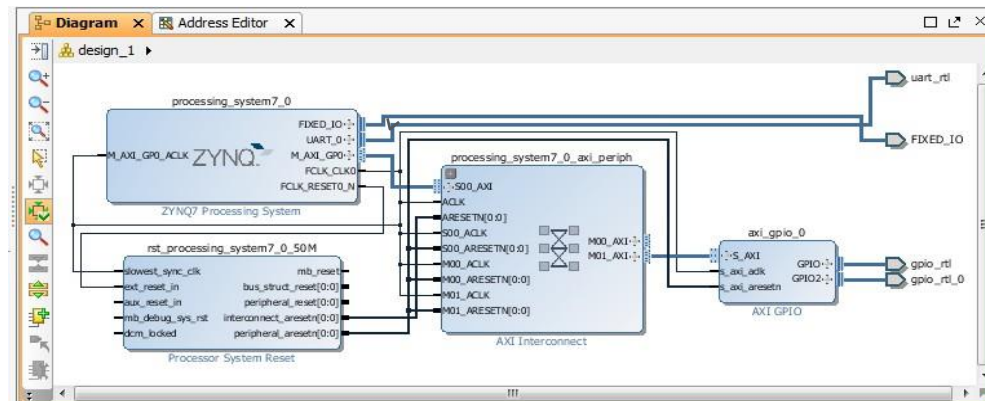


# Johns Hopkins Engineering for Professionals

## System-on-Chip FPGA Design Lab Introduction to Custom IP



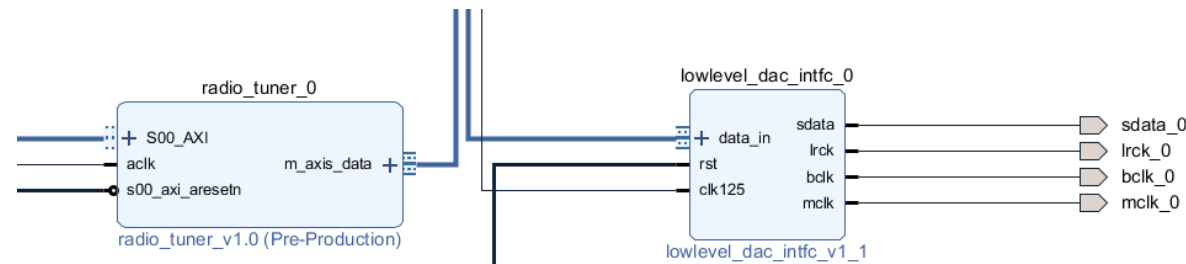






## IP Packaging

- Effective Design Re-Use Requires
  - Standard Interfaces (think AXI / AXI Stream) as much as possible
  - Well designed IP
    - Customizable for multiple uses
    - Bulletproof
  - Goal : a single file or directory that we can put in a catalog and people can use it without having to know anything about how it is constructed. We want a block that “just works”
  - With a catalog full of these, designs can be constructed and iterated rapidly
- Xilinx and other manufacturers come with a large suite of existing IP in the default catalog, but allow user defined IP “repositories” as a place to store IP purchased from other vendors, and IP built by you
- A project can be pointed to use an IP repository, and all of the IP which is stored in that repository will become accessible as part of your IP catalog window (which we are familiar with already)



*\*\* note – packaging up a design into a piece of IP is likely something you will do after you test a bit. It adds a few steps, so it isn't as agile of a process*



## IP “Repository”

- IP Repository is just a directory which contains a bunch of IP
  - A directory folder for each packaged IP
  - An IP repo can have a user-defined structure. The tools will browse through the entire tree of the repo looking for valid “packaged” IP
- Each IP in the repository when packaged is labelled by the person packing the IP with a few parameters which make up a unique identifier for the IP : “VLNV”
  - Vendor (e.g. “Johns Hopkins”)
  - Library (e.g. “DougBlocks”)
  - Name (e.g. “lowlevel\_dac\_interface”)
  - Version (e.g. v1.0)
- A project can specify multiple IP Repositories and then the user of that project will have a catalog which contains the union of all of the IP in each repo
  - Only duplicate VLNv will possible cause problems, as long as the IP directory itself is kept together, the location is irrelevant
  - Imagine – a single directory with all sort of nice building blocks that you’ve made that can be used across projects. Any project can be directed to the “repo” with a single click, and all those building blocks show up in the Vivado IP catalog.



