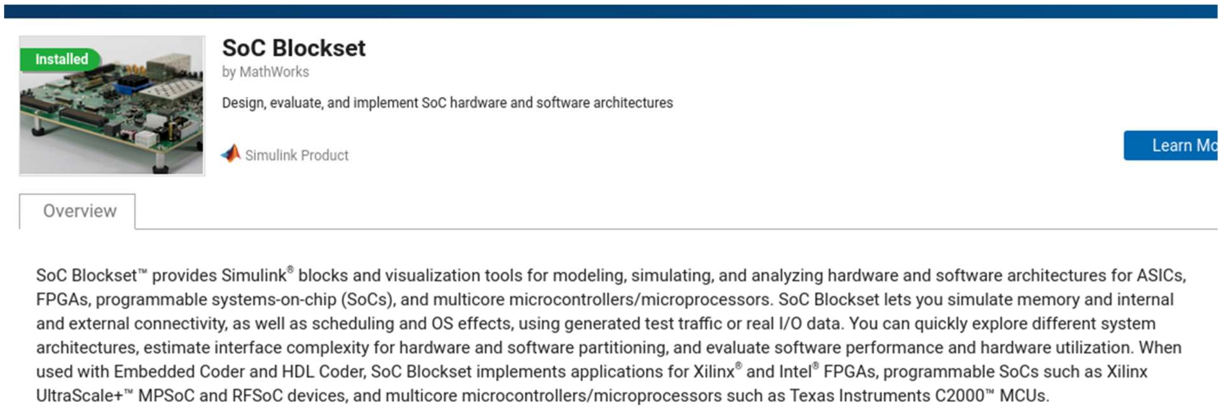
Configuring Custom HDL Models with AXI-Lite Registers using Simulink

Tools and Recommended Versions:

- **Vivado: 2021.1 (Vitis 2021.1** should also be installed)
- **MATLAB: R2021B_U4 or 2022B_U4**

  Make sure that the "SoC Blockset" and "SoC Blockset Support Package for Xilinx Devices" Add-ons are installed.



Installed

SoC Blockset

by MathWorks

Design, evaluate, and implement SoC hardware and software architectures

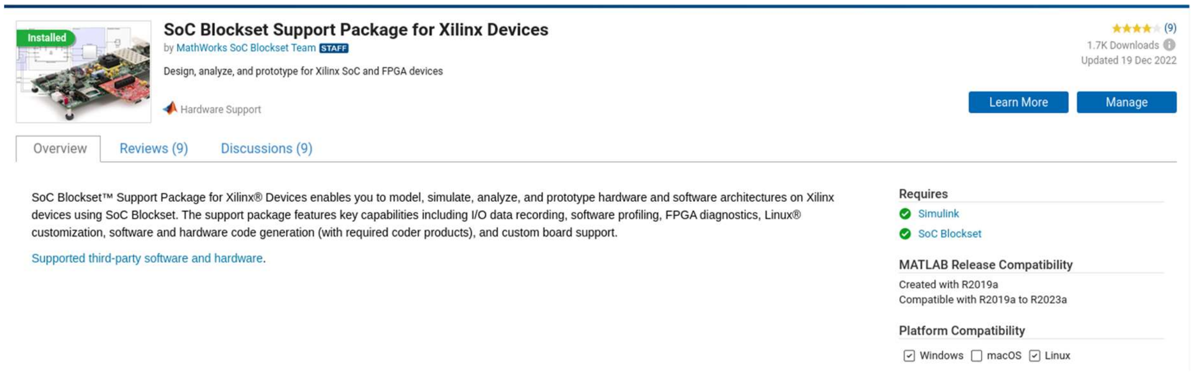
Simulink Product

Learn Mo

Overview

SoC Blockset™ provides Simulink® blocks and visualization tools for modeling, simulating, and analyzing hardware and software architectures for ASICs, FPGAs, programmable systems-on-chip (SoCs), and multicore microcontrollers/microprocessors. SoC Blockset lets you simulate memory and internal and external connectivity, as well as scheduling and OS effects, using generated test traffic or real I/O data. You can quickly explore different system architectures, estimate interface complexity for hardware and software partitioning, and evaluate software performance and hardware utilization. When used with Embedded Coder and HDL Coder, SoC Blockset implements applications for Xilinx® and Intel® FPGAs, programmable SoCs such as Xilinx UltraScale+™ MPSoC and RFSoC devices, and multicore microcontrollers/microprocessors such as Texas Instruments C2000™ MCUs.

Figure 1. SoC Blockset Add-On



Installed

SoC Blockset Support Package for Xilinx Devices

by MathWorks SoC Blockset Team STAFF

Design, analyze, and prototype for Xilinx SoC and FPGA devices

Hardware Support

★★★★ (9)

1.7K Downloads

Updated 19 Dec 2022

Learn More

Manage

Overview · Reviews (9) · Discussions (9)

SoC Blockset™ Support Package for Xilinx® Devices enables you to model, simulate, analyze, and prototype hardware and software architectures on Xilinx devices using SoC Blockset. The support package features key capabilities including I/O data recording, software profiling, FPGA diagnostics, Linux® customization, software and hardware code generation (with required coder products), and custom board support.

Supported third-party software and hardware.

Requires

✓ Simulink

✓ SoC Blockset

MATLAB Release Compatibility

Created with R2019a

Compatible with R2019a to R2023a

Platform Compatibility

☑ Windows ☐ macOS ☑ Linux

Figure 2. SoC Blockset Support Package for Xilinx Devices Add-On

- Recommended terminal for Windows: Cygwin (https://cygwin.com)

Instructions to Build the Toolbox from Terminal:

```
>git clone https://github.com/analogdevicesinc/HighSpeedConverterToolbox.git

>cd HighSpeedConverterToolbox

>git submodule update --init --recursive
```

>git checkout lldk_axi4lite_examples

>cd CI/scripts

>make build HDLBRANCH=dev_lldk_rebase

>export ADI_IGNORE_VERSION_CHECK=TRUE

Launch Matlab from the root of HighSpeedConverterToolbox folder .

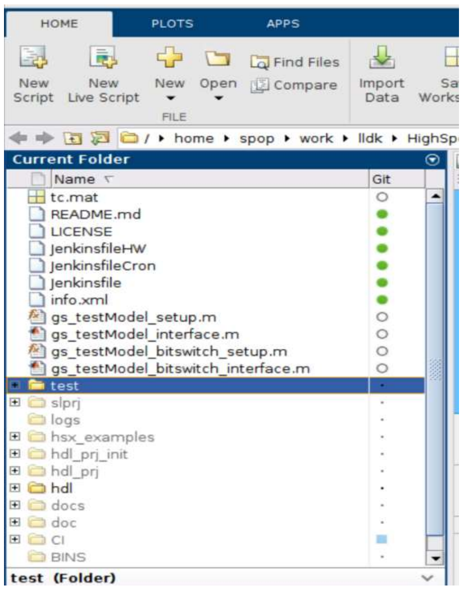


Figure 3. High Speed Converter Toolbox Sources

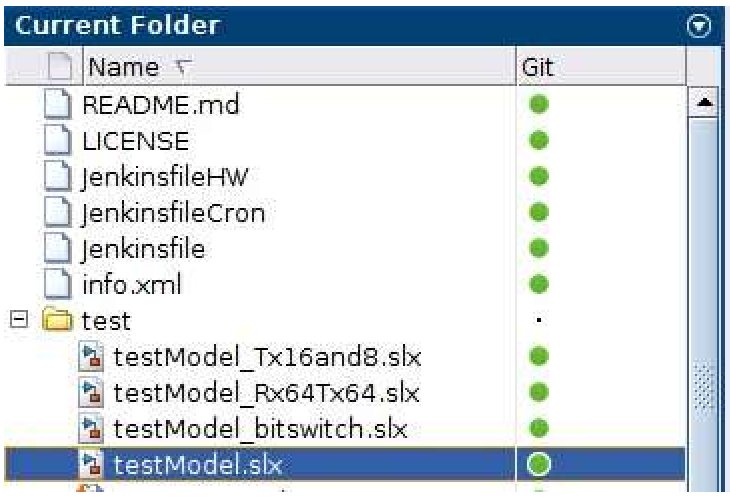
right click on test -> Add to Path -> Selected folders and subfolders

right click on hdl -> Add to Path -> Selected folders and subfolders

In the Matlab command window set the path to Vivaldo installation folder. The tool path should be replaced with the user's Vivado path.

e.g. **hdlsetuptoolpath('ToolName', 'Xilinx Vivado', 'ToolPath', '</opt/Xilinx/Vivado/2021.1/bin/vivado>')**

Expand the test folder and double click on the desired Simulink test model, as shown in Figure 4.



- README.md ●
- LICENSE ●
- JenkinsfileHW ●
- JenkinsfileCron ●
- Jenkinsfile ●
- info.xml ●
- test
  - testModel_Tx16and8.slx ●
  - testModel_Rx64Tx64.slx ●
  - testModel_bitswitch.slx ●
  - testModel.slx ●

Figure 4. Simulink Test Model

After opening the Simulink model, right click on the HDL_DUT and launch the HDL Workflow Advisor as shown in Figure 5, and Figure 6.

