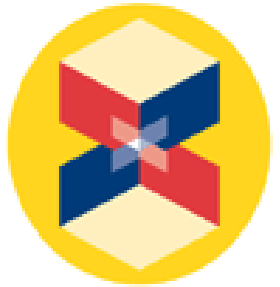
Digitronix Nepal
Innovation from Engineers...

FPGA Design with Matlab & Simulink

Course Prepared by
Digitronix Nepal
www.digitronixnepal.com
Email: digitronixnepali@gmail.com



XILINX
SYSTEM
GENERATOR™
For DSP

Sections and Lecturers of the Course:

Section 1. Installing Matlab/Simulink and Xilinx VIVADO/ISE Design Suit

Lecture 1 : Downloading and Intalling Matlab/Simulink and Xilinx ISE

Lecture 2 : Version Compatibility for Matlab/Simulink and ISE/VIVADO

Section 2. Introduction to Matlab/Simulink and HDL Coder/System Generator

Lecture 1 : Matlab and Simulink Introduction

Lecture 2 : HDL Coder and System Generator Introduction

Lecture 3 : Program Xilinx Zynq SoC Devices with Embedded Coder and HDL Coder

Lecture 4 : Program Xilinx FPGAs Using HDL Coder with Xilinx System Generator

Section 3. Basic Project with System Generator and FPGA

Lecture 1 : Setting up FPGA and JTAG configuration (Adding your Board) to Sys Gen.

Lecture 2 : Dump in to Spartan 3E FPGA

Lecture 3 : Dump in to Zynq FPGA (Zedboard/Zybo)

Sections and Lectures of the Course: Continue...

Digitronix Nepal
Innovation from Engineers...

Section 4. Advance Design with HDL Coder

Lecture 1 : Running an audio filter on live audio input using a Zynq board

Lecture 2 : Resettable Subsystem Support in HDL Coder

Lecture 3 : Creating IP on Matworks HDL Coder: LMS Noise Filter

Lecture 3: Creating IP with HDL Coder and Vivado Design Suite

(Creating LMS Noise Filter IP)

Download & Extract the Attachment from the “Resource Available” Section on Dashboard
“Section 4 hdl_coder_lms_sources”

What is an IP core?

- An IP(intellectual property) core is a functional block of logic or data that is used in making a FPGA or ASIC for a product.
- IP cores can be reused in many designs.
- IP cores fall into three categories:
  - Hard cores: Hard cores are physical manifestations of IP design. These are best for plug-and-play applications are less portable and flexible that the other two types of cores.
  - Firm cores: Firm cores or semi-hard cores also carry placement data but are configurable to various applications.
  - Soft cores: Soft cores is the most flexible of the three and they exist either as a netlist or HDL code.
- UARTs, Ethernet controllers, PCI interfaces, CPUs are examples of IP cores.

HDL Coder

- HDL coder is a Matlab tool that generates portable, synthesizable HDL code from Matlab functions and Simulink models.
- This HDL code generated by HDL coder can be used for FPGA programming.
- HDL Coder provides a workflow advisor that automates the programming Xilinx and Intel FPGAs.
- It can help us to control HDL architecture and implementation, highlight critical paths and generate hardware resource utilization estimates.

Configuring HDL coder in Matlab/Simulink and ISE 14.7 (we will show for Vivado later)

1. Recommended Version of MATLAB R2015a for ISE 14.7
2. Download and Install Xilinx ISE 14.7 (if you don't have)
3. Using windows explorer locate the Xilinx ISE Application within your local directory it might as: C:\Xilinx\14.7\ISE_DS\ISE\bin\nt\ise.exe

While our installation directory is: E:\Xilinx\14.7\ISE_DS\ISE\bin\nt, but we are showing now for above directory location for your setup.

4. Copy the address to the clipboard and open MATLAB. In MATLAB Command Window enter the following function: (command also shown in next slide)

hdlsetuptoolpath('ToolName','Xilinx ISE', 'ToolPath', 'C:\Xilinx\14.7\ISE_DS\ISE\bin\nt\ise.exe')

5. Successfully setting up the HDL Toolpath will result following result (shown in next slide)

Configuring HDL coder in Matlab/Simulink and ISE 14.7 (we will show for Vivado later)

6. Successfully setting up the HDL Toolpath will result following result

```
>> hdlsetuptoolpath('ToolName', 'Xilinx ISE', 'ToolPath', 'C:\Xilinx\14.7\ISE_DS\ISE\bin\nt\ise.exe')
Setting XILINX environment variable to:
C:\Xilinx\14.7\ISE_DS\ISE
Setting XILINX_EDK environment variable to:
C:\Xilinx\14.7\ISE_DS\EDK
Setting XILINX_PLANAHEAD environment variable to:
C:\Xilinx\14.7\ISE_DS\PlanAhead
```

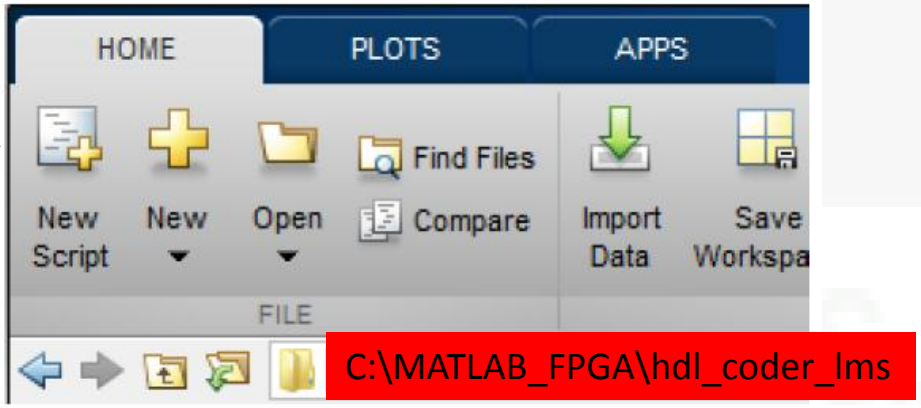
7. Now HDL coder can be used to Synthesize HDL code for Xilinx Platform. Now download the folder we have provided on video in “C:\MATLAB FPGA” folder.

8. Now change the working directory of Matlab,

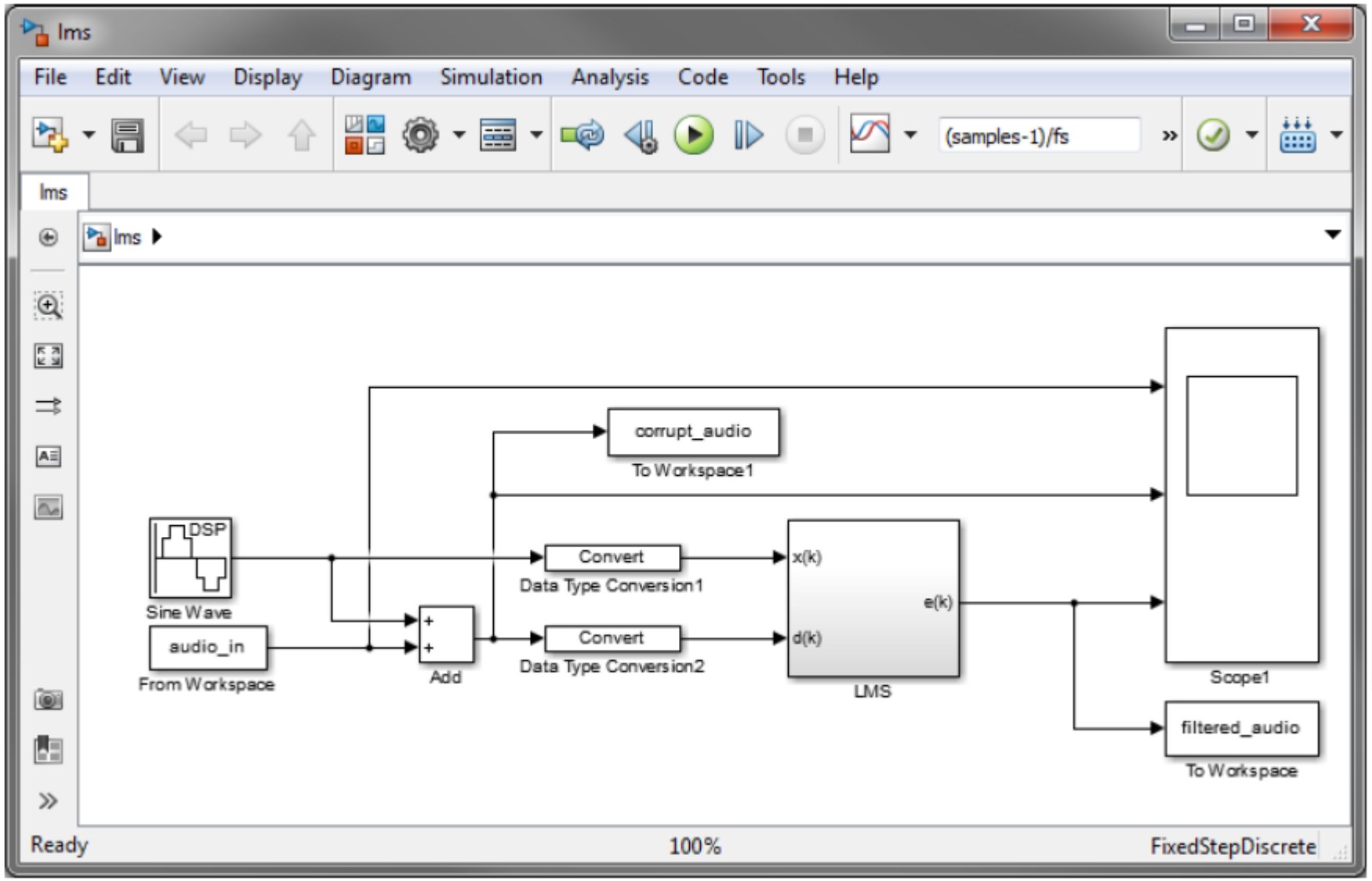
   As shown in the figure

9. Open LMS simulnk model,

   By double click on lms.slx



10. You will see following Simulink Model



Configuring HDL Coder with VIVADO

(This Library are Not Needed for this session!!!)

The MathWorks Zynq 700 support packages need to be installed while need to design with Zynq based Simulink Block as Zynq Block:

- Embedded Coder Support Package for Xilinx Zynq-7000 Platform - File Exchange - MATLAB Central
- HDL Coder Support Package for Xilinx Zynq-7000 Platform - File Exchange - MATLAB Central

Configuring HDL coder in Matlab/Simulink and VIVADO Design Suit 2017.2

1. Recommended Version of MATLAB 2016a,2016b,2017a for VIVADO 2017.2 (but we are showing with MATLAB R2013b however you can follow recommended one)
2. Download and Install Xilinx VIVADO 2017.2 or (if you don't have)
3. Using windows explorer locate the Xilinx VIVADO Application within your local directory it might as: **C:\Xilinx\Vivado\2017.2\bin\Vivado.bat**

   which is: PATH_TO_YOUR_VIVADO.BAT_FOR_2017_2

4. Copy the address to the clipboard and open MATLAB. In MATLAB Command Window enter the following function: (command also shown in next slide)

>>hdlsetuptoolpath('ToolName','Xilinx Vivado', 'ToolPath','C:\Xilinx\Vivado\2017.2\bin\Vivado.bat');

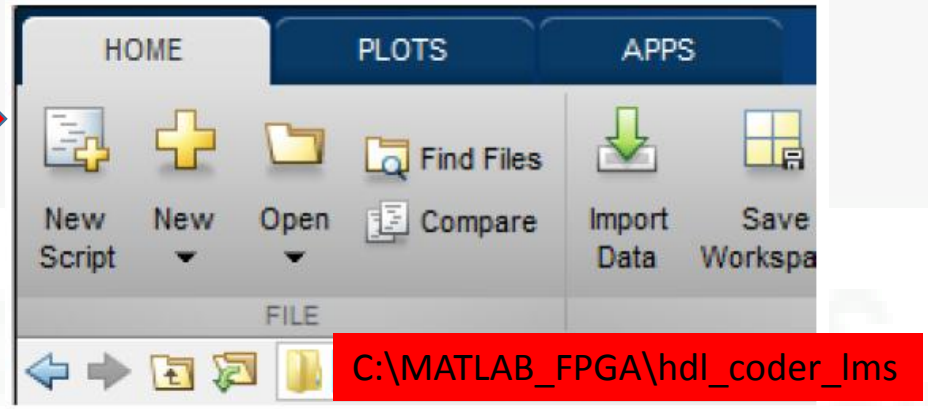
Configuring HDL coder in Matlab/Simulink and VIVADO Design Suit 2017.2

6. Successfully setting up the HDL Toolpath will result following result

7. Now HDL coder can be used to Synthesize HDL code for Xilinx Platform.Now download the folder we have provided on video in “C:\MATLAB_FPGA” folder.