## Example IP Repository

ad9361_to_axis	12/9/2019 2:51 PM	File folder
axi_ad9361	12/9/2019 2:51 PM	File folder
axi_polled_if	12/9/2019 2:51 PM	File folder
clock_crosser	12/9/2019 2:51 PM	File folder
dma_packetizer	12/9/2019 2:51 PM	File folder
gps_1pps_sampler	12/9/2019 2:51 PM	File folder
gps_handler	12/9/2019 2:51 PM	File folder
msix_intc	12/9/2019 2:51 PM	File folder
system_info_1_7	12/9/2019 2:51 PM	File folder
system_info_block	12/9/2019 2:51 PM	File folder
timestamp_engine	12/9/2019 2:51 PM	File folder
tx_engine	12/9/2019 2:51 PM	File folder
tx_engine_tp8	12/9/2019 2:51 PM	File folder

Directory (or zip) for each IP Core



All the source files for that core

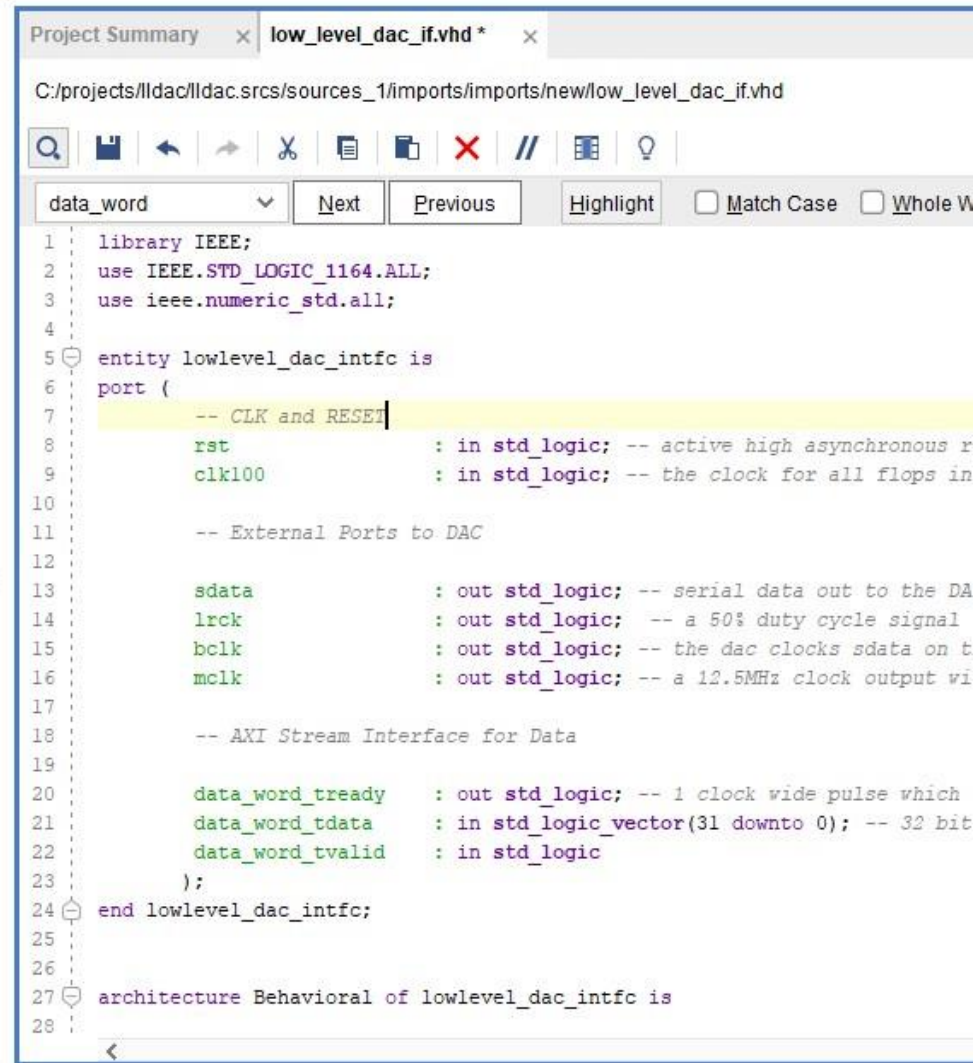
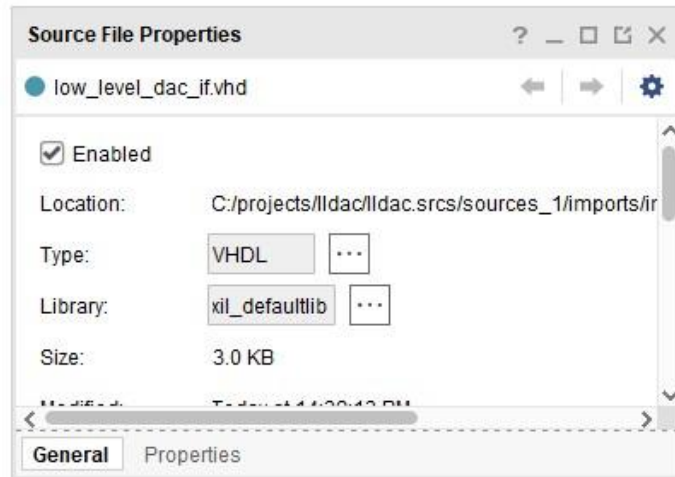
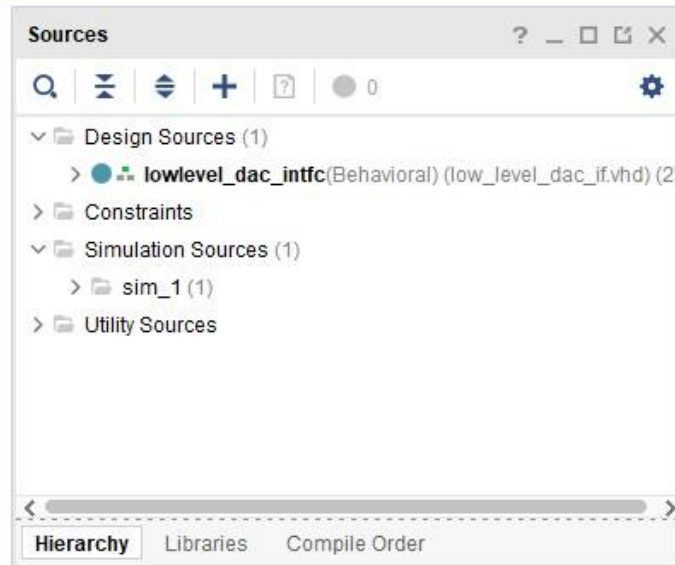
src	12/9/2019 2:51 PM	File folder	
xgui	12/9/2019 2:51 PM	File folder	
component	12/9/2019 2:51 PM	XML Document	21 KB

information about the IP, what the ports are, the VLNV...etc. This is what makes an IP an IP, and not just a bunch of files

Designs get turned into IP (in this format) through a process called “Packaging”. The next few slides will show that process for our lowlevel\_dac\_interface.







- Start with a standard Vivado project
- Note that we've changed a few port names. This isn't required, but I want to package the dac\_interface as an AXI stream component, so best practice is to name the signals with the convention. (It will also allow the tool to autodetect the AXI-S interconnect)
- Select Tools->Create and Package New IP


## Create and Package New IP

Choose this now




We will do this later!