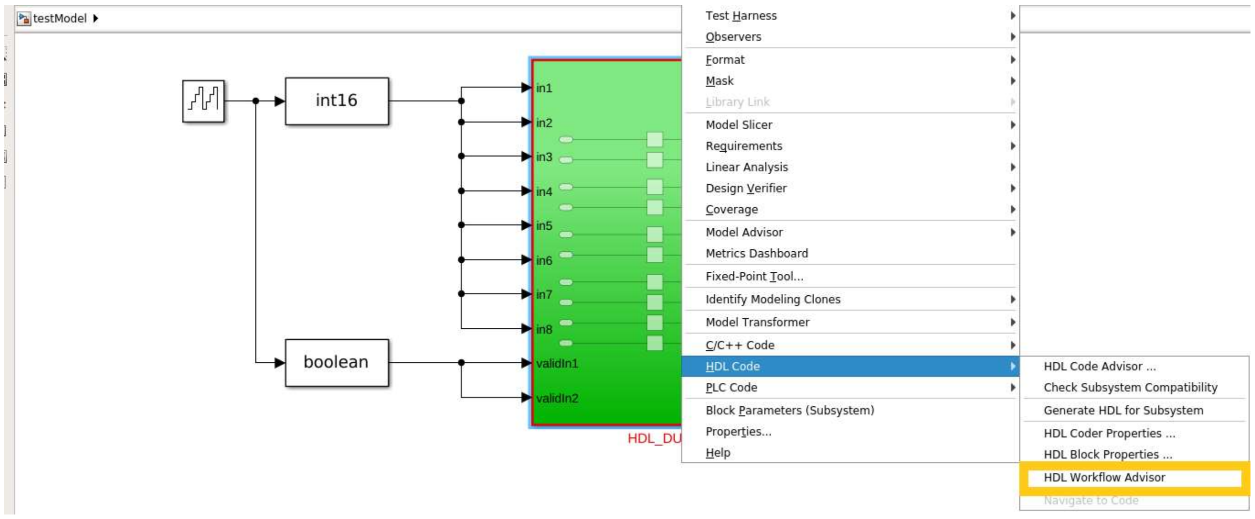


Figure 5. HDL Workflow Advisor Launching

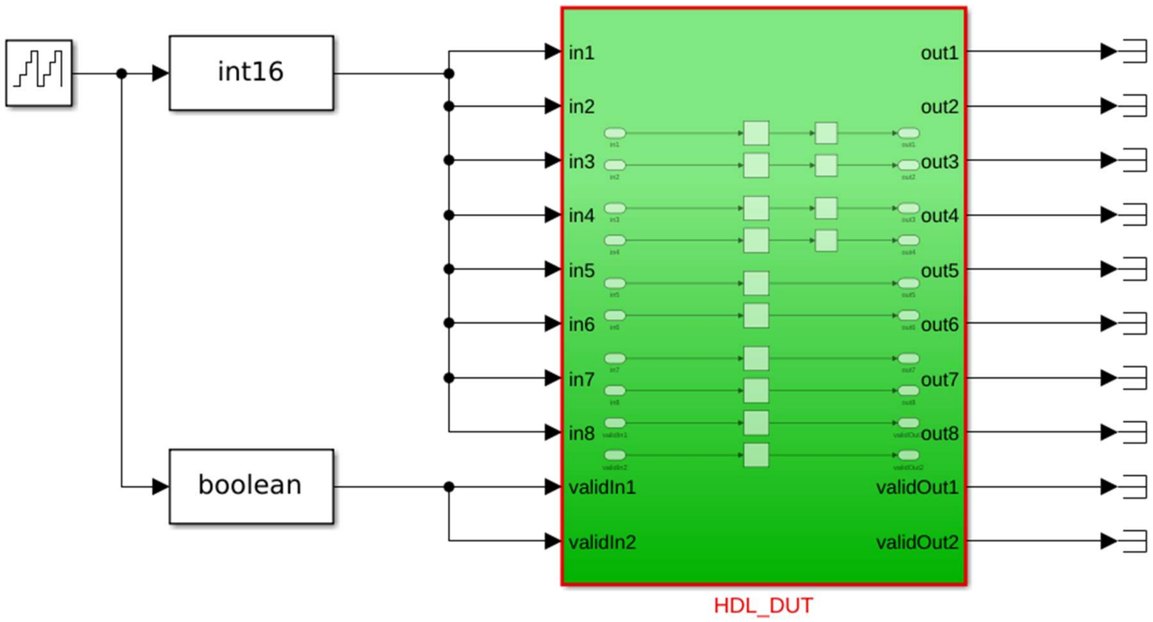
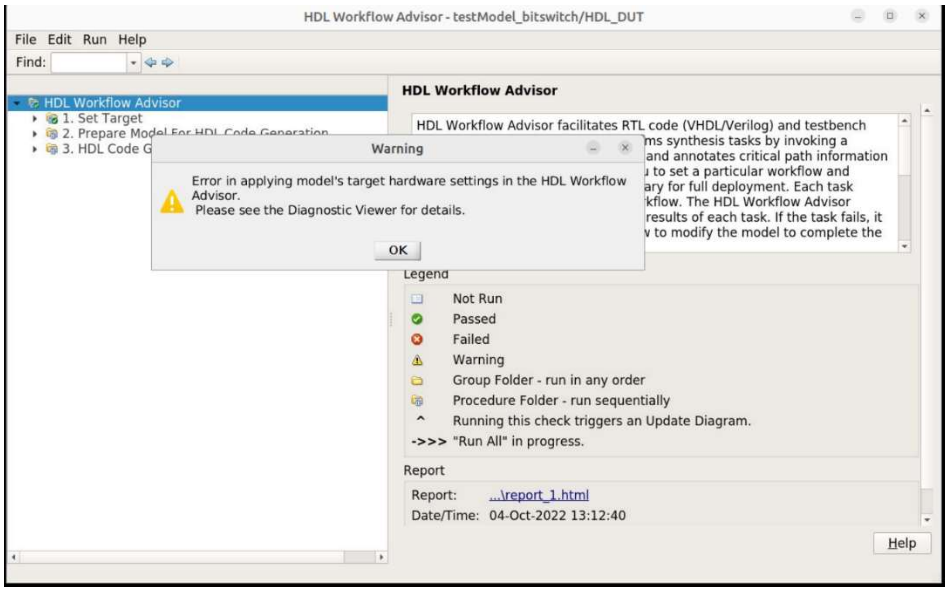


Figure 6. Simulink Device Under Test

Close this expected warning that will appear, as shown in Figure 7.


HDL Workflow Advisor - testModel_bitswitch/HDL_DUT
File Edit Run Help
Find:
HDL Workflow Advisor
1. Set Target
2. Prepare Model For HDL Code Generation
3. HDL Code G
Legend
Not Run
Passed
Failed
Warning
Group Folder - run in any order
Procedure Folder - run sequentially
Running this check triggers an Update Diagram.
->>> "Run All" in progress.
Report
Help

Warning

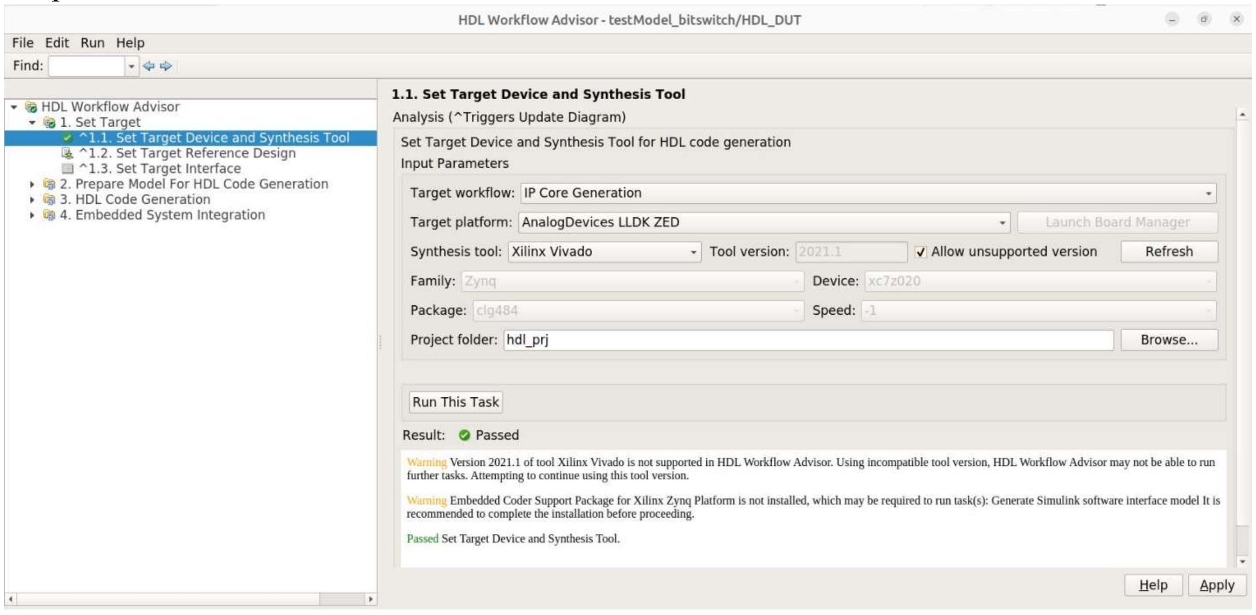
Error in applying model's target hardware settings in the HDL Workflow Advisor.
Please see the Diagnostic Viewer for details.

OK

Report: ...\report_1.html
Date/Time: 04-Oct-2022 13:12:40

Figure 7. Expected HDL Workflow Advisor Warning

1.1 Select IP Core Generation, choose the desired project and carrier from the dropdown list and check the Allow unsupported version box. Change the project folder name if desired. Finally press the Run this Task button.