8. Now change the working directory of Matlab,

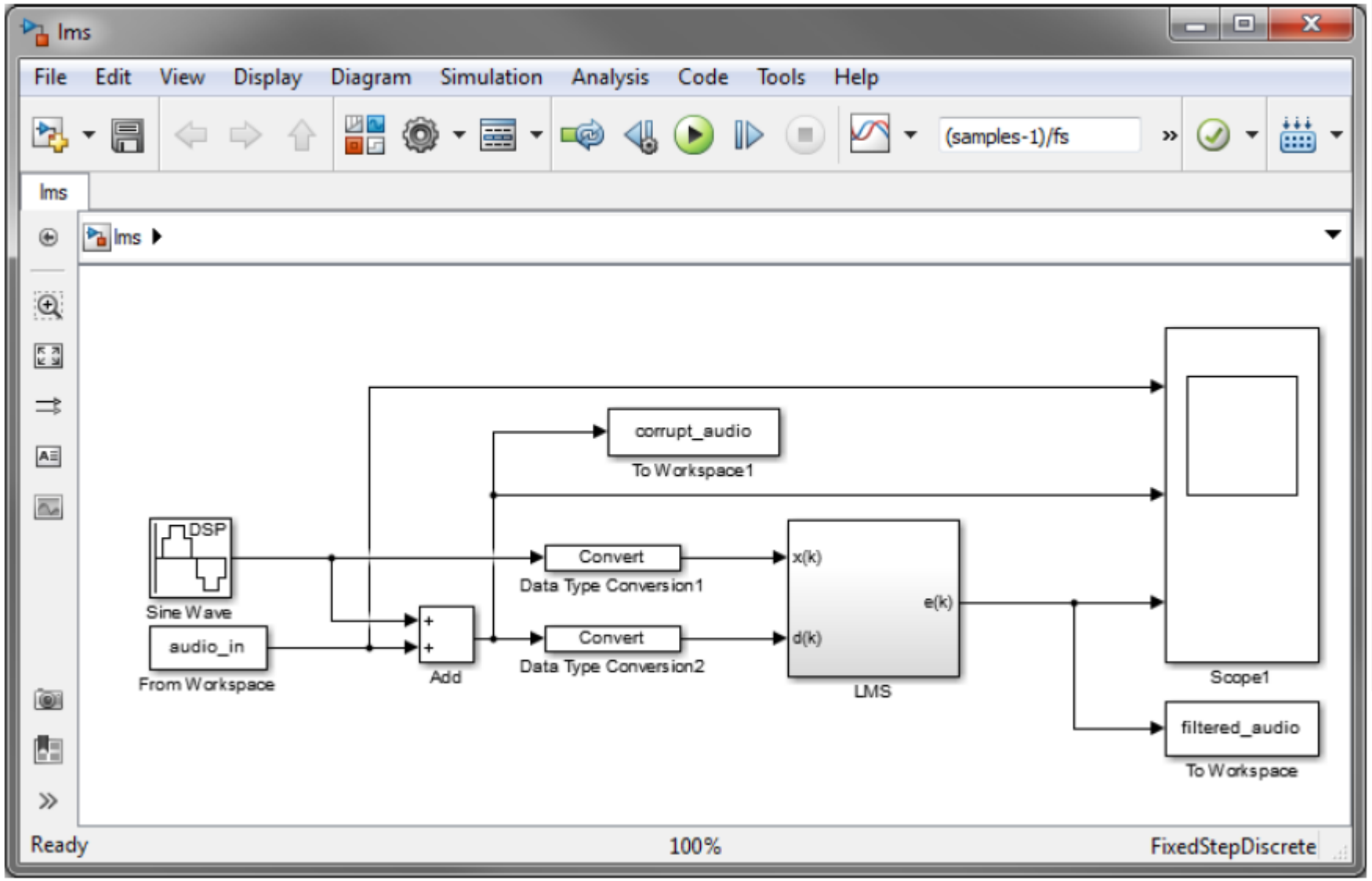
   As shown in the figure



9. Open LMS simulnk model,

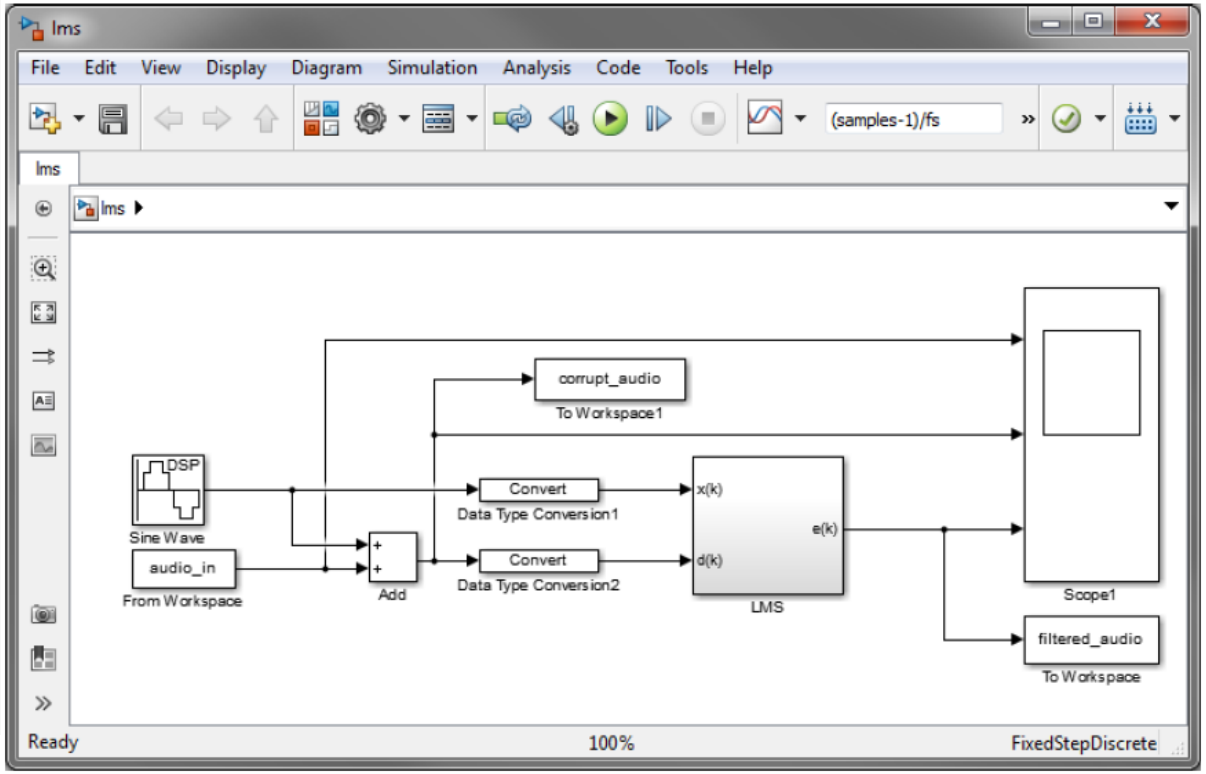
   By double click on lms.slx

10. You will see following Simulink Model



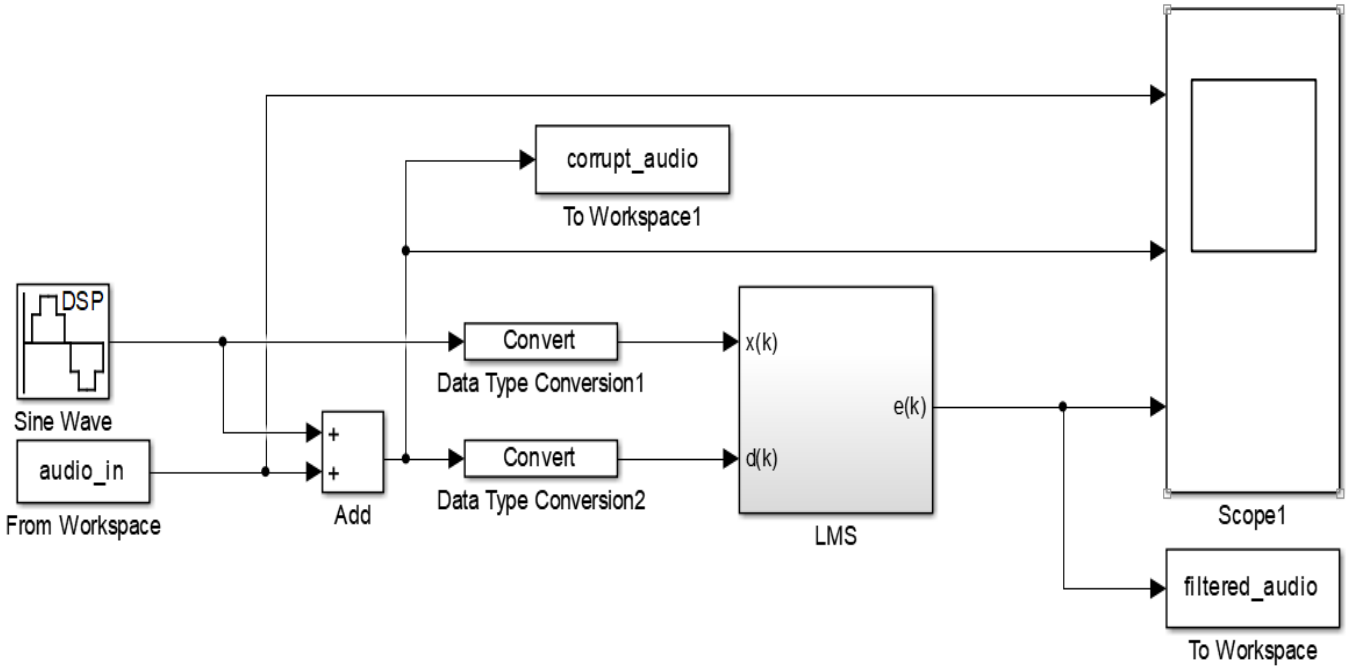
Simulink Model: Explanation

- A **Sine Wave** block which generates tonal noise.
- A **From Workspace** block which imports the audio samples from the MATLAB workspace.
- To generate HDL code for the Simulink LMS model using HDL coder, input to the system must be in fixed point numerical format.
- So there are two **Data Type Conversion** blocks for converting corrupt audio signal and tonal noise in to fixed point.
- Two **To workspace** are used to show LMS output and corrupt audio signal.



