OK Cancel

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)



UNIVERSITY OF
CAMBRIDGE

EPSRC

Pioneering research
and skills



The Leverhulme Trust