HDL Workflow Advisor - testModel_bitswitch/HDL_DUT
File Edit Run Help
Find:
HDL Workflow Advisor
1. Set Target
^1.1. Set Target Device and Synthesis Tool
^1.2. Set Target Reference Design
^1.3. Set Target Interface
2. Prepare Model For HDL Code Generation
3. HDL Code Generation
4. Embedded System Integration
1.1. Set Target Device and Synthesis Tool
Analysis (^Triggers Update Diagram)
Set Target Device and Synthesis Tool for HDL code generation
Input Parameters
Launch Board Manager
Refresh
Browse...
Run This Task
Help
Apply

Target workflow: IP Core Generation
Target platform: AnalogDevices LLDK ZED
Synthesis tool: Xilinx Vivado
Tool version: 2021.1
✓ Allow unsupported version
Family: Zynq
Device: xc7z020
Package: clg484
Speed: -1
Project folder: hdl_prj

Result: Passed

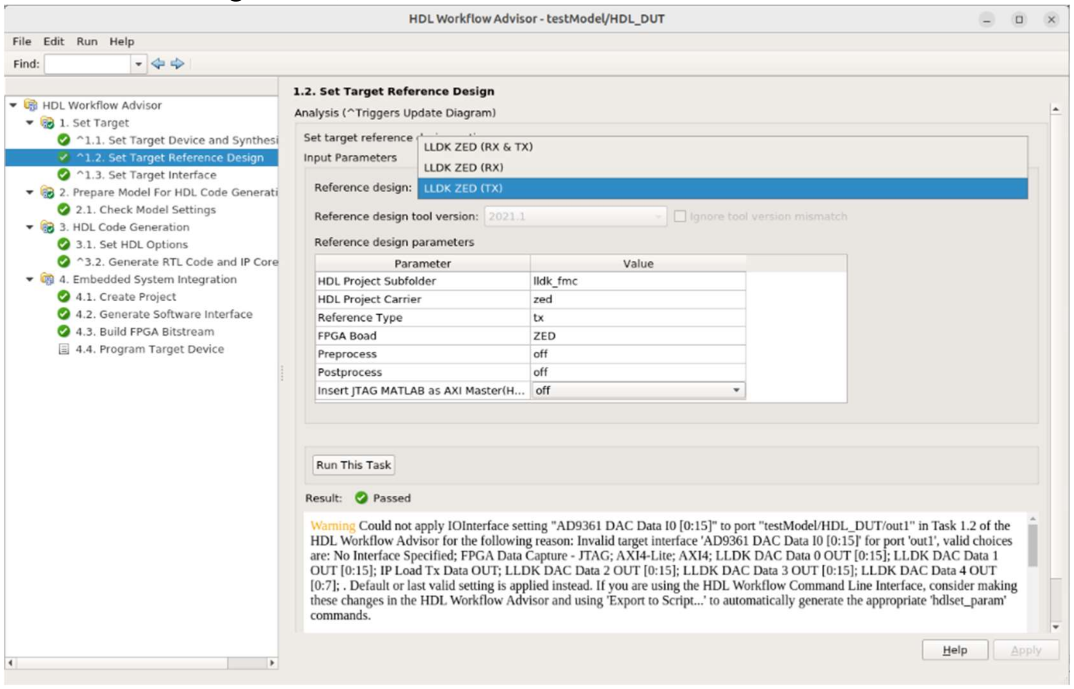
Warning Version 2021.1 of tool Xilinx Vivado is not supported in HDL Workflow Advisor. Using incompatible tool version, HDL Workflow Advisor may not be able to run further tasks. Attempting to continue using this tool version.

Warning Embedded Coder Support Package for Xilinx Zynq Platform is not installed, which may be required to run task(s): Generate Simulink software interface model It is recommended to complete the installation before proceeding.

Passed Set Target Device and Synthesis Tool.

Figure 8. Set Target Device and Synthesis Tool

1.2 Choose the TX configuration, then run the task.



HDL Workflow Advisor - testModel/HDL_DUT
File Edit Run Help
Find:
HDL Workflow Advisor
1. Set Target
^1.1. Set Target Device and Synthesi
^1.2. Set Target Reference Design
^1.3. Set Target Interface
2. Prepare Model For HDL Code Generati
2.1. Check Model Settings
3. HDL Code Generation
3.1. Set HDL Options
^3.2. Generate RTL Code and IP Core
4. Embedded System Integration
4.1. Create Project
4.2. Generate Software Interface
4.3. Build FPGA Bitstream
4.4. Program Target Device
Run This Task
Help
Apply

1.2. Set Target Reference Design

Analysis (^Triggers Update Diagram)

Set target reference

Input Parameters

LLDK ZED (RX & TX)

LLDK ZED (RX)

Reference design: LLDK ZED (TX)

Reference design tool version: 2021.1 ☐ Ignore tool version mismatch

Reference design parameters

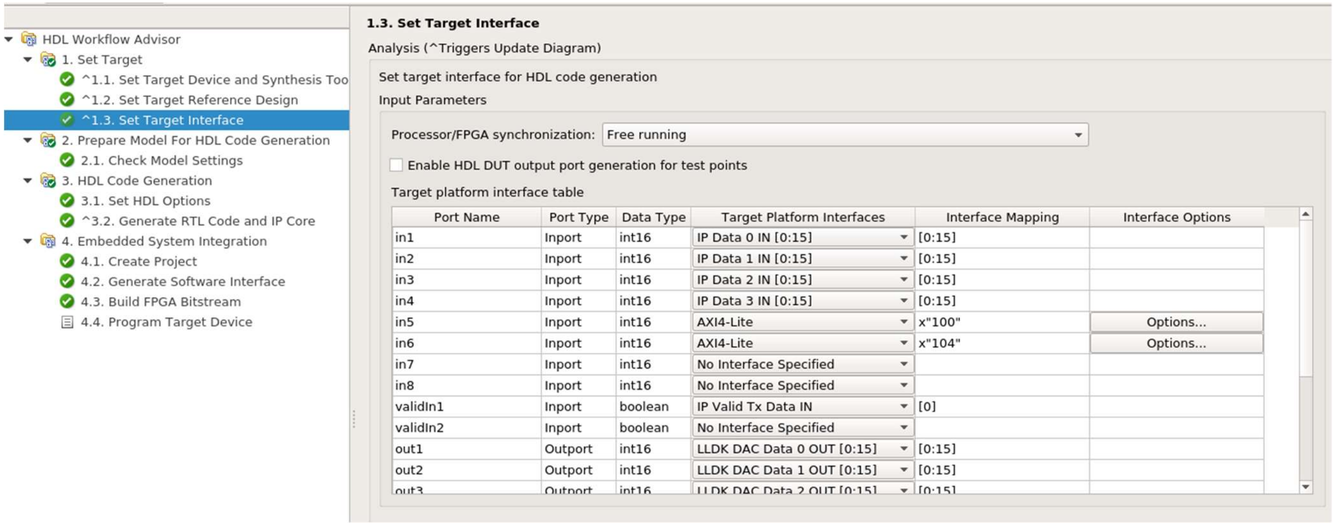
Parameter	Value
HDL Project Subfolder	lldk_fmc
HDL Project Carrier	zed
Reference Type	tx
FPGA Boad	ZED
Preprocess	off
Postprocess	off
Insert JTAG MATLAB as AXI Master(H...	off

Result: ✓ Passed

Warning Could not apply IOInterface setting "AD9361 DAC Data I0 [0:15]" to port "testModel/HDL_DUT/out1" in Task 1.2 of the HDL Workflow Advisor for the following reason: Invalid target interface 'AD9361 DAC Data I0 [0:15]' for port 'out1', valid choices are: No Interface Specified; FPGA Data Capture - JTAG; AXI4-Lite; AXI4; LLDK DAC Data 0 OUT [0:15]; LLDK DAC Data 1 OUT [0:15]; IP Load Tx Data OUT; LLDK DAC Data 2 OUT [0:15]; LLDK DAC Data 3 OUT [0:15]; LLDK DAC Data 4 OUT [0:7]; . Default or last valid setting is applied instead. If you are using the HDL Workflow Command Line Interface, consider making these changes in the HDL Workflow Advisor and using 'Export to Script...' to automatically generate the appropriate 'hdlset_param' commands.

Figure 9. Set Target Reference Design

1.3 Assign the data ports as described in Figure 9 and Figure 10, add as many Input/Output registers as you need. Table 1 shows port descriptions for HDL DUT Tx Reference Design.