 Create and Package New IP ✕

**Create Peripheral, Package IP or Package a Block Design**  
Please select one of the following tasks.



**Packaging Options**

☒ Package your current project  
Use the project as the source for creating a new IP Definition.

☐ Package a block design from the current project  
Choose a block design as the source for creating a new IP Definition.

☐ Package a specified directory  
Choose a directory as the source for creating a new IP Definition.

**Create AXI4 Peripheral**

☐ Create a new AXI4 peripheral  
Create an AXI4 IP, driver, software test application, IP Integrator AXI4 VIP simulation and debug demonstration design.



## New IP Creation

The following pieces of information will be gathered:

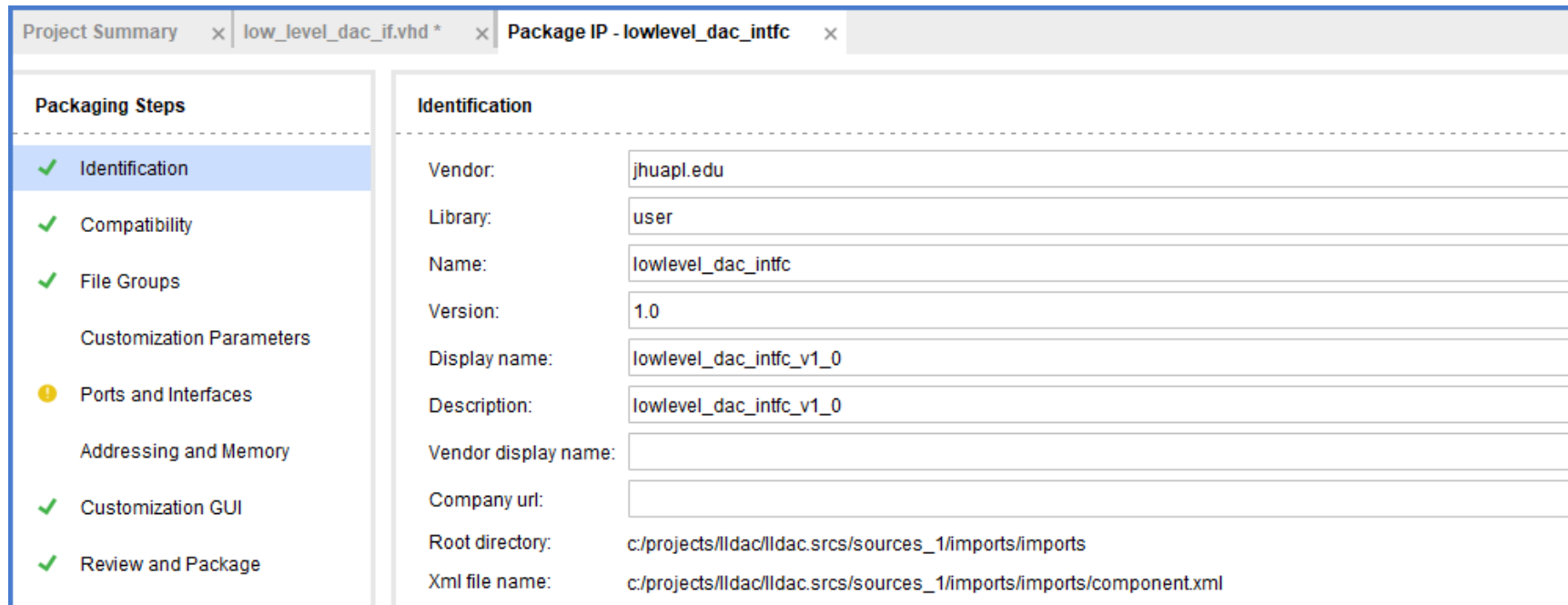
- Identification information based on top module name
- Family compatibility based on part in the project
- File(s) from Synthesis and Simulation file sets
- Ports from the file containing the top module
- Parameters from the file containing the top module
- Bus Interfaces based on port names
- Address Spaces and Memory Maps based on inferred bus interfaces