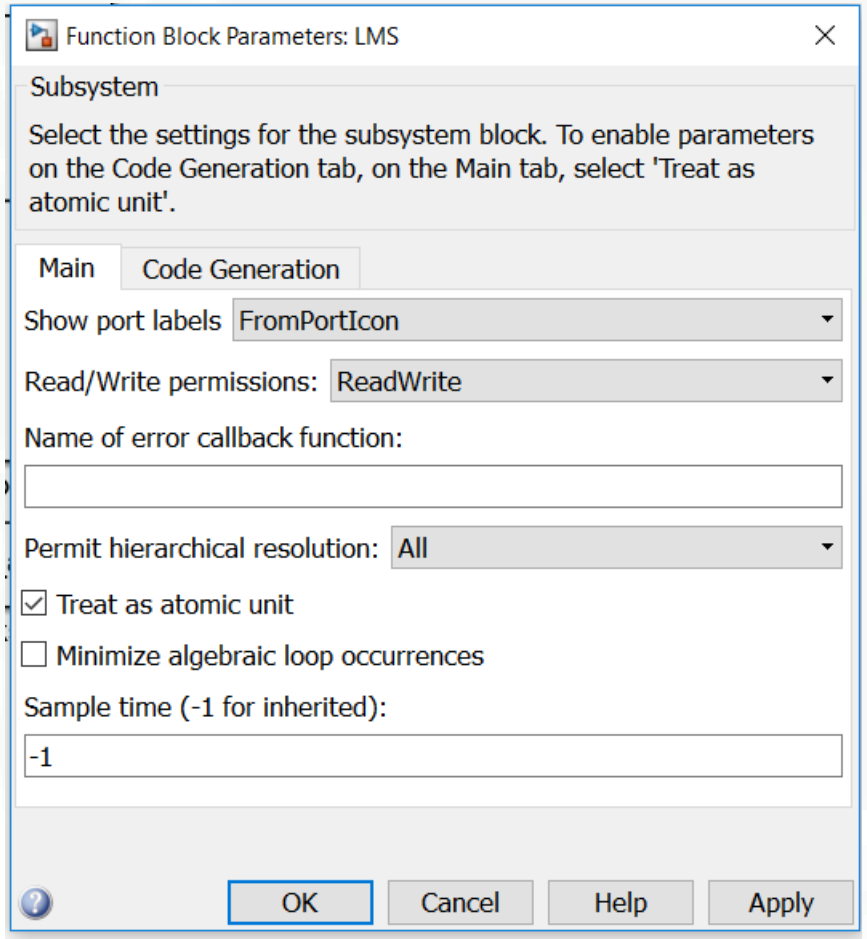
Reference: Zynqbook Tutorials

Generating HDL code



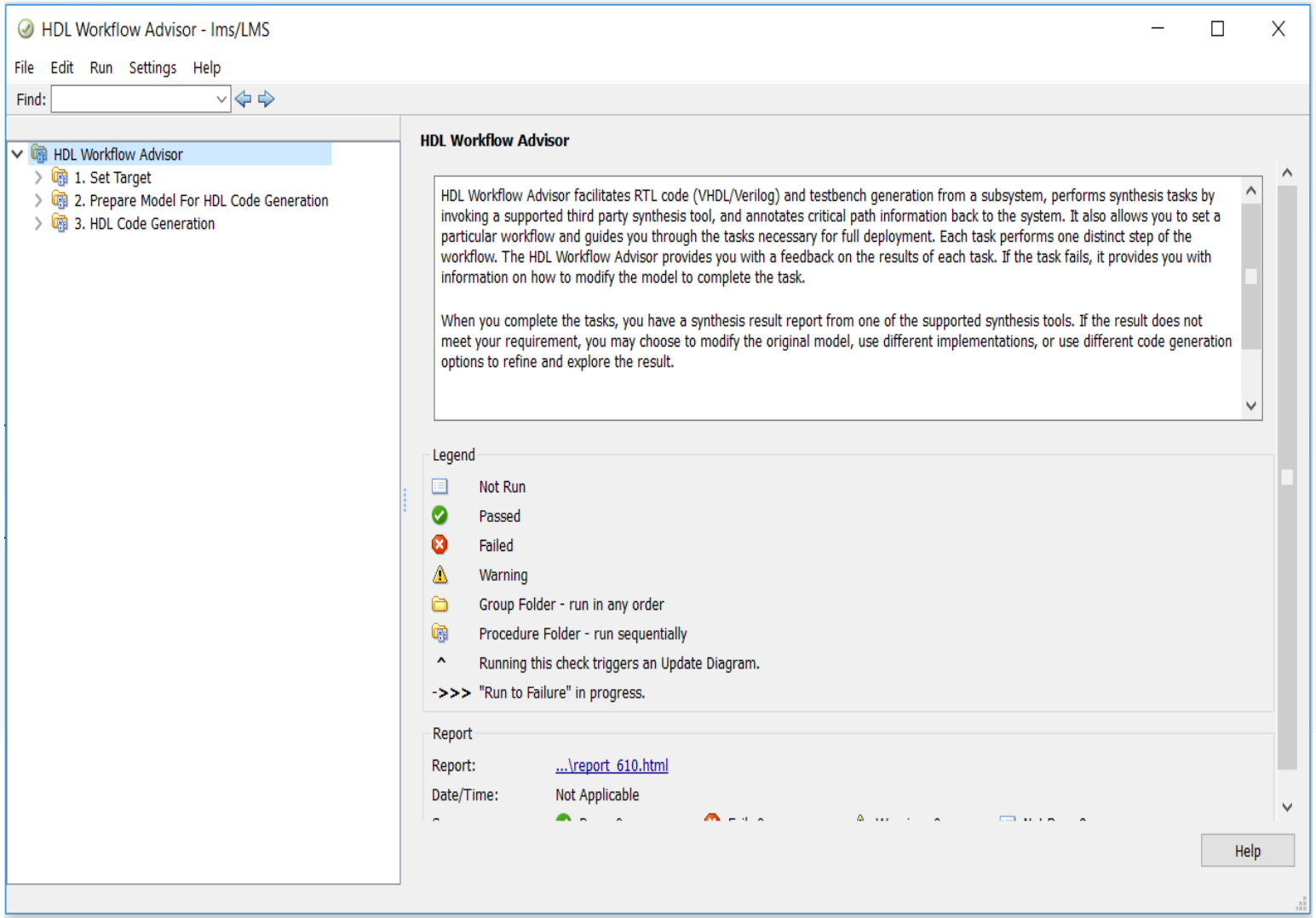
- We will use this model to generate HDL code
- We will generate HDL code for LMS subsystem.

Generating HDL code



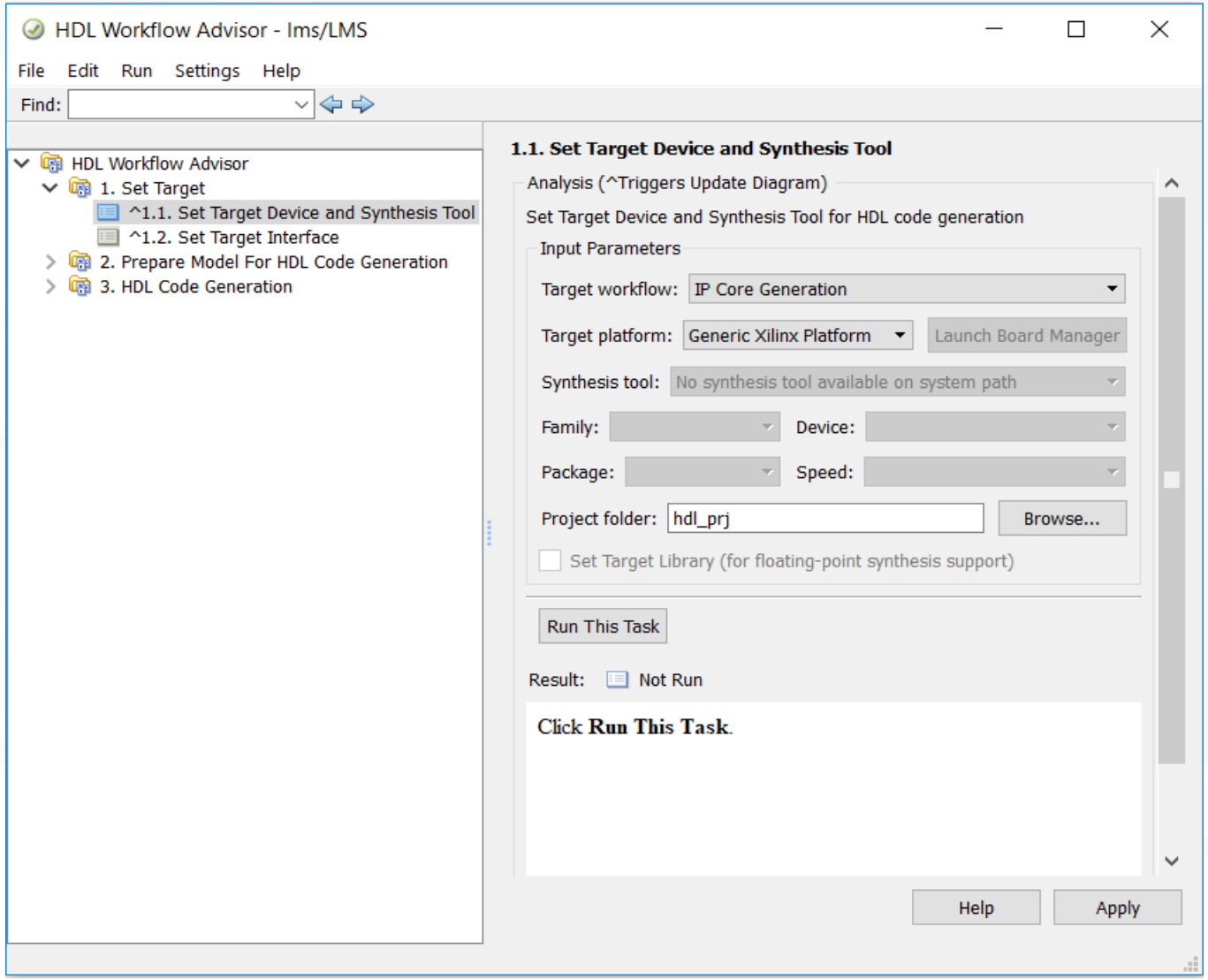
- Before generating HDL code, LMS subsystem must be an atomic subsystem.
- Right Click on LMS subsystem and select Block Parameters(Subsystem) option.
- Then a dialog box shown on left opens.
- Tick option “Treat as atomic unit” and then ok.
- In Simulink Subsystems can be virtual or atomic. Simulink ignores virtual subsystem boundaries when determining block update order. By contrast, Simulink executes all blocks within an atomic subsystem before moving on to the next block.
- So if we select “Treat as atomic unit” option when it comes time to execute the subsystem, Simulink executes all blocks within the subsystem before executing any other block at the same level as the subsystem block.

HDL Workflow Advisor



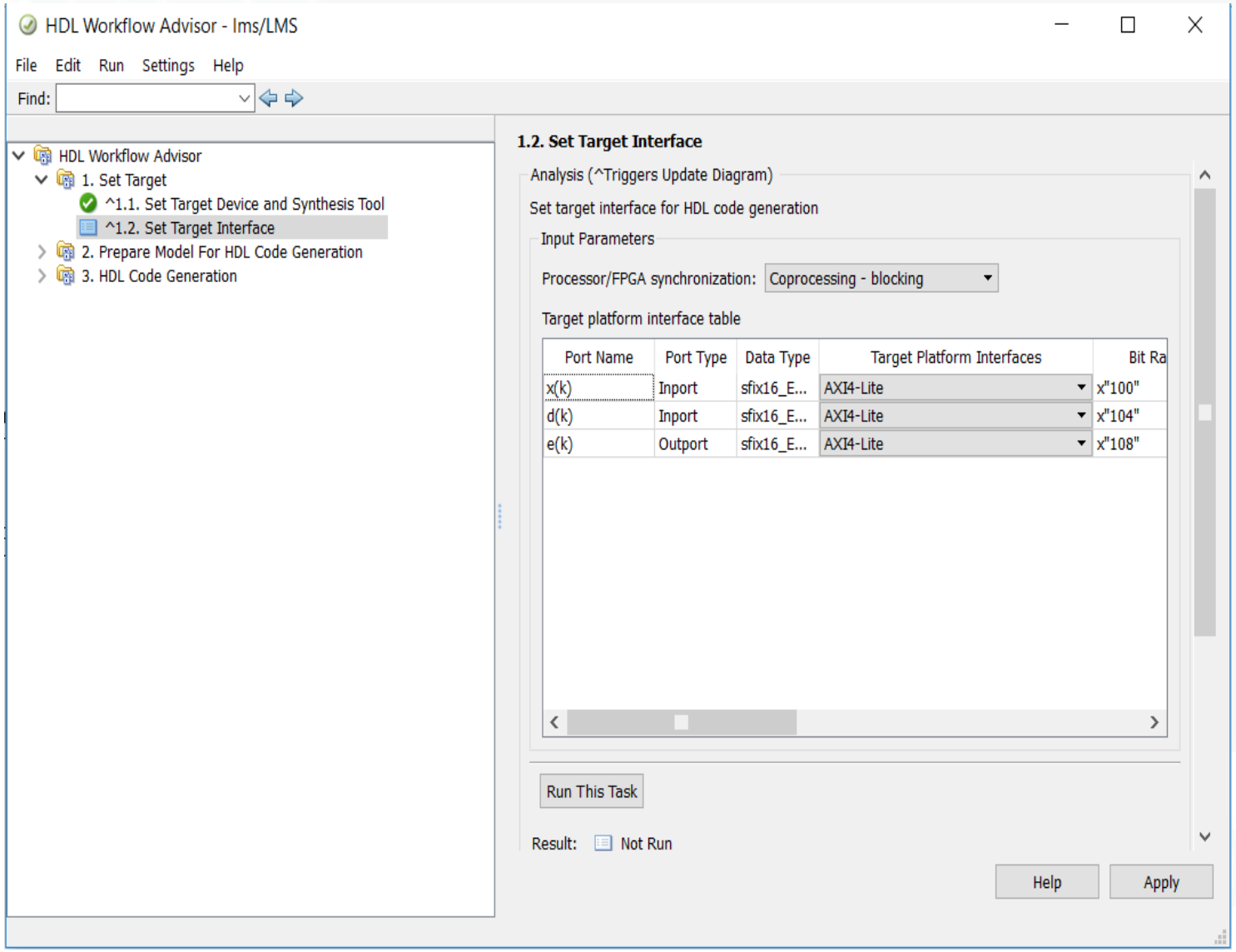
- Right-Click LMS Subsystem and Select HDL Code>HDL Workflow Advisor
- HDL workflow dialog box will open as shown in the figure.
- The HDL Workflow Advisor guides you through the steps required to generate RTL code for your design