HDL Workflow Advisor
1. Set Target
^1.1. Set Target Device and Synthesis Too
^1.2. Set Target Reference Design
^1.3. Set Target Interface
2. Prepare Model For HDL Code Generation
2.1. Check Model Settings
3. HDL Code Generation
3.1. Set HDL Options
^3.2. Generate RTL Code and IP Core
4. Embedded System Integration
4.1. Create Project
4.2. Generate Software Interface
4.3. Build FPGA Bitstream
4.4. Program Target Device

1.3. Set Target Interface

Analysis (^Triggers Update Diagram)

Set target interface for HDL code generation

Input Parameters

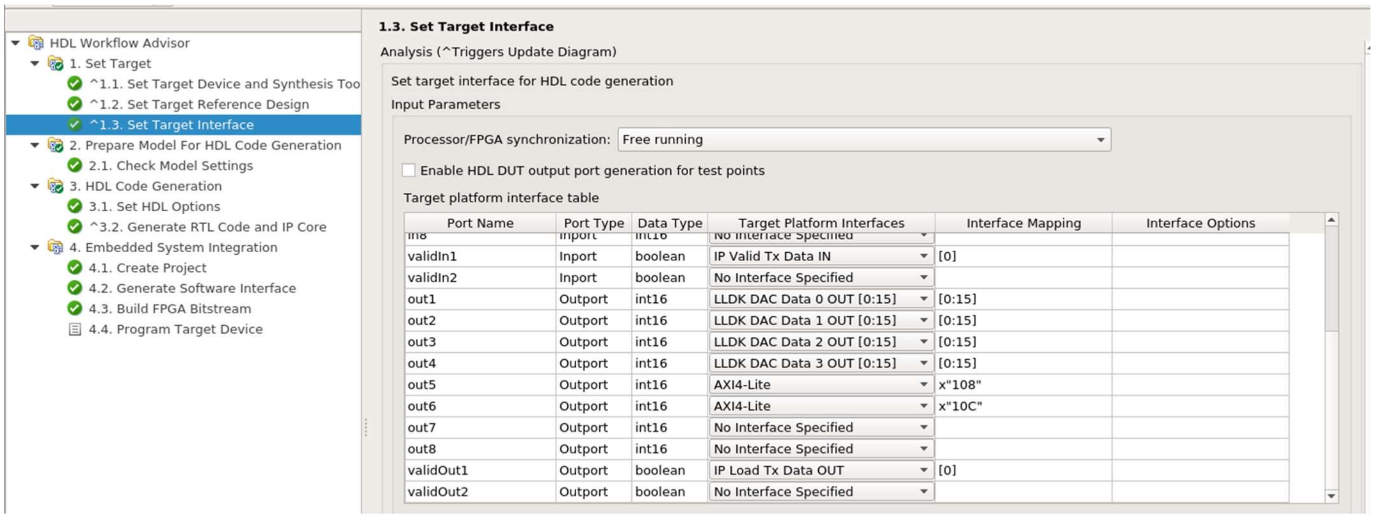
Processor/FPGA synchronization: Free running

☐ Enable HDL DUT output port generation for test points

Target platform interface table

Port Name	Port Type	Data Type	Target Platform Interfaces	Interface Mapping	Interface Options
in1	Inport	int16	IP Data 0 IN [0:15]	[0:15]	
in2	Inport	int16	IP Data 1 IN [0:15]	[0:15]	
in3	Inport	int16	IP Data 2 IN [0:15]	[0:15]	
in4	Inport	int16	IP Data 3 IN [0:15]	[0:15]	
in5	Inport	int16	AXI4-Lite	x"100"	Options...
in6	Inport	int16	AXI4-Lite	x"104"	Options...
in7	Inport	int16	No Interface Specified		
in8	Inport	int16	No Interface Specified		
validIn1	Inport	boolean	IP Valid Tx Data IN	[0]	
validIn2	Inport	boolean	No Interface Specified		
out1	Outport	int16	LLDK DAC Data 0 OUT [0:15]	[0:15]	
out2	Outport	int16	LLDK DAC Data 1 OUT [0:15]	[0:15]	
out3	Outport	int16	LLDK DAC Data 2 OUT [0:15]	[0:15]	

Figure 10. Set Input Target Interface



- HDL Workflow Advisor
  - 1. Set Target
    - ^1.1. Set Target Device and Synthesis Too
    - ^1.2. Set Target Reference Design
    - ^1.3. Set Target Interface
  - 2. Prepare Model For HDL Code Generation
    - 2.1. Check Model Settings
  - 3. HDL Code Generation
    - 3.1. Set HDL Options
    - ^3.2. Generate RTL Code and IP Core
  - 4. Embedded System Integration
    - 4.1. Create Project
    - 4.2. Generate Software Interface
    - 4.3. Build FPGA Bitstream
    - 4.4. Program Target Device

1.3. Set Target Interface

Analysis (^Triggers Update Diagram)

Set target interface for HDL code generation

Input Parameters

Processor/FPGA synchronization: Free running

☐ Enable HDL DUT output port generation for test points

Target platform interface table

Port Name	Port Type	Data Type	Target Platform Interfaces	Interface Mapping	Interface Options
in8	Inport	int16	No Interface Specified		
validIn1	Inport	boolean	IP Valid Tx Data IN	[0]	
validIn2	Inport	boolean	No Interface Specified		
out1	Outport	int16	LLDK DAC Data 0 OUT [0:15]	[0:15]	
out2	Outport	int16	LLDK DAC Data 1 OUT [0:15]	[0:15]	
out3	Outport	int16	LLDK DAC Data 2 OUT [0:15]	[0:15]	
out4	Outport	int16	LLDK DAC Data 3 OUT [0:15]	[0:15]	
out5	Outport	int16	AXI4-Lite	x"108"	
out6	Outport	int16	AXI4-Lite	x"10C"	
out7	Outport	int16	No Interface Specified		
out8	Outport	int16	No Interface Specified		
validOut1	Outport	boolean	IP Load Tx Data OUT	[0]	
validOut2	Outport	boolean	No Interface Specified		

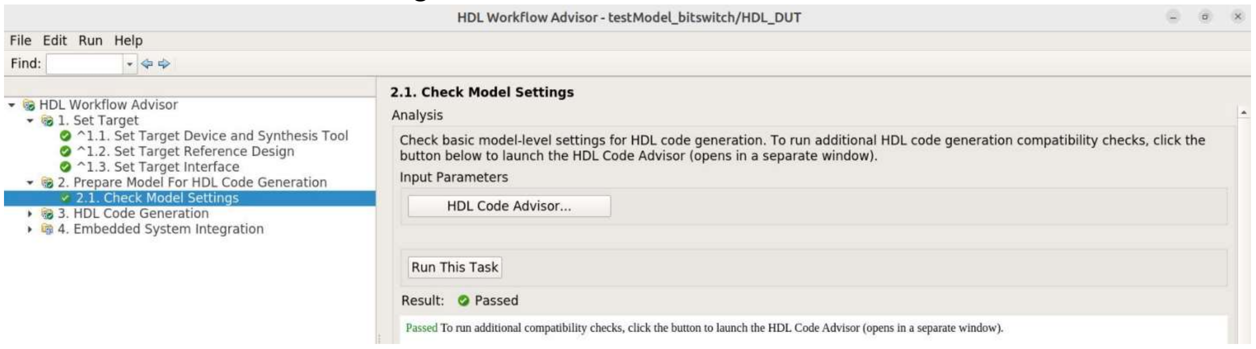
Figure 11. Set Output Target Interface

Interface signal name	**Width**	**Description**
IP Data 0 IN	16	Custom IP data input signal. This signal is connected to the AXI_LTC2387_0 IP ADC DATA port in the ADI reference design.
IP Data 1 IN	16	Custom IP data input signal. This signal is connected to the AXI_LTC2387_1 IP ADC DATA port in the ADI reference design.
IP Data 2 IN	16	Custom IP data input signal. This signal is connected to the AXI_LTC2387_2 IP ADC DATA port in the ADI reference design.
IP Data 3 IN	16	Custom IP data input signal. This signal is connected to the AXI_LTC2387_3 IP ADC DATA port in the ADI reference design.
IP Valid Tx Data IN	1	Input signal that has logic 1 value for a clock cycle period when the data starts to be valid.
CN0585 DAC Data 0 OUT	16	AD3552R_0 DAC 0 channel data. To be used as input into the AD3552R interface IP.
CN0585 DAC Data 1 OUT	16	AD3552R_0 DAC 1 channel data. To be used as input into the AD3552R interface IP.
CN0585 DAC Data 2 OUT	16	AD3552R_1 DAC 0 channel data. To be used as input into the AD3552R interface IP.
CN0585 DAC Data 3 OUT	16	AD3552R_1 DAC 1 channel data. To be used as input into the AD3552R interface IP.
IP Load Tx Data OUT	1	Custom IP output signal used to notify the design that the IP is ready to receive new input data. Output signal that has to be logic '1' for a clock cycle period when the data starts to be valid.

Table 1: HDL DUT Ports for Transmit Reference Design (Tx)

AXI registers are defined in the Simulink model as input or output ports (AXI-lite option is selected in “Target Platform Interfaces” column. Register addresses are set in “Interface Mapping” column and written like x”<number of bits that can be used, hex address>”.) AXI registers that are input ports are write-only, and AXI registers that are output ports are read-only. If you connect those two together in the model, you now have a read-only register connected to the write-only register so it is readable, but at a different address.

1.4 Run the task, as shown in Figure 12.


HDL Workflow Advisor - testModel_bitswitch/HDL_DUT

- HDL Workflow Advisor
  - 1. Set Target
    - ^1.1. Set Target Device and Synthesis Tool
    - ^1.2. Set Target Reference Design
    - ^1.3. Set Target Interface
  - 2. Prepare Model For HDL Code Generation
    - 2.1. Check Model Settings
  - 3. HDL Code Generation
  - 4. Embedded System Integration

2.1. Check Model Settings

Analysis

Check basic model-level settings for HDL code generation. To run additional HDL code generation compatibility checks, click the button below to launch the HDL Code Advisor (opens in a separate window).