Following file will be created on disk along with corresponding customization files:  
c:/projects/lldac/lldac.srscs/sources\_1/imports/imports/component.xml

This dialog is a very good summary of the packaging process. In other words, what metadata about this core (and the use thereof) is required for it to be a plug-in block in IP integrator? Also, what files should go along with the core if we are sending it to be used by someone else? Instruction manual? Testbench? This is also the appropriate place for including things which would be useful even if not strictly required to be compiled into a design

## Main Packaging Dialog

- Project itself has a new file added that makes the project a “Packager Project”, meaning that this extra tab pops up when you open it
- Info about the IP is collected through all these fields (walkthrough live done in lecture)
- When you click “Package” the actual final product is saved off to a directory of your choice (it can be moved freely at any time to any repository location)



Packaging Steps	
✓	Identification
✓	Compatibility
✓	File Groups
	Customization Parameters
!	Ports and Interfaces
	Addressing and Memory
✓	Customization GUI
✓	Review and Package

Identification	
Vendor:	jhuapl.edu
Library:	user
Name:	lowlevel_dac_intf
Version:	1.0
Display name:	lowlevel_dac_intf_v1_0
Description:	lowlevel_dac_intf_v1_0
Vendor display name:	
Company url:	
Root directory:	c:/projects/ldac/ldac.srscs/sources_1/imports/imports
Xml file name:	c:/projects/ldac/ldac.srscs/sources_1/imports/imports/component.xml



## End Result – Packaged IP

Share with				
This PC > Local Disk (C:) > projects > ip_repo > lowlevel_dac_interface				
^	Name	Date modified	Type	Size
	src	4/13/2020 3:51 PM	File folder	
	xgui	4/13/2020 3:51 PM	File folder	
	component	4/13/2020 3:52 PM	XML Document	15 KB

- In any project – part of the project settings are a list of IP repositories that you point to. If (in this case) c:\projects\ip\_repo is selected, then “lowlevel\_dac\_interface” will show up as a choice in your IP catalog
- If you put it in the wrong spot – no worries at all, you can grab the created files and move them around wherever you want as long as you move the whole IP together





## Editing / Repackaging Previously Packaged IP

Share with				
This PC > Local Disk (C:) > projects > ip_repo > lowlevel_dac_interface				
^	Name	Date modified	Type	Size
	src	4/13/2020 3:51 PM	File folder	
	xgui	4/13/2020 3:51 PM	File folder	
	component	4/13/2020 3:52 PM	XML Document	15 KB

- Editing the source of an IP and changing it's packaging information is done in a very similar way to when you first packaged it, but this time you can start with the existing IP.
- In any design that can see the IP in the catalog, right click on the IP in the catalog, or in any design that is using it, and say "Edit in IP Packager". This will open up a temporary new Vivado project which will repackage that IP in the same repository. Here you can edit it, don't forget to Repackage.
- At the last step of packaging, click "yes" to create an archive. This is a zip file of what you packaged and nothing else



## Some Mechanics

- To see the IP (and the status of that IP) used in your design, Tools -> Report IP Status
  - Shows IP that can be upgraded (the design uses an old version, but there is a later version available in the Repo)
  - Shows IP that is missing (the design uses an IP, but it isn't in the repository)
- The tool does not automatically upgrade an IP, you need to do this explicitly
- See the list of Repositories where Vivado will look in “Project Manager → Settings → IP → Repository.
- When you use an IP, your project contains none of the HDL in the IP itself
  - If IP in a block diagram, the BD file itself contains a reference to the core and it's parameters.
  - If you use the core by instantiating it in HDL, the file which is in your project is an “XCI” file.  
The XCI file should be thought of as a reference to a particular core (VLNV) and parameters
- Now we will package the instructor solution lowlevel\_dac\_interface together. For Lab 5 it will be expected that you use the lowlevel-dac-interface IP instead of HDL in your project (just to give some practice). Then, in lab 6 we will build the entire radio as a single AXI peripheral.