HDL Workflow Advisor



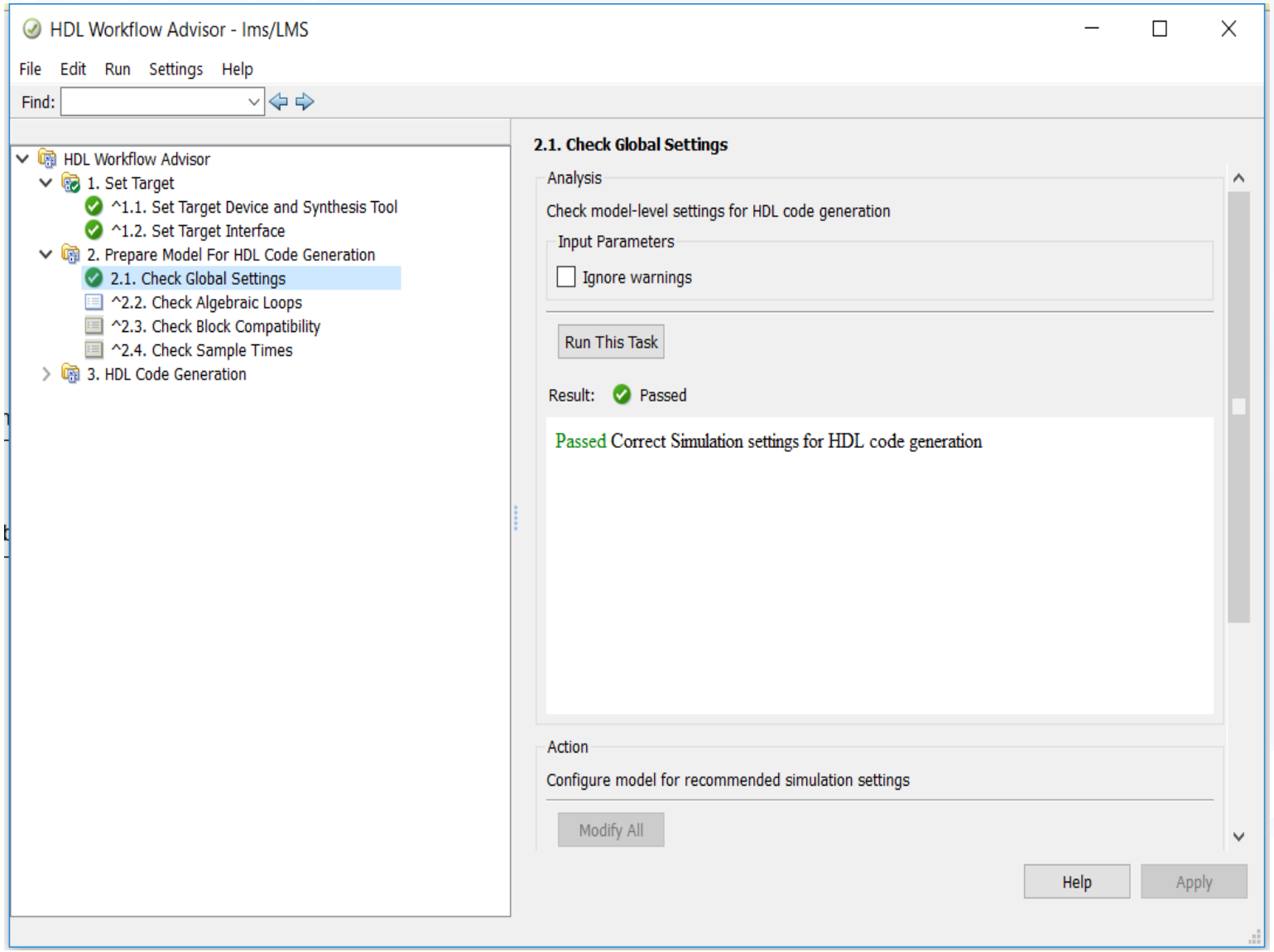
- In the left panel, expand Set Target and select 1.1 Set Target Device and Synthesis Tool.
- In the Input Parameters, select IP Core Generation as Target workflow and Generic Xilinx Platform as the Target platform as shown in figure.
- Click Run This Task to apply the settings.

HDL Workflow Advisor



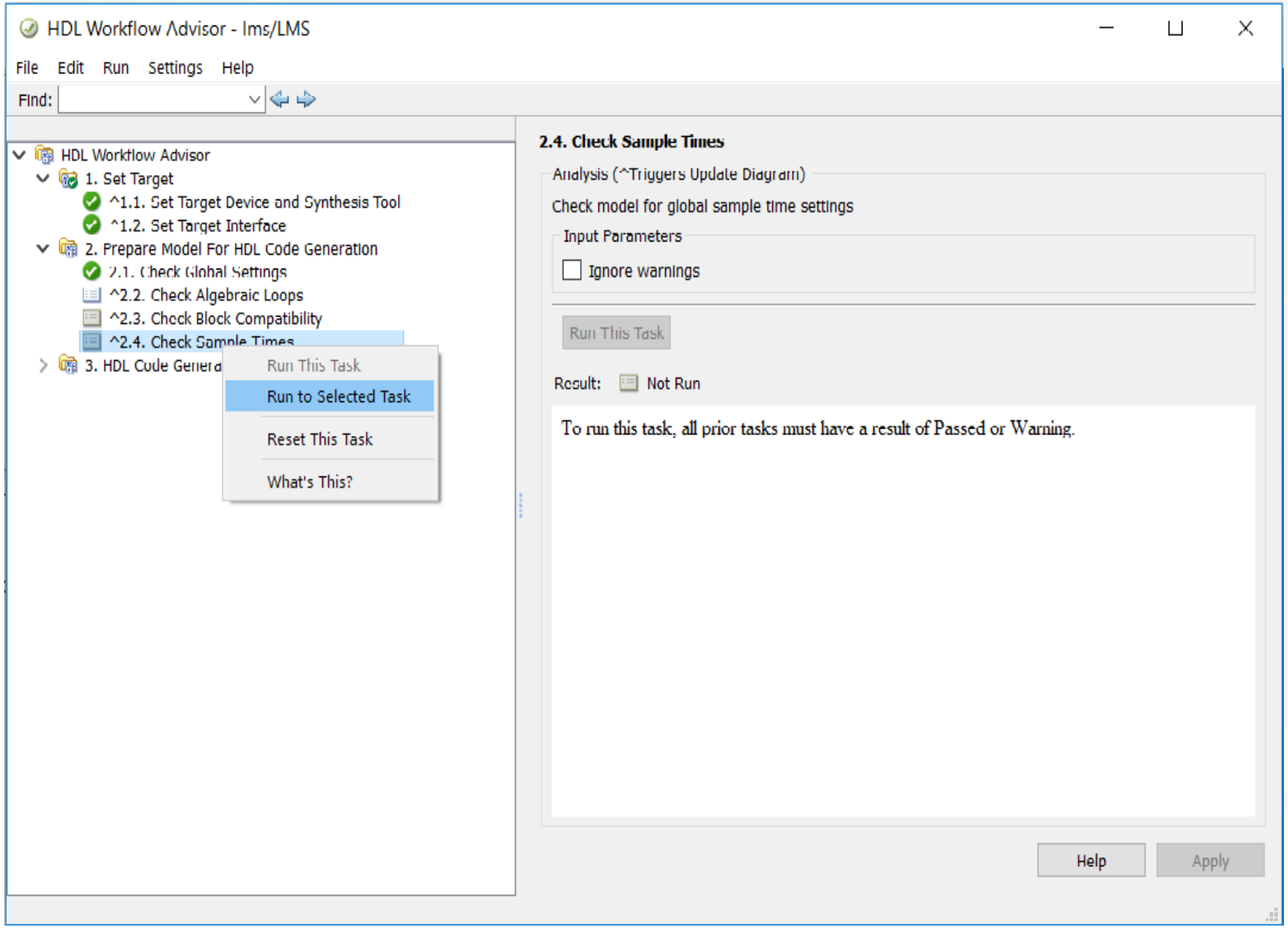
- Select Set Target Interface and specify the target interface for HDL code generation.
- Select Coprocessing-blocking as the Processor/FPGA synchronization.
- This will automatically infer an AXI4-Lite interface for all ports in the design, and specify a memory address for each.
- Click Run This Task to apply the settings.

HDL Workflow Advisor



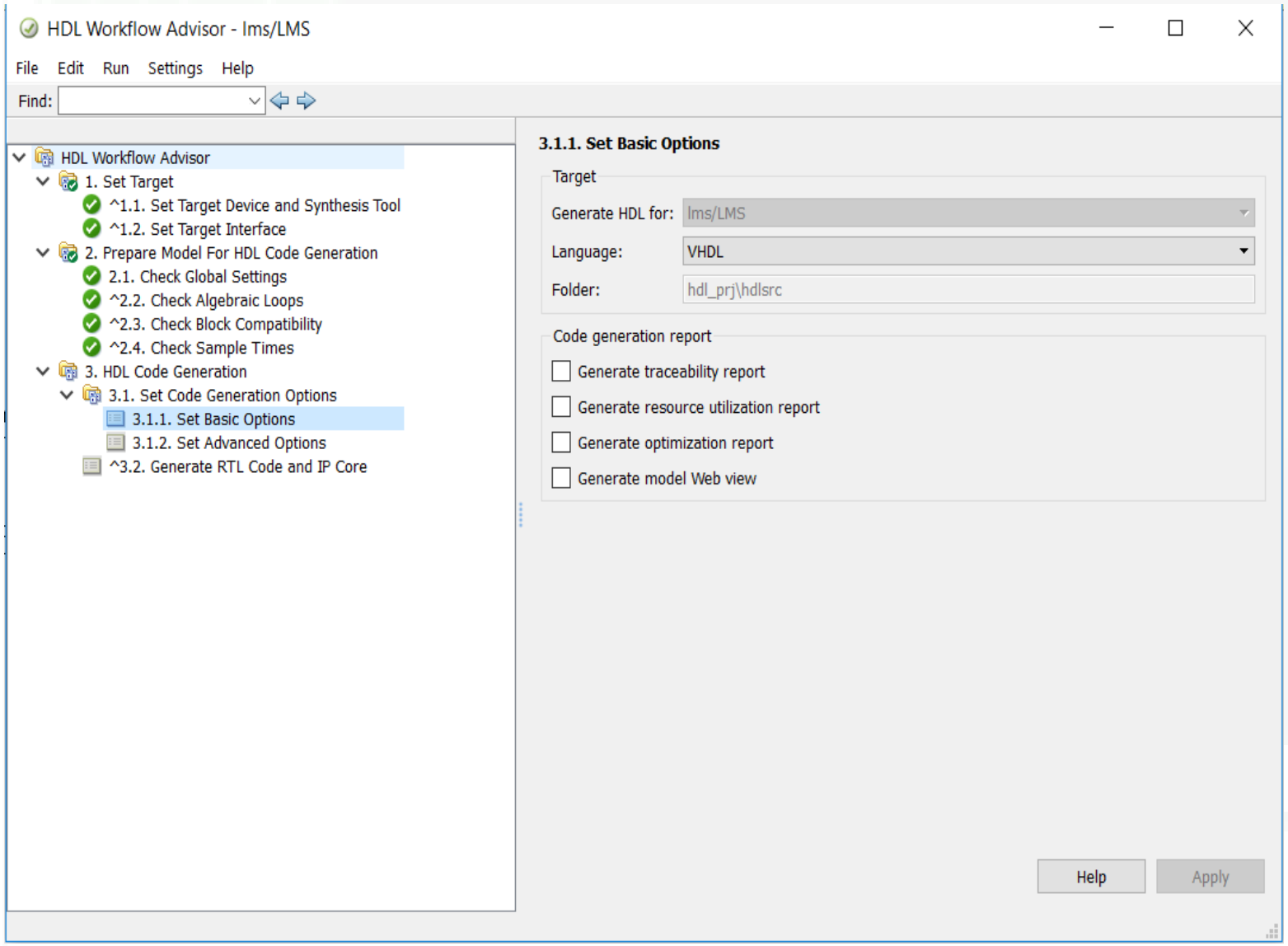
- Now expand Prepare Model for HDL Code Generation in the left panel and select Check Global Settings.
- Click Run This Task to check the model-level settings.
- If this step fails, click Modify All to allow HDL Workflow Advisor to modify the settings.
- Now this step should pass and you will be presented with a table of results.