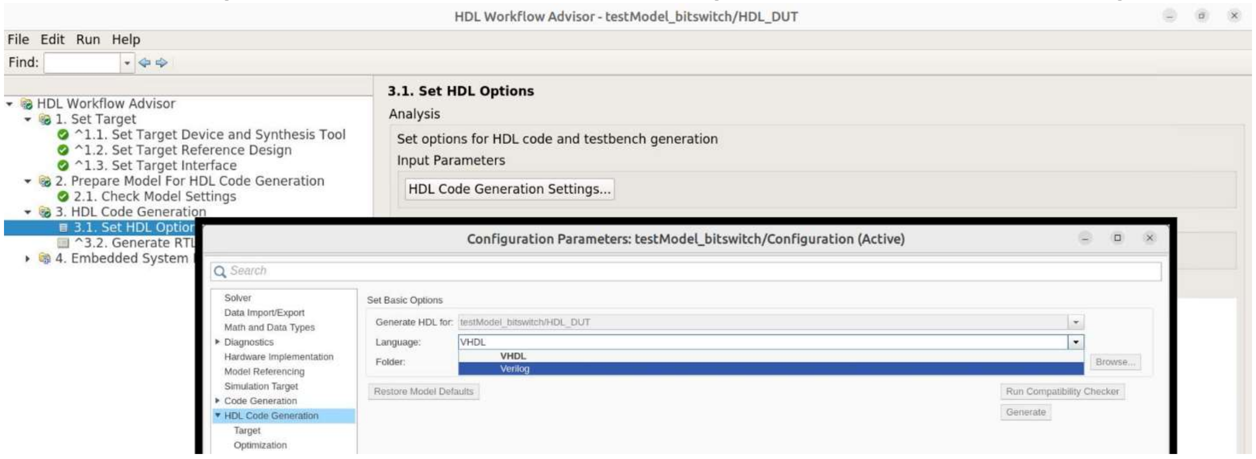
Input Parameters

Result: Passed

Passed To run additional compatibility checks, click the button to launch the HDL Code Advisor (opens in a separate window).

Figure 12. Check Model Settings

2.1 Select Verilog for the HDL Code Generation Settings, then run task as shown in Figure 13.


HDL Workflow Advisor - testModel_bitswitch/HDL_DUT

- HDL Workflow Advisor
  - 1. Set Target
    - ^1.1. Set Target Device and Synthesis Tool
    - ^1.2. Set Target Reference Design
    - ^1.3. Set Target Interface
  - 2. Prepare Model For HDL Code Generation
    - 2.1. Check Model Settings
  - 3. HDL Code Generation
    - 3.1. Set HDL Option
    - ^3.2. Generate RTL
  - 4. Embedded System

3.1. Set HDL Options

Analysis

Set options for HDL code and testbench generation

Input Parameters

Configuration Parameters: testModel_bitswitch/Configuration (Active)

Set Basic Options

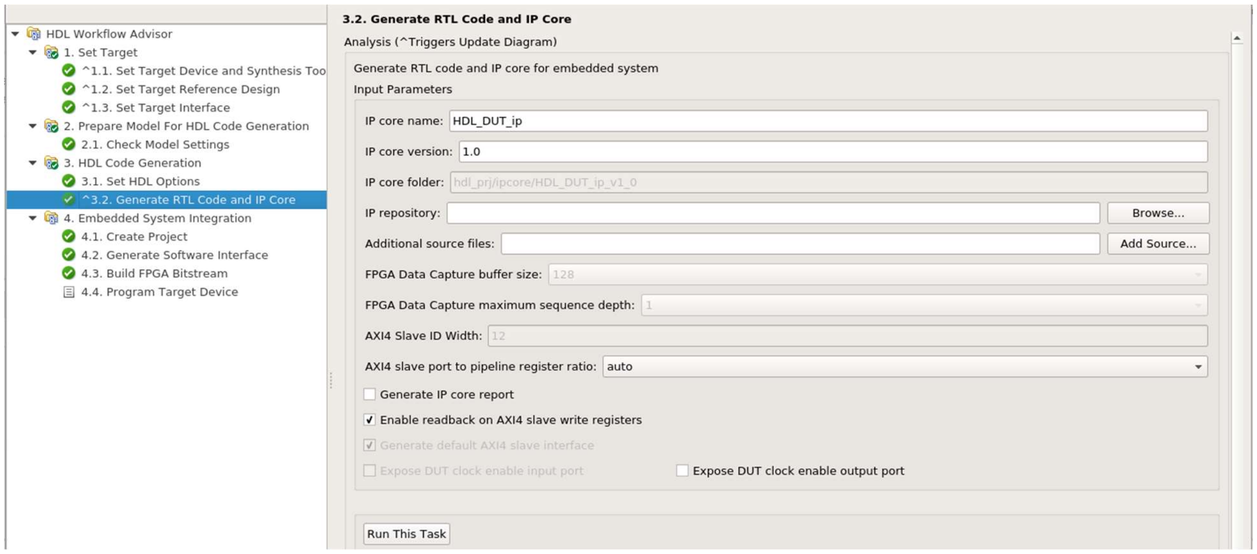
Generate HDL for: testModel_bitswitch/HDL_DUT

Language: VHDL

Folder:

Figure 13. Set HDL Options

2.2 Check the Enable readback on AXI4 slave write registers as described in Figure 14.



3.2. Generate RTL Code and IP Core

Analysis (^Triggers Update Diagram)

Generate RTL code and IP core for embedded system

Input Parameters

IP core name: HDL_DUT_ip

IP core version: 1.0

IP core folder: hdl_prj/ipcore/HDL_DUT_ip_v1_0

IP repository:

Additional source files:

FPGA Data Capture buffer size: 128

FPGA Data Capture maximum sequence depth: 1

AXI4 Slave ID Width: 12

AXI4 slave port to pipeline register ratio: auto

☐ Generate IP core report

☑ Enable readback on AXI4 slave write registers

☑ Generate default AXI4 slave interface

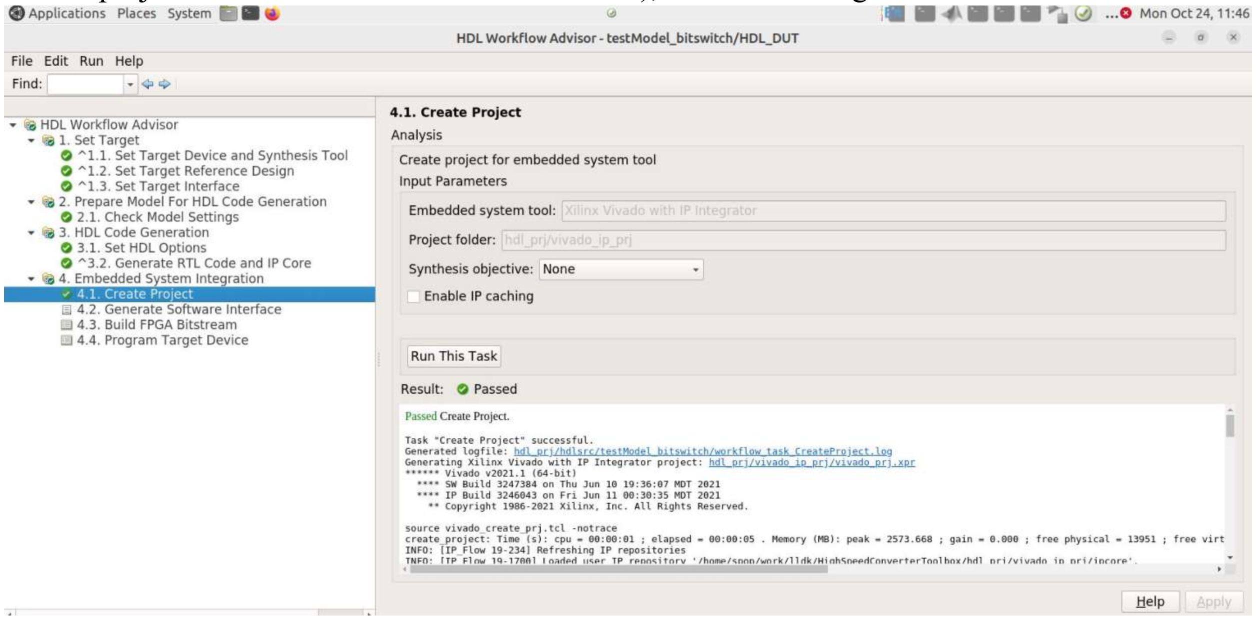
☐ Expose DUT clock enable input port

☐ Expose DUT clock enable output port

HDL Workflow Advisor
1. Set Target
^1.1. Set Target Device and Synthesis Too
^1.2. Set Target Reference Design
^1.3. Set Target Interface
2. Prepare Model For HDL Code Generation
2.1. Check Model Settings
3. HDL Code Generation
3.1. Set HDL Options
^3.2. Generate RTL Code and IP Core
4. Embedded System Integration
4.1. Create Project
4.2. Generate Software Interface
4.3. Build FPGA Bitstream
4.4. Program Target Device
Browse...
Add Source...
Run This Task

Figure 14. Generate RTL code and IP Core

3.1 Run the task (this will create the Vivado block design in the hdl_prj/vivado_ip_prj folder, or the project folder name that was chosen in 1.1), as shown in Figure 15.



HDL Workflow Advisor - testModel_bitswitch/HDL_DUT

4.1. Create Project

Analysis

Create project for embedded system tool

Input Parameters

Embedded system tool: Xilinx Vivado with IP Integrator

Project folder: hdl_prj/vivado_ip_prj

Synthesis objective: None

☐ Enable IP caching

Result: Passed

Passed Create Project.