HDL Workflow Advisor



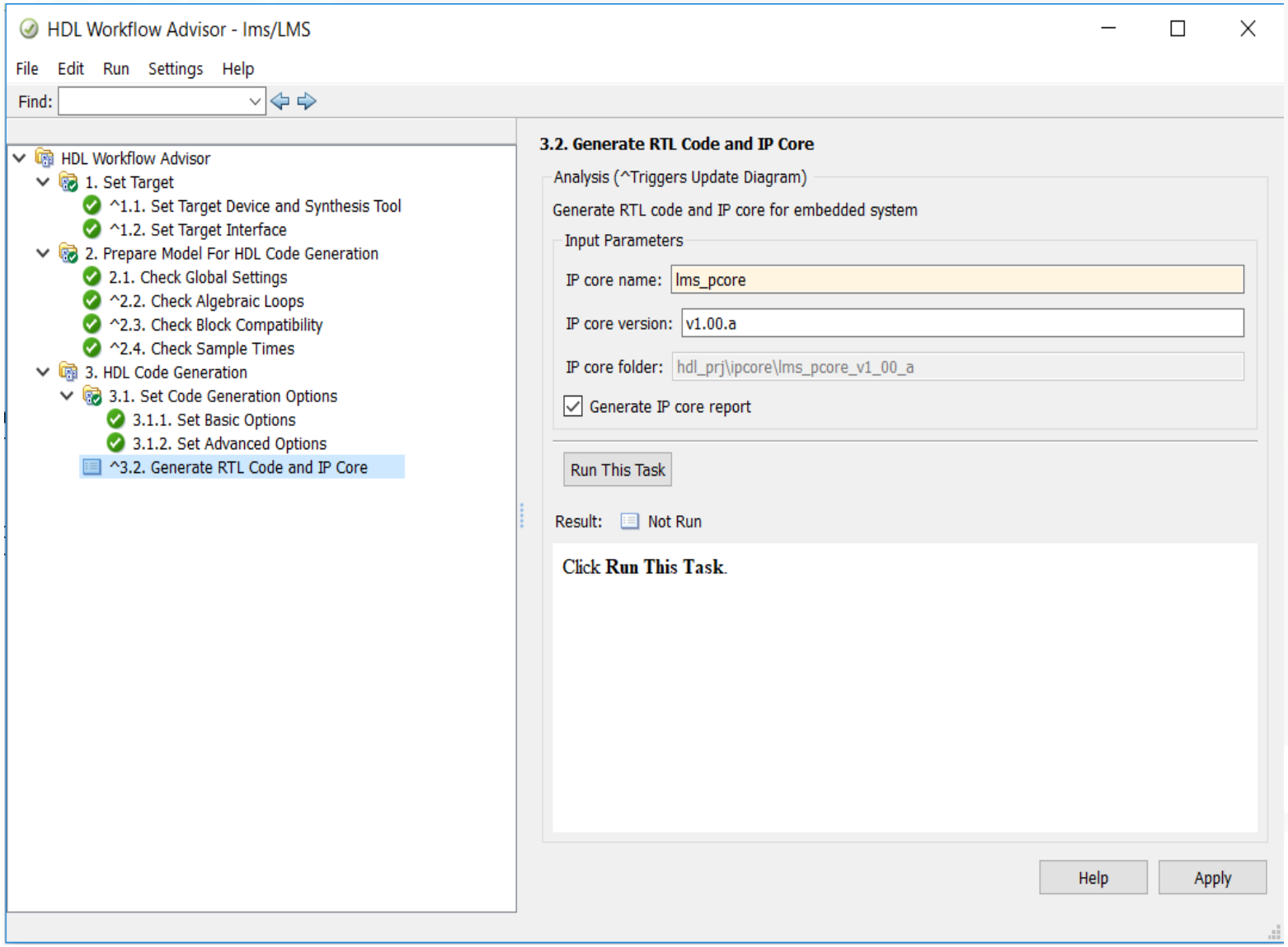
- 2.2 Check Algebraic Loops, 2.3 Check Block Compatibility and 2.4 Check Sample Times are all checking operations and can be performed in batch
- Right-click on Check Sample Times in the left hand pane and select Run to Selected Task.
- This will perform checks one after another till check sample times is completed.

HDL Workflow Advisor



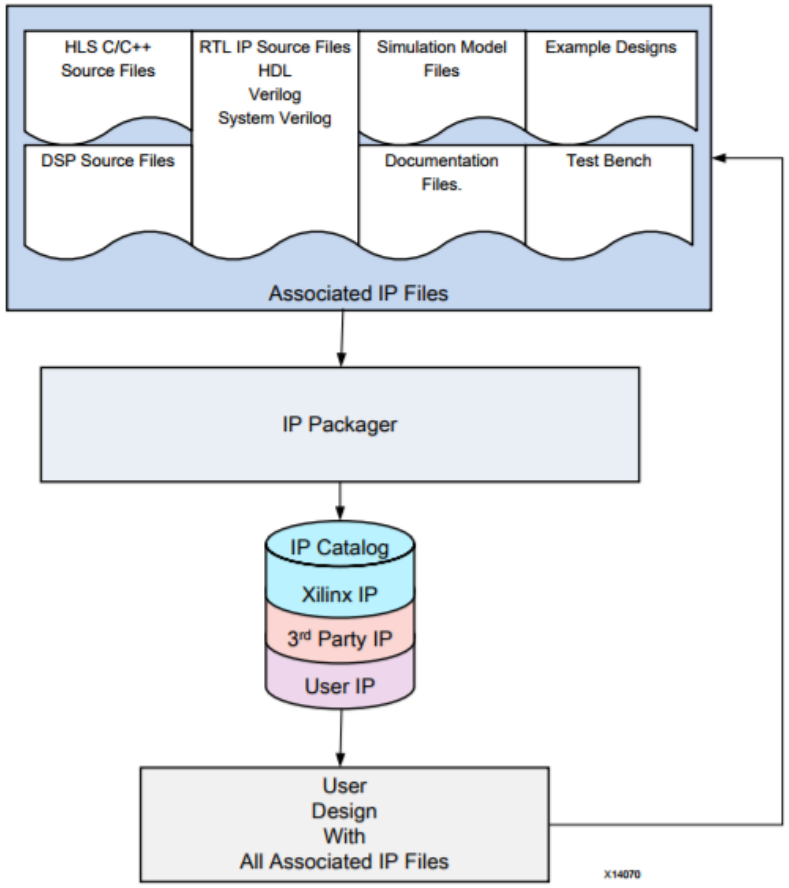
- Expand HDL Code Generation in the left pane, and further expand Set Code Generation Options.
- Click on Set Basic Options
- Select VHDL as the Language in the Target pane.
- Select Set Advanced options. Here we can specify more advanced options for the HDL code.
- We will leave all defaults for now.
- Right Click on Set Advanced Options and select Run to Selected Task.

HDL Workflow Advisor



- Select Generate RTL code and IP core from the left hand panel.
- This step will generate HDL code for our LMS IP core.
- Set the IP core name as lms_pcore and Click Run This Task.

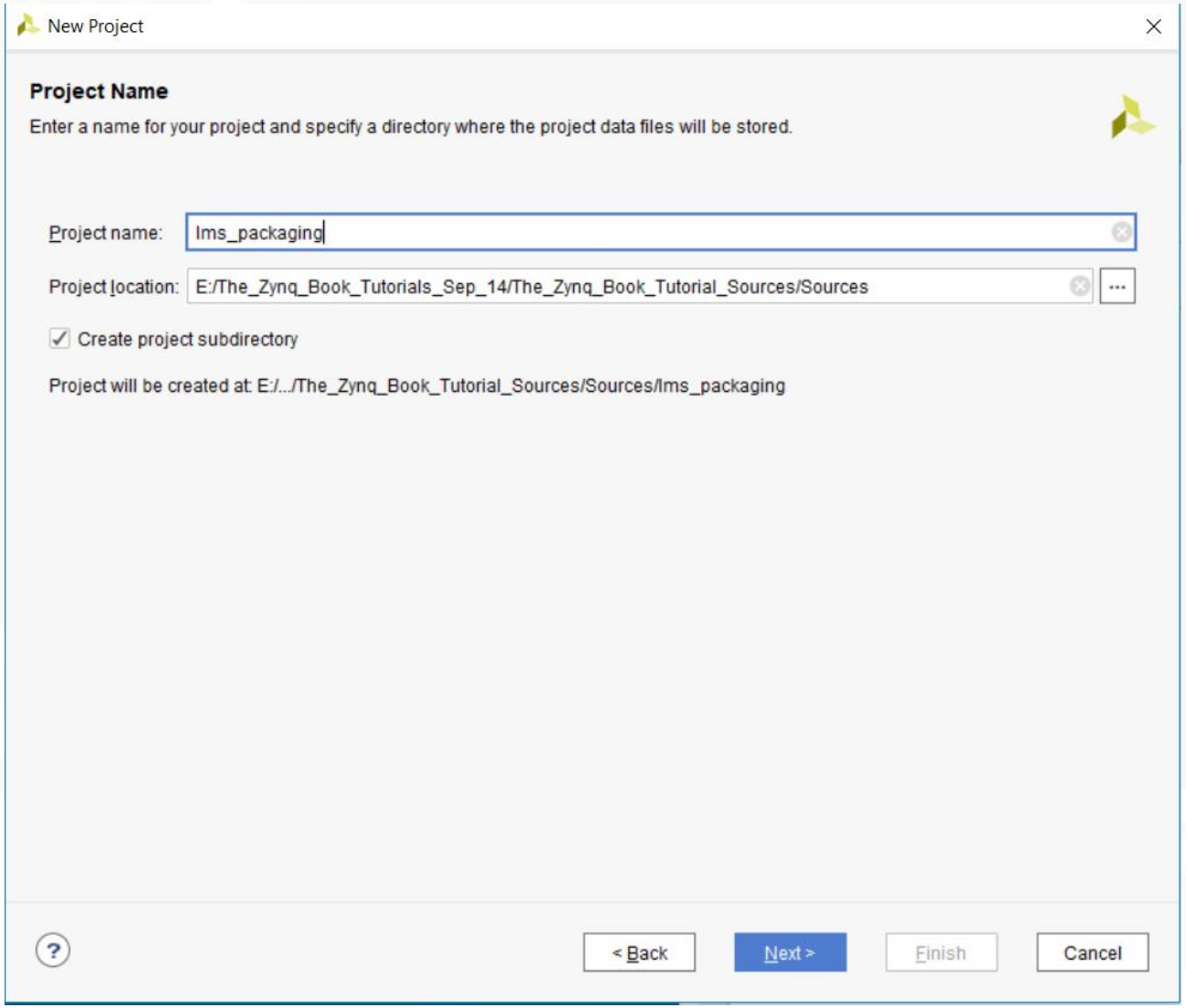
IP Packager



Vivado Design Suite IP Design Flow

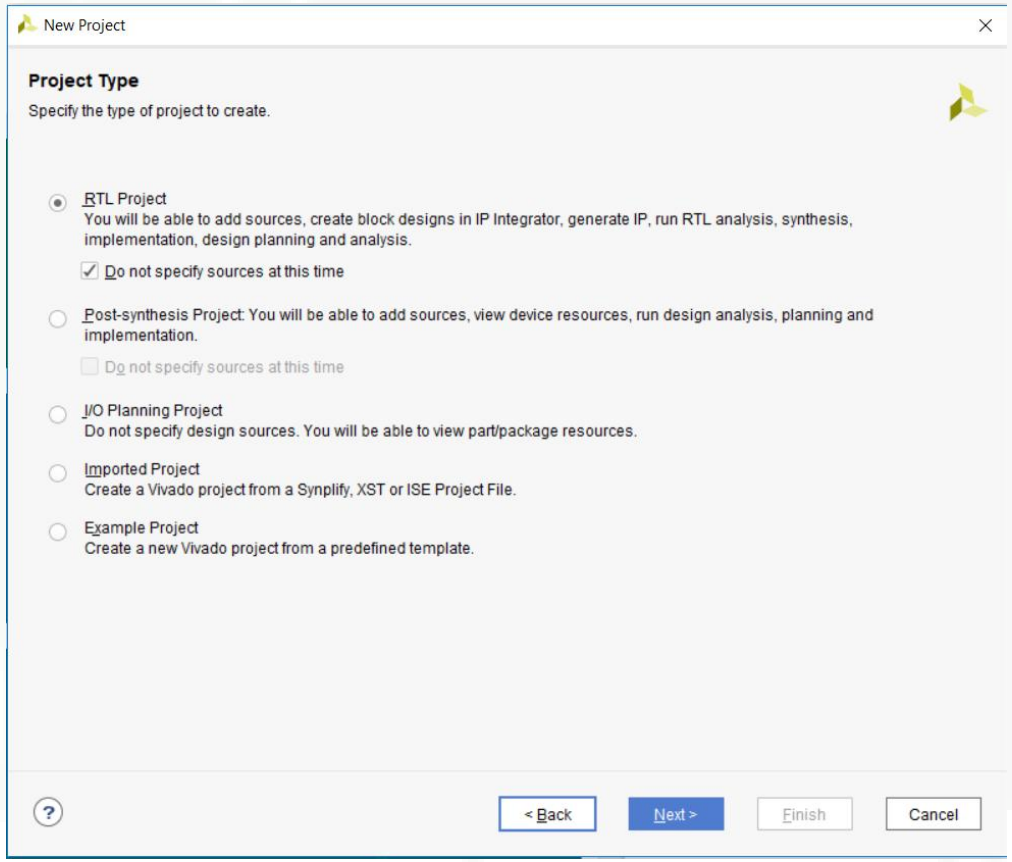
- The Vivado Design Suite provides an IP-centric design flow that helps you quickly turn designs and algorithms into reusable IP.
- As shown in the following figure, the Vivado IP catalog is a unified IP repository that provides the framework for the IP-centric design flow. This catalog consolidates IP from all sources including Xilinx® IP, IP obtained from third parties, and end-user designs targeted for reuse as IP into a single environment.
- The Vivado IP packager tool is a unique design reuse feature based on the IP-XACT standard. The IP packager tool provides any Vivado user the ability to package a design at any stage of the design flow and deploy the core as system-level IP.

Creating New Project

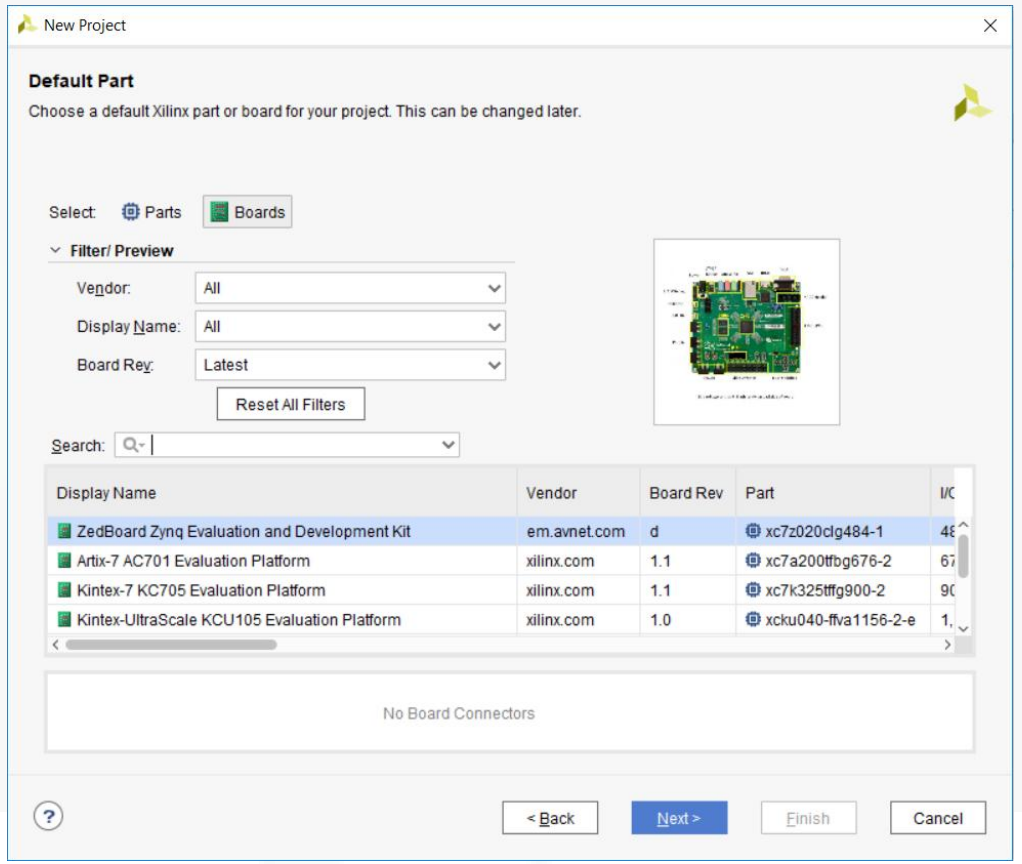


- Launch Vivado and create a new project called lms_packaging at any location, ensuring that the option to create a project subdirectory is selected.
- Also select VHDL as the target language and Zedboard as the default part.

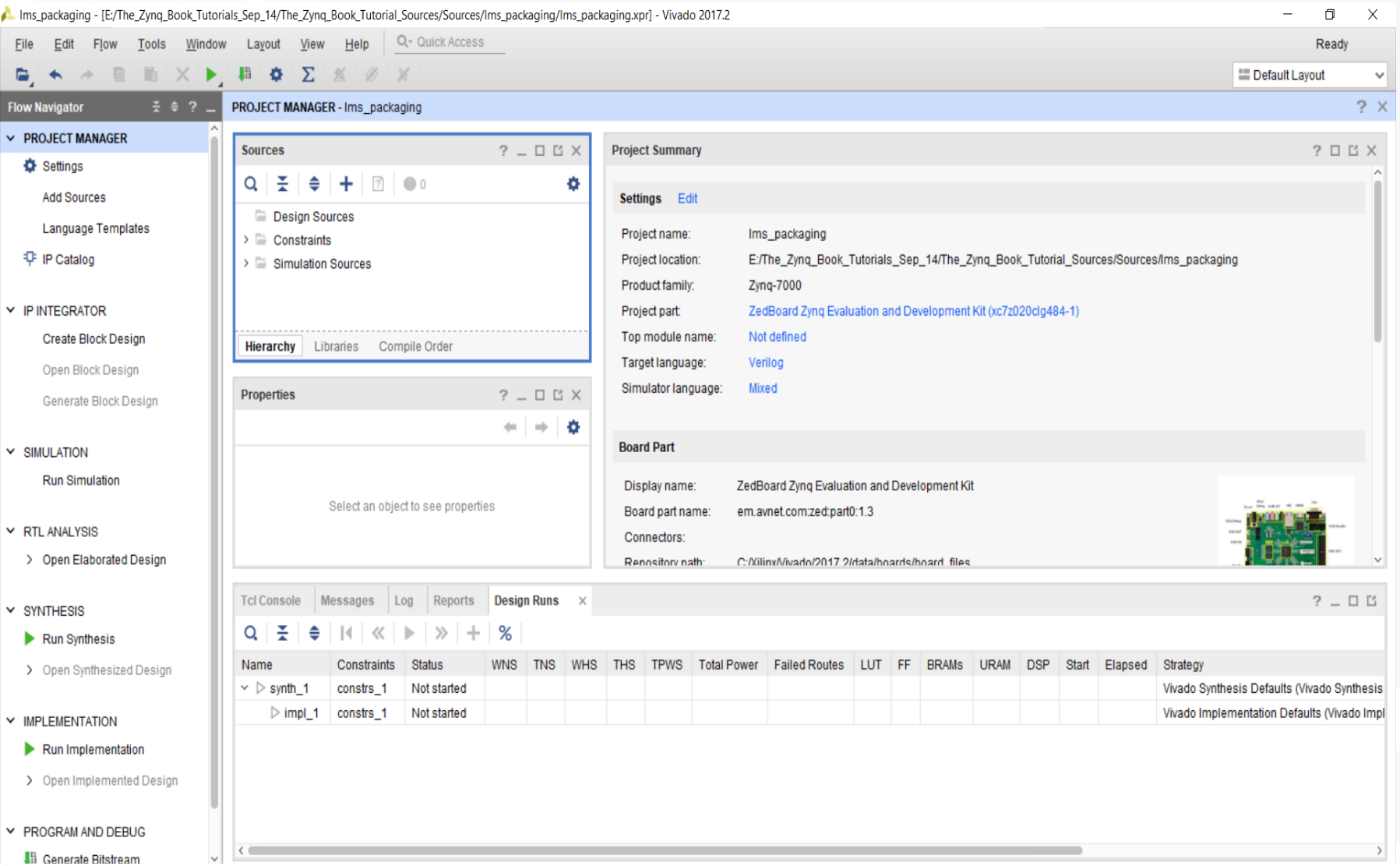
Creating New Project



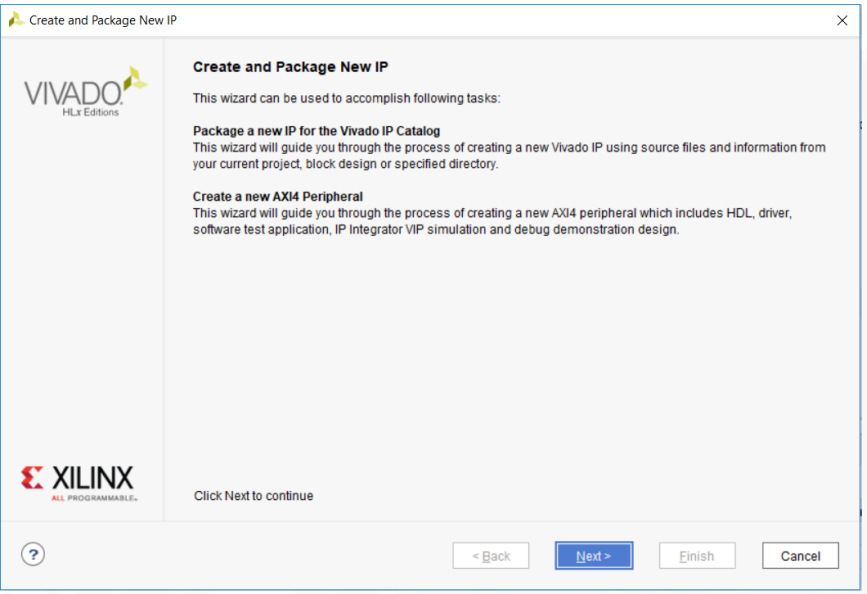
Next

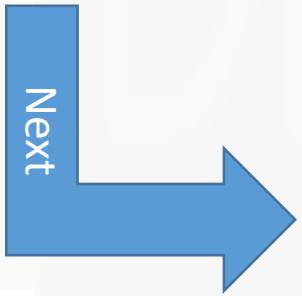


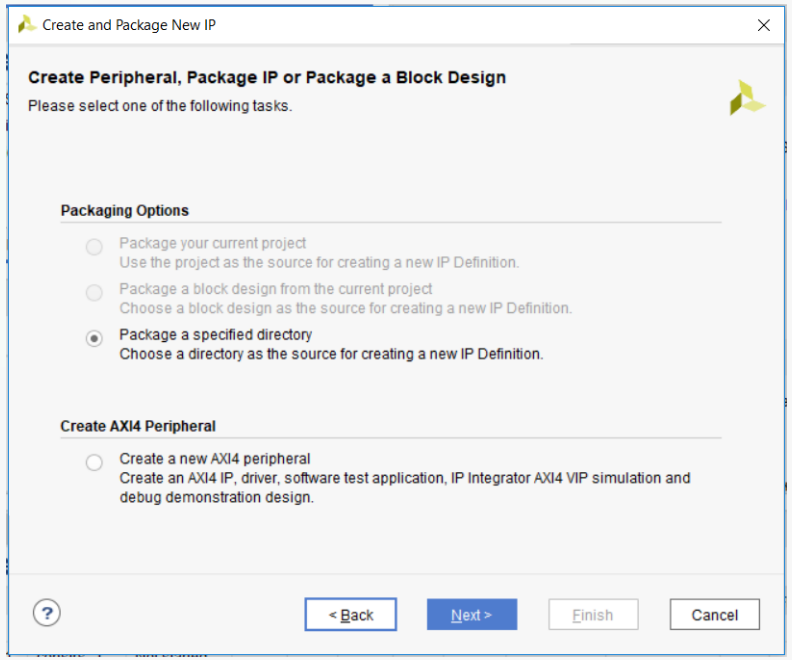
Creating New Project

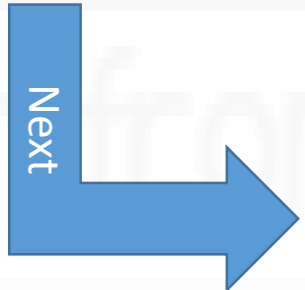


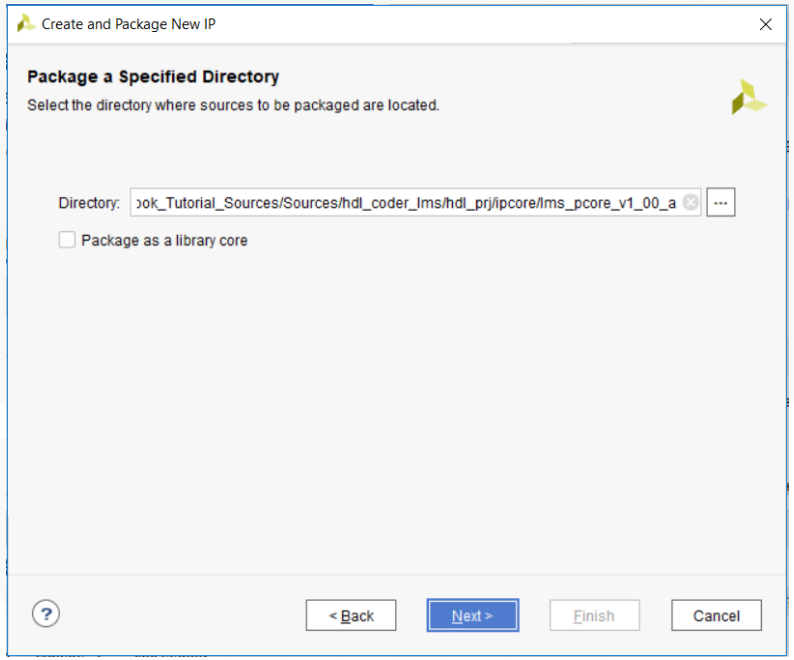
- After Creating Project we get to Window as shown in the figure.
- Now Select Tools>Create and Package IP from menu bar and Click Next
- Select the option to Package a specified directory and click Next.
- Enter location where you stored lms_pcore_v1_00_a as the IP Location. In this case it is E:/The_Zynq_Book_Tutorials_Sep_14/The_Zynq_Book_Tutorial_Sources/Sources/hdl_coder_lms/hdl_prj/ipcore/lms_pcore_v1_00_a