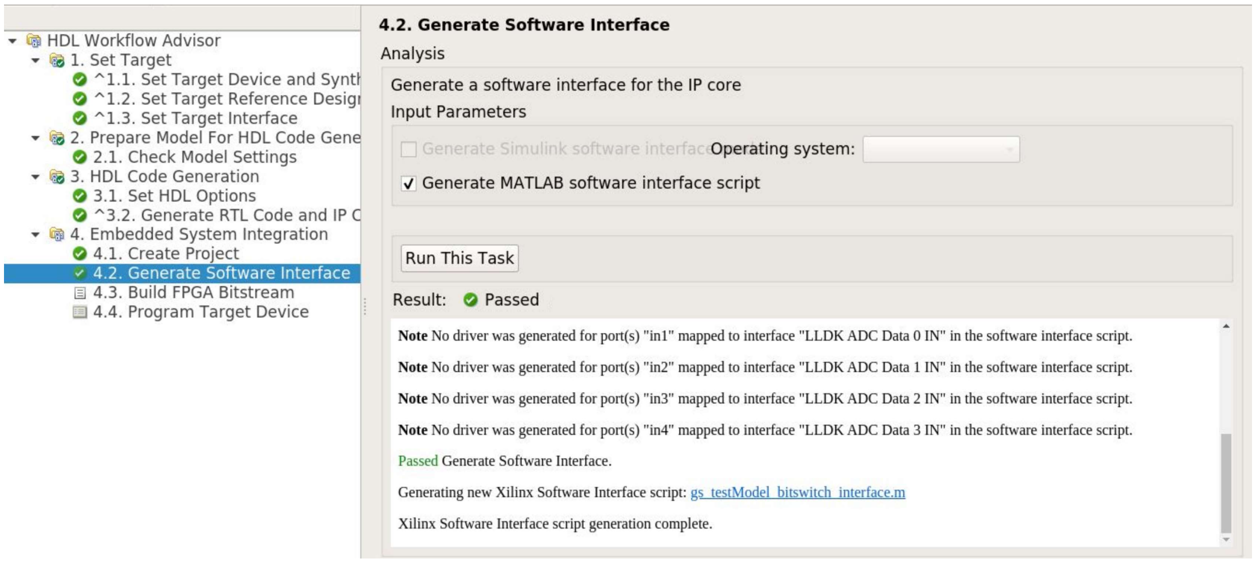
```
Task "Create Project" successful.
Generated logfile: hdl_prj/hdlsrc/testModel_bitswitch/workflow_task_CreateProject.log
Generating Xilinx Vivado with IP Integrator project: hdl_prj/vivado_ip_prj/vivado_prj.xpr
****** Vivado v2021.1 (64-bit)
  **** SW Build 3247384 on Thu Jun 10 19:36:07 MDT 2021
  **** IP Build 3246043 on Fri Jun 11 00:30:35 MDT 2021
    ** Copyright 1986-2021 Xilinx, Inc. All Rights Reserved.

source vivado_create_prj.tcl -notrace
create_project: Time (s): cpu = 00:00:01 ; elapsed = 00:00:05 . Memory (MB): peak = 2573.668 ; gain = 0.000 ; free physical = 13951 ; free virt
INFO: [IP_Flow 19-234] Refreshing IP repositories
INFO: [IP_Flow 19-1700] Loaded user IP repository '/home/spon/work/lldk/HighSpeedConverterToolbox/hdl_prj/vivado_ip_prj/ipcore'.
```

Applications
Places
System
Mon Oct 24, 11:46
File
Edit
Run
Help
Find:
HDL Workflow Advisor
1. Set Target
^1.1. Set Target Device and Synthesis Tool
^1.2. Set Target Reference Design
^1.3. Set Target Interface
2. Prepare Model For HDL Code Generation
2.1. Check Model Settings
3. HDL Code Generation
3.1. Set HDL Options
^3.2. Generate RTL Code and IP Core
4. Embedded System Integration
4.1. Create Project
4.2. Generate Software Interface
4.3. Build FPGA Bitstream
4.4. Program Target Device
Run This Task
Help
Apply

Figure 15. Create Project

3.2 Run the task in Figure 16.


HDL Workflow Advisor
1. Set Target
^1.1. Set Target Device and Synth
^1.2. Set Target Reference Design
^1.3. Set Target Interface
2. Prepare Model For HDL Code Gene
2.1. Check Model Settings
3. HDL Code Generation
3.1. Set HDL Options
^3.2. Generate RTL Code and IP C
4. Embedded System Integration
4.1. Create Project
4.2. Generate Software Interface
4.3. Build FPGA Bitstream
4.4. Program Target Device
4.2. Generate Software Interface
Analysis
Generate a software interface for the IP core
Input Parameters
Generate Simulink software interface
Operating system:
Run This Task

✓ Generate MATLAB software interface script

Result: Passed

Note No driver was generated for port(s) "in1" mapped to interface "LLDK ADC Data 0 IN" in the software interface script.

Note No driver was generated for port(s) "in2" mapped to interface "LLDK ADC Data 1 IN" in the software interface script.

Note No driver was generated for port(s) "in3" mapped to interface "LLDK ADC Data 2 IN" in the software interface script.

Note No driver was generated for port(s) "in4" mapped to interface "LLDK ADC Data 3 IN" in the software interface script.

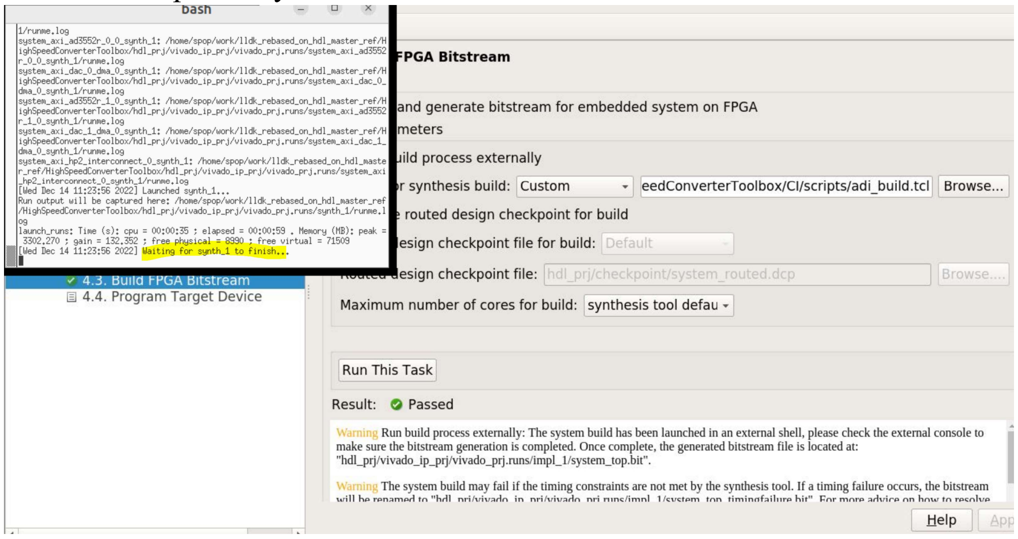
Passed Generate Software Interface.

Generating new Xilinx Software Interface script: gs_testModel_bitswitch_interface.m

Xilinx Software Interface script generation complete.

Figure 16. Generate Software Interface

3.3 Choose the "Custom" option for the Tcl file synthesis build, then Browse for the adi_build.tcl file located under HighSpeedConverterToolbox/CI/scripts, as shown in Figure 17. A bash prompt will open, and you can see the entire build process log file, as shown in Figure 17 and Figure 18. This step usually takes about an hour.


bash
4.3. Build FPGA Bitstream
4.4. Program Target Device
FPGA Bitstream
and generate bitstream for embedded system on FPGA
meters
Browse...
Run This Task
Help
App

```
1/runme.log
system_axi_ad3552r_0_0_synth_1: /home/spop/work/lldk_rebased_on_hdl_master_ref/H
ighSpeedConverterToolbox/hdl_prj/vivado_ip_prj/vivado_prj.runs/system_axi_ad3552
r_0_0_synth_1/runme.log
system_axi_dac_0_dma_0_synth_1: /home/spop/work/lldk_rebased_on_hdl_master_ref/H
ighSpeedConverterToolbox/hdl_prj/vivado_ip_prj/vivado_prj.runs/system_axi_dac_0_
dma_0_synth_1/runme.log
system_axi_ad3552r_1_0_synth_1: /home/spop/work/lldk_rebased_on_hdl_master_ref/H
ighSpeedConverterToolbox/hdl_prj/vivado_ip_prj/vivado_prj.runs/system_axi_ad3552
r_1_0_synth_1/runme.log
system_axi_dac_1_dma_0_synth_1: /home/spop/work/lldk_rebased_on_hdl_master_ref/H
ighSpeedConverterToolbox/hdl_prj/vivado_ip_prj/vivado_prj.runs/system_axi_dac_1_
dma_0_synth_1/runme.log
system_axi_hp2_interconnect_0_synth_1: /home/spop/work/lldk_rebased_on_hdl_maste
r_ref/HighSpeedConverterToolbox/hdl_prj/vivado_ip_prj/vivado_prj.runs/system_axi
_hp2_interconnect_0_synth_1/runme.log
[Wed Dec 14 11:23:56 2022] Launched synth_1...
Run output will be captured here: /home/spop/work/lldk_rebased_on_hdl_master_ref
/HighSpeedConverterToolbox/hdl_prj/vivado_ip_prj/vivado_prj.runs/synth_1/runme.l
og
launch_runs: Time (s): cpu = 00:00:35 ; elapsed = 00:00:59 . Memory (MB): peak =
 3302.270 ; gain = 132.352 ; free physical = 8990 ; free virtual = 71509
[Wed Dec 14 11:23:56 2022] Waiting for synth_1 to finish...
```

uild process externally

r synthesis build: Custom | eedConverterToolbox/CI/scripts/adi_build.tcl

e routed design checkpoint for build

esign checkpoint file for build: Default

Routed design checkpoint file: hdl_prj/checkpoint/system_routed.dcp

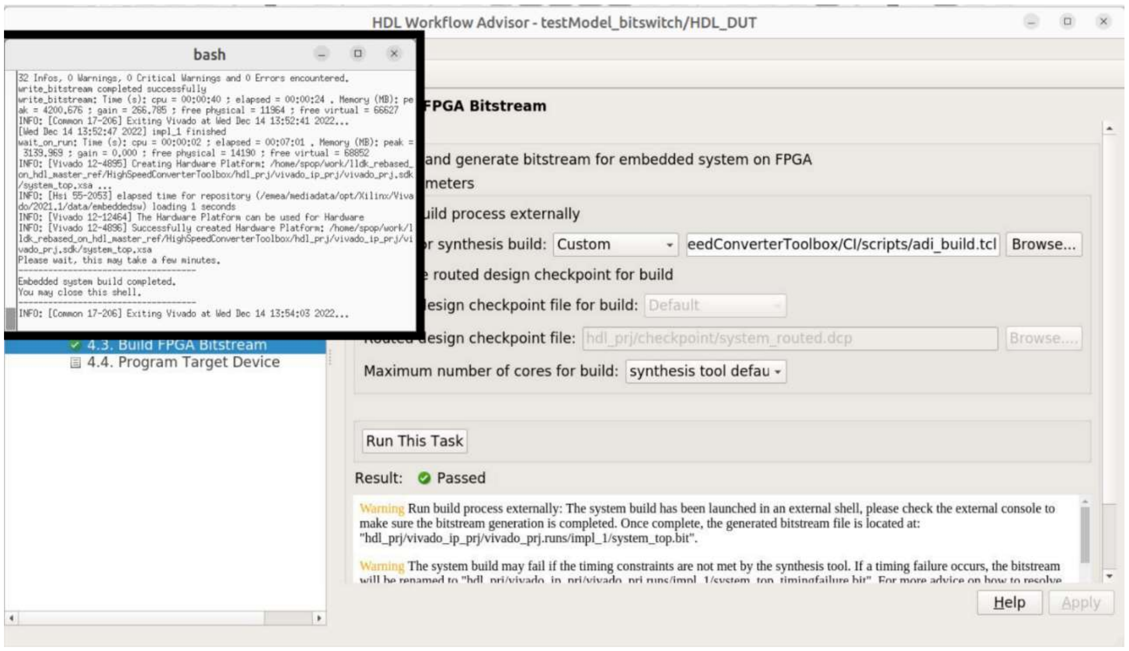
Maximum number of cores for build: synthesis tool defau

Result: Passed

Warning Run build process externally: The system build has been launched in an external shell, please check the external console to make sure the bitstream generation is completed. Once complete, the generated bitstream file is located at: "hdl_prj/vivado_ip_prj/vivado_prj.runs/impl_1/system_top.bit".

Warning The system build may fail if the timing constraints are not met by the synthesis tool. If a timing failure occurs, the bitstream will be renamed to "hdl_prj/vivado_ip_prj/vivado_prj.runs/impl_1/system_top_timingfailure.bit". For more advice on how to resolve

Figure 17. Build FPGA Bitstream


```
32 Infos, 0 Warnings, 0 Critical Warnings and 0 Errors encountered.
write_bitstream completed successfully
write_bitstream: Time (s): cpu = 00:00:40 ; elapsed = 00:00:24 . Memory (MB): pe
ak = 4200.676 ; gain = 266.785 ; free physical = 11964 ; free virtual = 66627
INFO: [Common 17-206] Exiting Vivado at Wed Dec 14 13:52:41 2022...
[Wed Dec 14 13:52:47 2022] impl_1 finished
wait_on_run: Time (s): cpu = 00:00:02 ; elapsed = 00:07:01 . Memory (MB): peak =
 3139.969 ; gain = 0.000 ; free physical = 14190 ; free virtual = 68852
INFO: [Vivado 12-4895] Creating Hardware Platform: /home/spop/work/1ldk_rebased_
on_hdl_master_ref/HighSpeedConverterToolbox/hdl_prj/vivado_ip_prj/vivado_prj.sdk
/system_top.xsa ...
INFO: [Hsi 55-2053] elapsed time for repository (/emea/mediadata/opt/Xilinx/Viva
do/2021.1/data/embeddedsw) loading 1 seconds
INFO: [Vivado 12-12464] The Hardware Platform can be used for Hardware
INFO: [Vivado 12-4896] Successfully created Hardware Platform: /home/spop/work/1
ldk_rebased_on_hdl_master_ref/HighSpeedConverterToolbox/hdl_prj/vivado_ip_prj/vi
vado_prj.sdk/system_top.xsa
Please wait, this may take a few minutes.
-------------------------------------
Embedded system build completed.
You may close this shell.
-------------------------------------
INFO: [Common 17-206] Exiting Vivado at Wed Dec 14 13:54:03 2022...
```

HDL Workflow Advisor - testModel_bitswitch/HDL_DUT
4.3. Build FPGA Bitstream
4.4. Program Target Device
FPGA Bitstream
Run This Task
Help
Apply

and generate bitstream for embedded system on FPGA

meters

uild process externally

r synthesis build: Custom | eedConverterToolbox/CI/scripts/adi_build.tcl

e routed design checkpoint for build

esign checkpoint file for build: Default

design checkpoint file: hdl_prj/checkpoint/system_routed.dcp

Maximum number of cores for build: synthesis tool defau

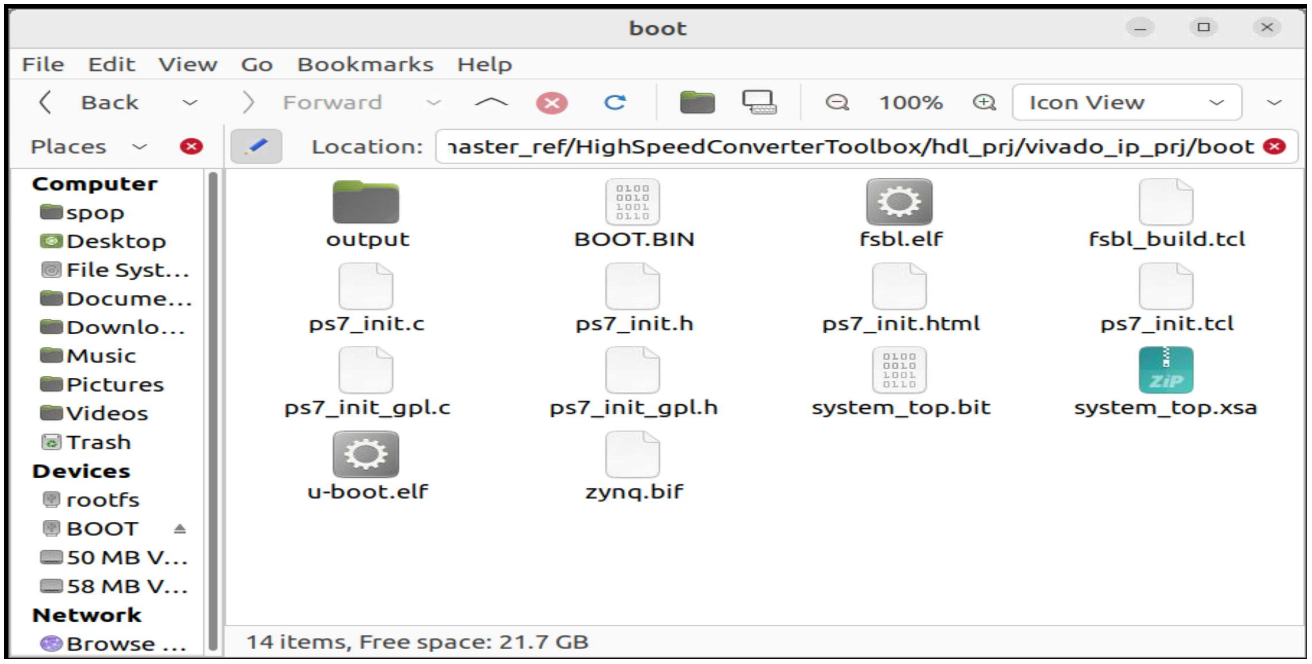
Result: Passed

Warning Run build process externally: The system build has been launched in an external shell, please check the external console to make sure the bitstream generation is completed. Once complete, the generated bitstream file is located at: "hdl_prj/vivado_ip_prj/vivado_prj.runs/impl_1/system_top.bit".

Warning The system build may fail if the timing constraints are not met by the synthesis tool. If a timing failure occurs, the bitstream will be renamed to "hdl_prj/vivado_ip_prj/vivado_prj.runs/impl_1/system_top_timingfailure.bit". For more advice on how to resolve

Figure 18. Build FPGA Bitstream Task Complete Message

In the end you will get this message, and the generated BOOT.BIN file will be located here, Figure 19.



Location: ıaster_ref/HighSpeedConverterToolbox/hdl_prj/vivado_ip_prj/boot

output
BOOT.BIN
fsbl.elf
fsbl_build.tcl
ps7_init.c
ps7_init.h
ps7_init.html
ps7_init.tcl
ps7_init_gpl.c
ps7_init_gpl.h
system_top.bit
system_top.xsa
u-boot.elf
zynq.bif

14 items, Free space: 21.7 GB

Figure 19. Location of BOOT.BIN File

3.4 Program target device

This tab in the HDL Workflow Advisor is incompatible with The ADI SD card flow. Instead, choose one of the following methods to update the BOOT.BIN file on the SD card.

After the BOOT.BIN file is generated, you have 2 options:

- Copy the BOOT.BIN file on the SD Card directly.
- Send it via network using a terminal (CMD for Windows machine):
    1. Go to the folder where the BOOT.BIN file is:

HighSpeedConverterToolbox/hdl_prj/vivado_ip_prj/boot

2. Run this command:

scp BOOT.BIN root@<your_board_ip>:/boot

Finally, reboot the board.