Create and Package New IP
VIVADO
HLx Editions
Create and Package New IP
This wizard can be used to accomplish following 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, block design 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 VIP simulation and debug demonstration design.
XILINX
ALL PROGRAMMABLE.
Click Next to continue
< Back
Next >
Finish
Cancel

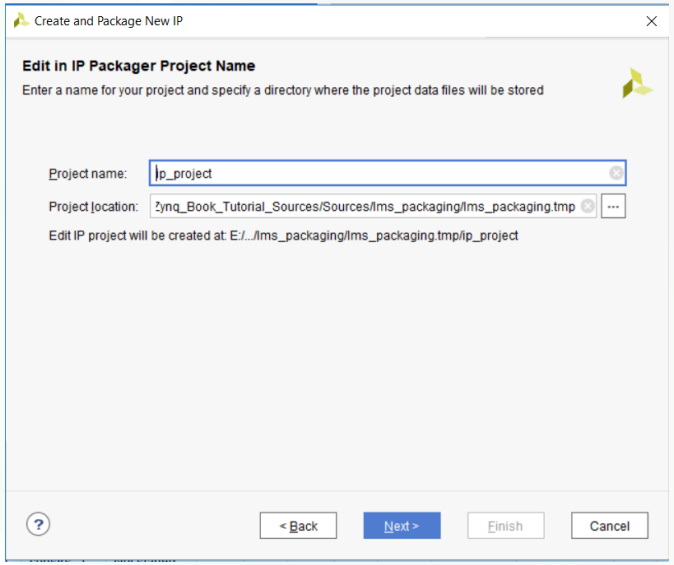

Next


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.
< Back
Next >
Finish
Cancel

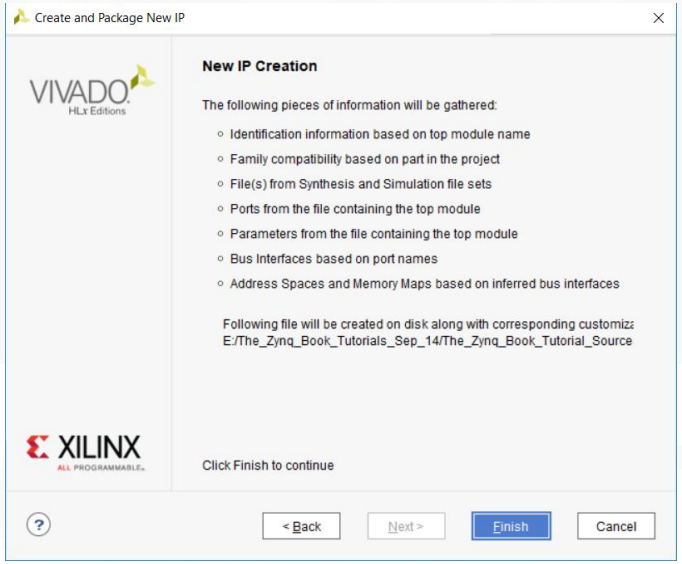

Next


Create and Package New IP
Package a Specified Directory
Select the directory where sources to be packaged are located.
Directory:
ok_Tutorial_Sources/Sources/hdl_coder_lms/hdl_prj/ipcore/lms_pcore_v1_00_a
Package as a library core
< Back
Next >
Finish
Cancel

IP Packaging

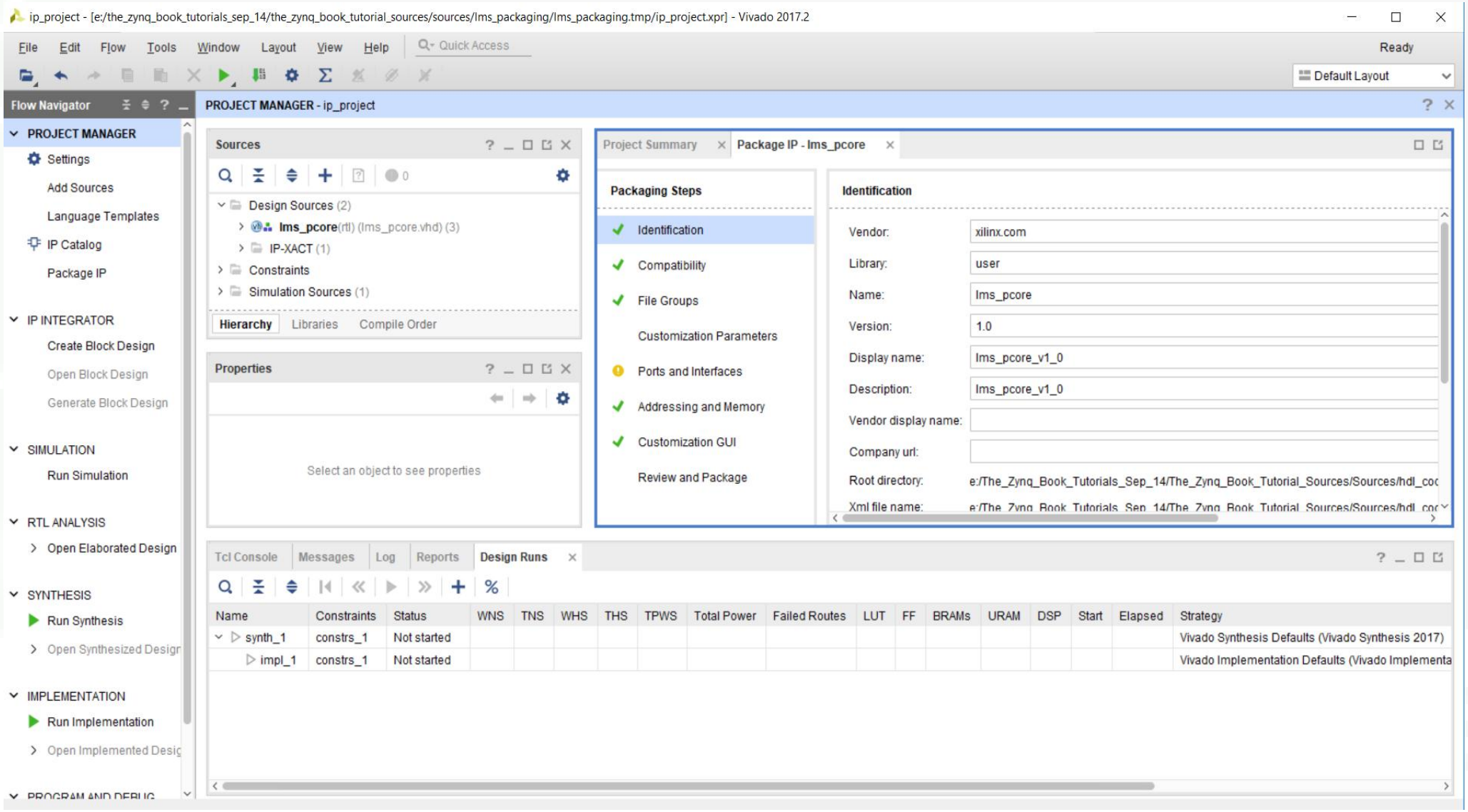


Next

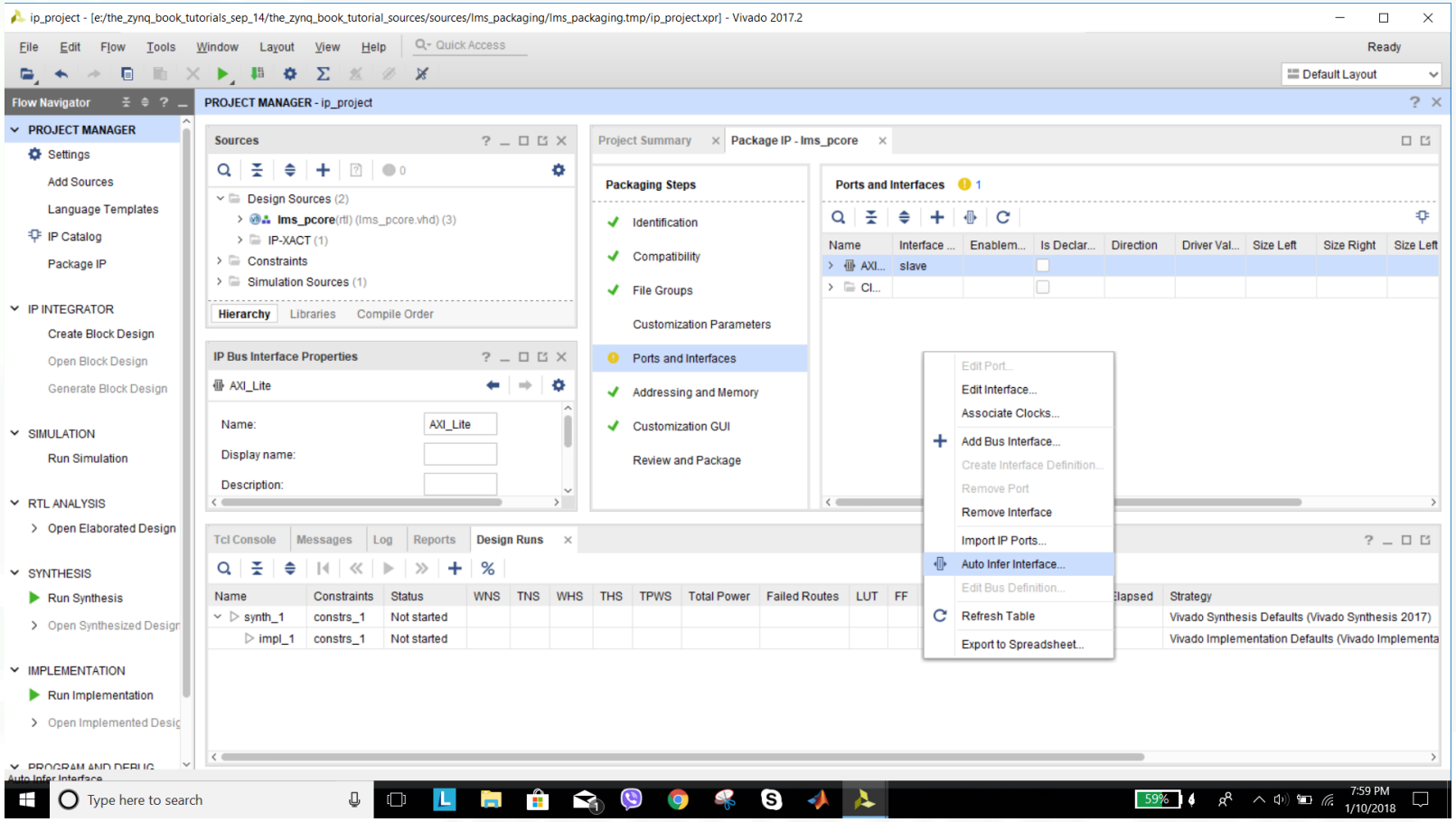


- Click Next to move to Edit in IP Packager Project Name dialogue and click Next to accept the default Project Name and Project Location
- At Summary window, click Finish to launch IP Packager

IP Packaging Window

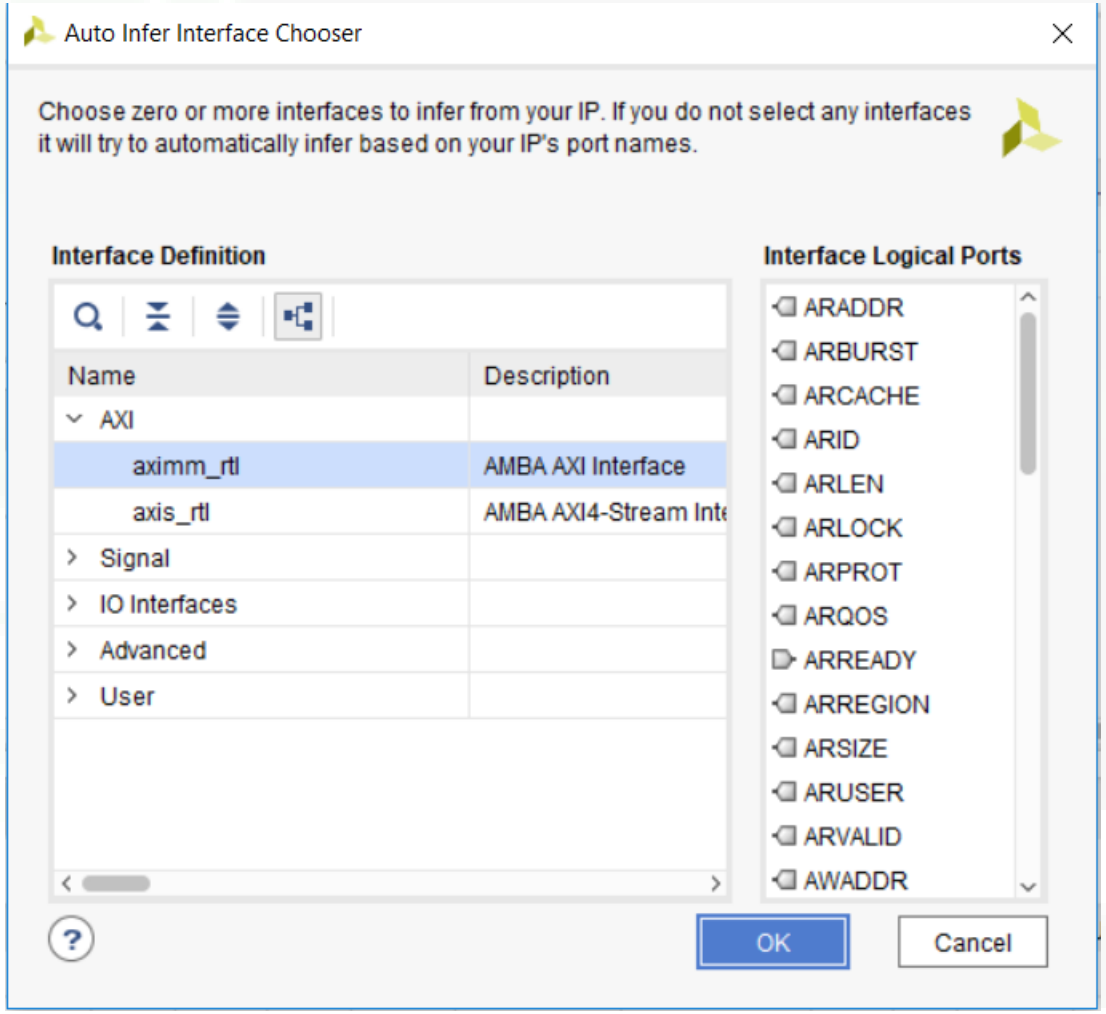


IP Packaging Window



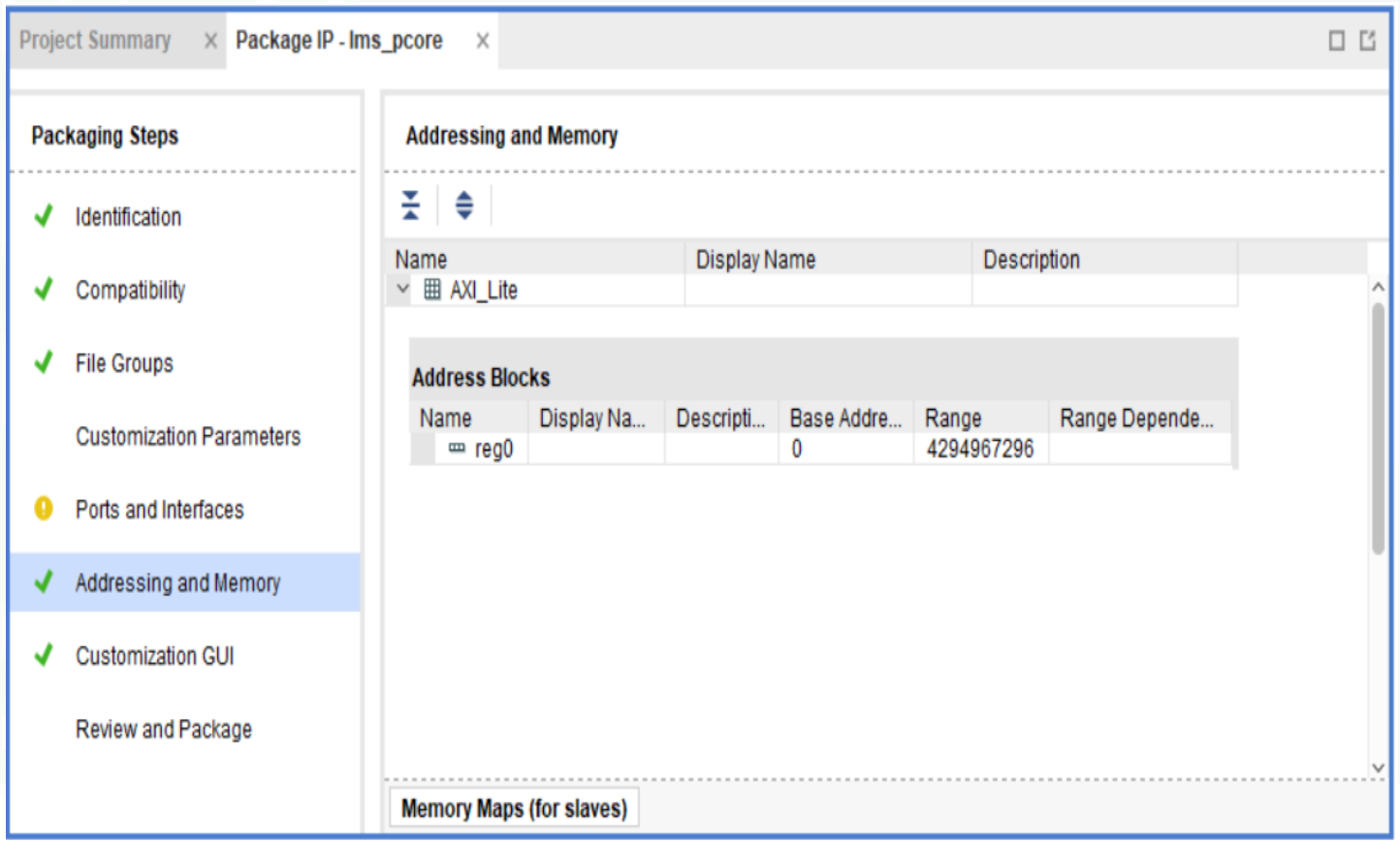
- In the left hand panel of the IP Packager window, select IP Ports and Interfaces.
- The IP Interfaces panel will open, and you should see that IP Packager has identified the individual AXI ports, but has not inferred and AXI Interface
- Right Click on a blank section of IP ports and Interfaces pane, and select Auto Infer Interface.

Auto Infer Interface



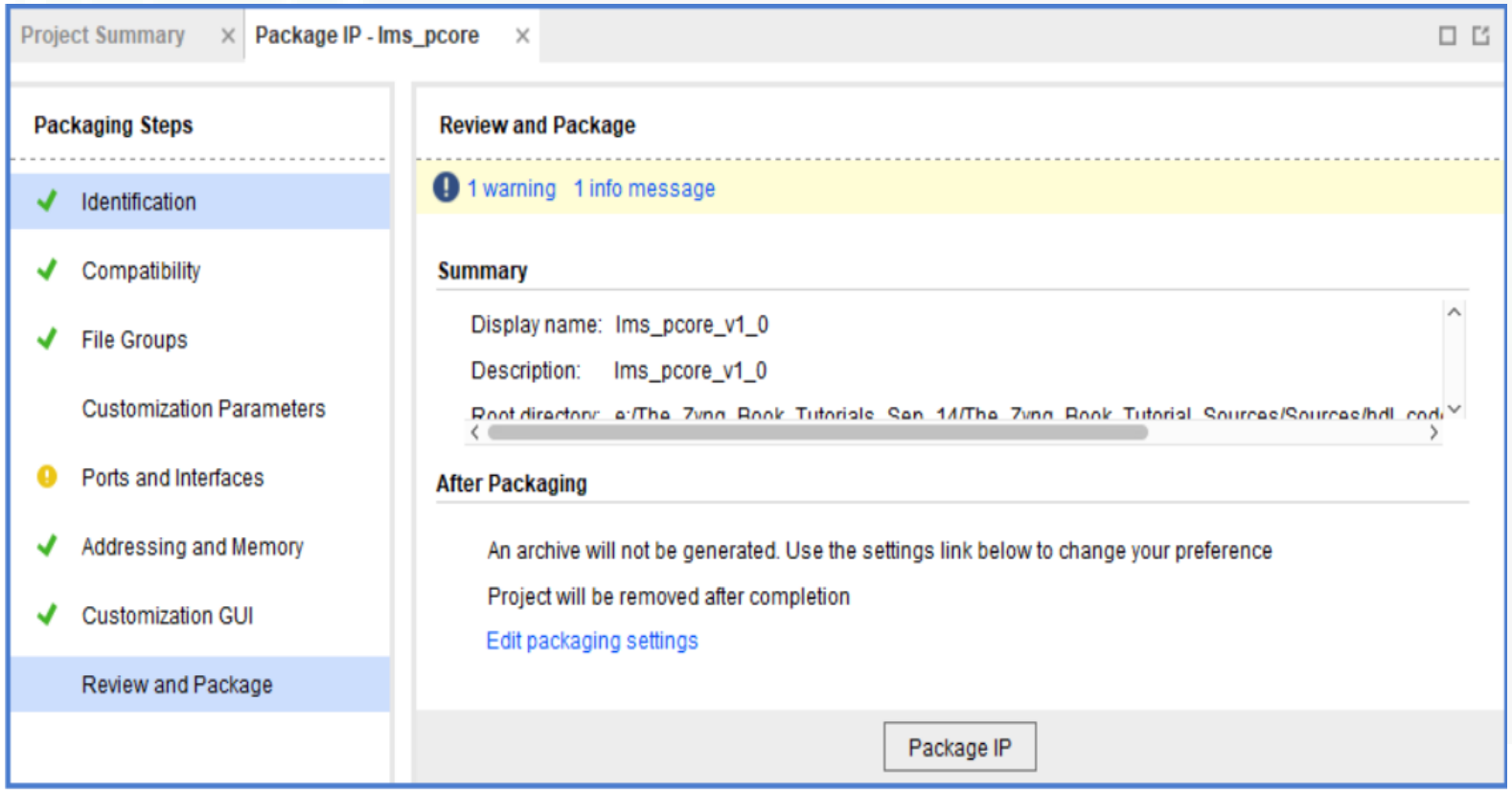
- The Auto Infer Interface Chooser window will open.
- Select aximm_rtl from the list, as shown in the figure and click ok.
- The individual AXI ports in our design will be mapped to an AXILite interface.

IP Addressing and Memory



- Select IP Addressing and Memory from the left hand panel.
- Here IP packager has incorrectly specified an address Range of 4294967296.
- Click on the Range, and change the value to 32

Package IP



- Select Review and Package from the left hand menu.
- Review the information provided, and click Package IP.
- After this you will get a window to close the project, you can click yes.

Import that Packaged IP on VIVADO IP Catalog

i. Goto project settings in VIVADO, go to IP→Repository Manager
ii. Add the repository, where that recent VIVADO IP packaged
iii. Refresh will show the IP on that Menu
iv. Go to IP catalog and search your IP “lms_pcore_v1_0” or you can directly import your IP to Block Design.
v. Now integration of lms_pcore_v1_0 can be done with other master block as Processing System (for Zynq based Design) or Microblaze.

Let's go to MATLAB/Simulink

and then VIVADO

for Lab Session

Thank You!

Innovation from Engineers...