Register Access Options:

AXI-Lite registers in HDL_DUT can be accessed using one of the below three options:

1. *PyADI-IIO*

git clone https://github.com/analogdevicesinc/pyadi-iio.git

git checkout **cn0585_axi_reg**

cd pyadi-iio

export PYTHONPATH=**C:/work/python_LLDK/documentation_clone/pyadi-iio/**

The path is the location where you cloned the pyadi-iio repository.

```
> ../pyadi-iio > pip install .

> ../pyadi-iio > pip install -r requirements.txt

> ../pyadi-iio > pip install -r requirements_dev.txt

> ../pyadi-iio> python examples/cn0585_fmcz_example.py ip:<your_board_ip>
```

The updated version of the console output will contain these 2 new lines:

```
AXI4-Lite 0x108 register value: 0xC

AXI4-Lite 0x10c register value: 0xD
```

These are the functions that were added to be able to access the HDL DUT IP registers trough AXI4-Lite:

```
hdl_dut_write_channel.axi4_lite_register_write(0x100, 0xc)

hdl_dut_write_channel.axi4_lite_register_write(0x104, 0xd)
```

reg_value = hdl_dut_read_channel.axi4_lite_register_read(0x108)

reg_value1 = hdl_dut_read_channel.axi4_lite_register_read(0x10c)

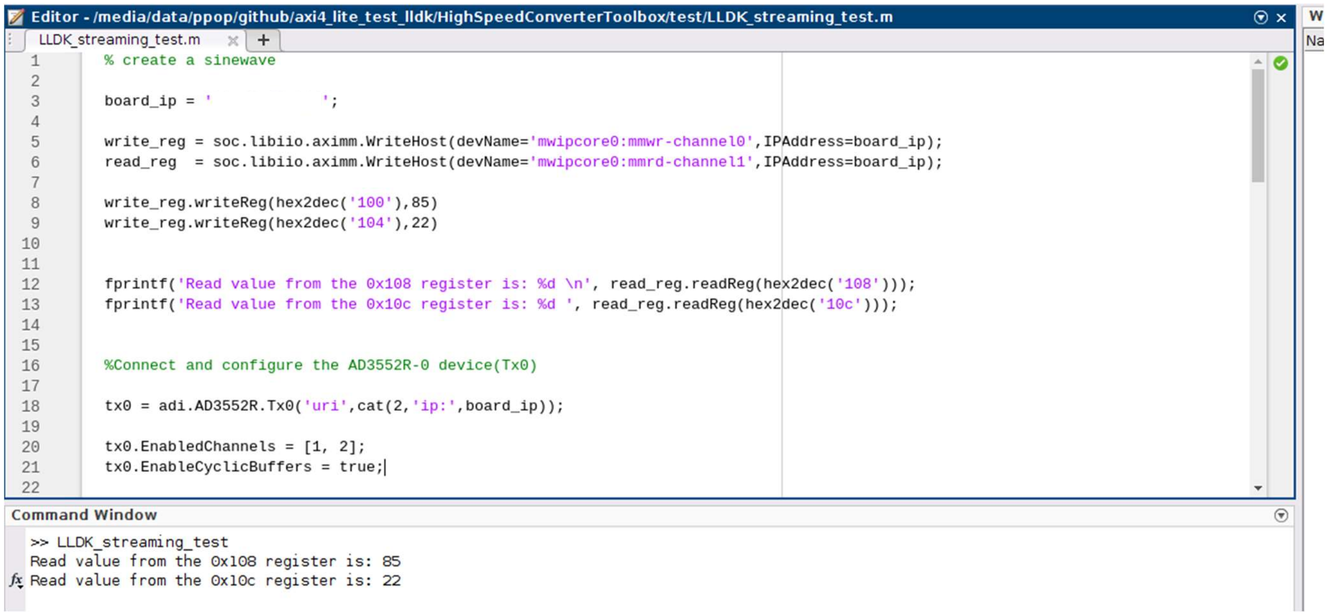
print("AXI4-Lite 0x108 register value:", reg_value)

print("AXI4-Lite 0x10c register value:", reg_value1)

2. MATLAB

1. Open the CN0585StreamingTest.m file in Matlab
2. Update the board_ip variable with your board IP.
3. Run the CN0585StreamingTest.m example.

The output described by Figure 20 can be observed in the Command Window.



Editor - /media/data/ppop/github/axi4_lite_test_lldk/HighSpeedConverterToolbox/test/LLDK_streaming_test.m
LLDK_streaming_test.m
Command Window

```
% create a sinewave

board_ip = '            ';

write_reg = soc.libiio.aximm.WriteHost(devName='mwipcore0:mmwr-channel0',IPAddress=board_ip);
read_reg  = soc.libiio.aximm.WriteHost(devName='mwipcore0:mmrd-channel1',IPAddress=board_ip);

write_reg.writeReg(hex2dec('100'),85)
write_reg.writeReg(hex2dec('104'),22)


fprintf('Read value from the 0x108 register is: %d \n', read_reg.readReg(hex2dec('108')));
fprintf('Read value from the 0x10c register is: %d ', read_reg.readReg(hex2dec('10c')));


%Connect and configure the AD3552R-0 device(Tx0)

tx0 = adi.AD3552R.Tx0('uri',cat(2,'ip:',board_ip));

tx0.EnabledChannels = [1, 2];
tx0.EnableCyclicBuffers = true;

```

```
>> LLDK_streaming_test
Read value from the 0x108 register is: 85
Read value from the 0x10c register is: 22
```

Figure 20. MATLAB Command Window Output

These are the functions that were added to be able to access the HDL DUT IP registers trough AXI4-Lite:

write_reg = soc.libiio.aximm.WriteHost(devName='mwipcore0:mmwr-channel0',IPAddress=board_ip);

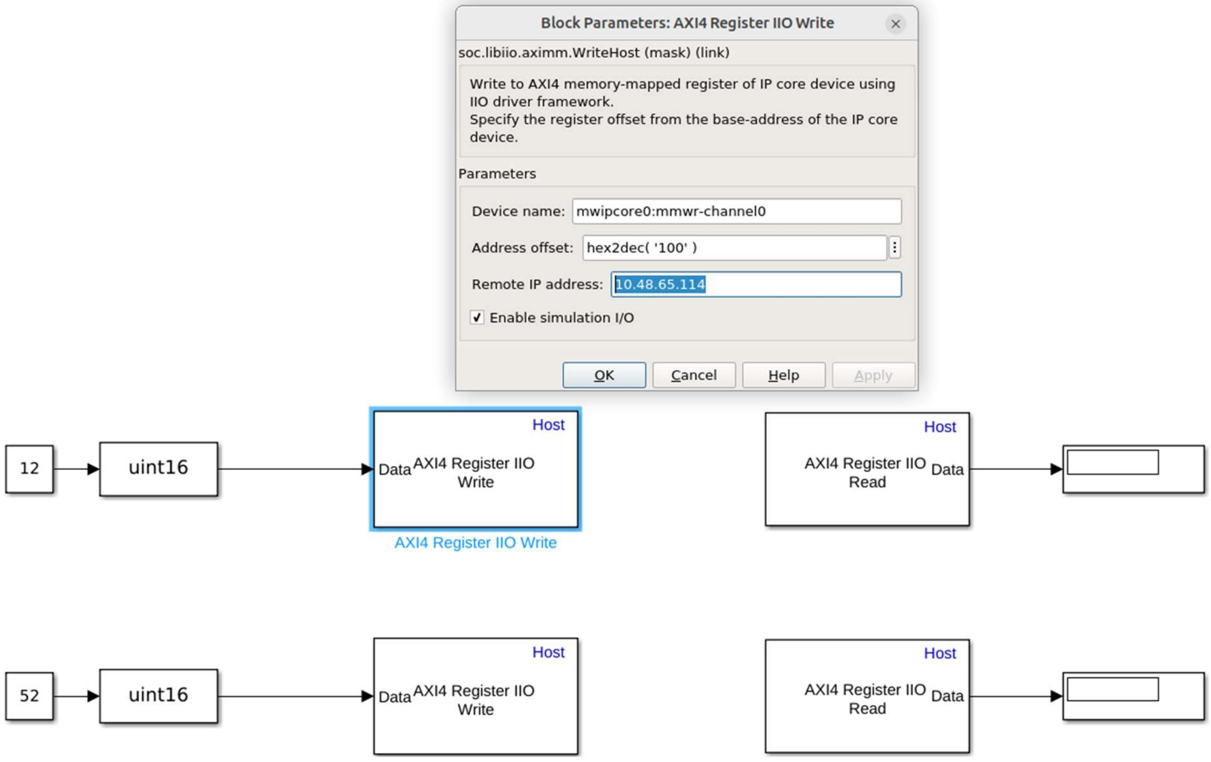
read_reg = soc.libiio.aximm.WriteHost(devName='mwipcore0:mmrd-channel1',IPAddress=board_ip);

write_reg**.**writeReg**(**hex2dec**(**'100'**)**,85**)**

write_reg**.**writeReg**(**hex2dec**(**'104'**)**,22**)**

3. Simulink

1. From the HighSpeedConverterToolbox/test folder open the cn0585_host_axi4_lite_read_write_example.slx file.
2. Update the **IP address** for all the blocks existing in the host diagram.



Block Parameters: AXI4 Register IIO Write

soc.libiio.aximm.WriteHost (mask) (link)

Write to AXI4 memory-mapped register of IP core device using IIO driver framework.
Specify the register offset from the base-address of the IP core device.

Parameters

Device name: mwipcore0:mmwr-channel0

Address offset: hex2dec('100')

Remote IP address: 10.48.65.114

✓ Enable simulation I/O

Figure 21. Host Simulink Block Diagram