

Using the AMD Vitis™ Model Composer Hub Block

Agenda

Introduction

Hardware Selection

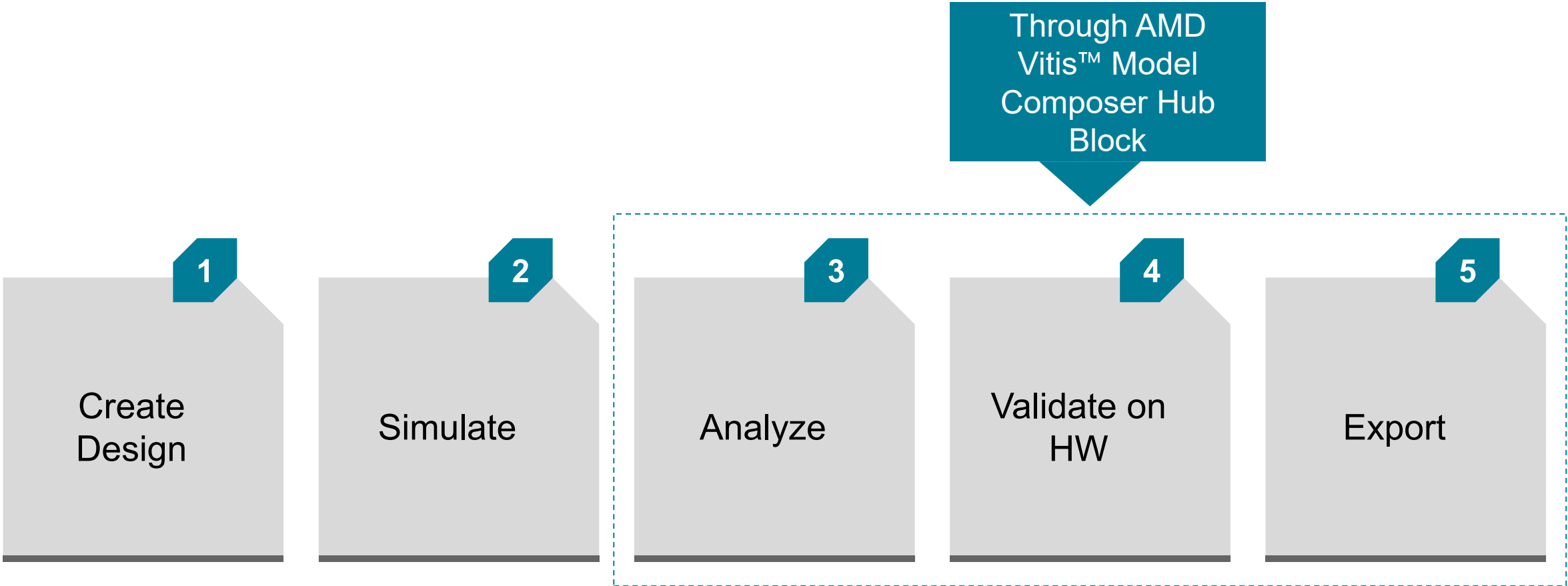
Code Generation

Analyzing and Verifying Designs

Hardware Validation

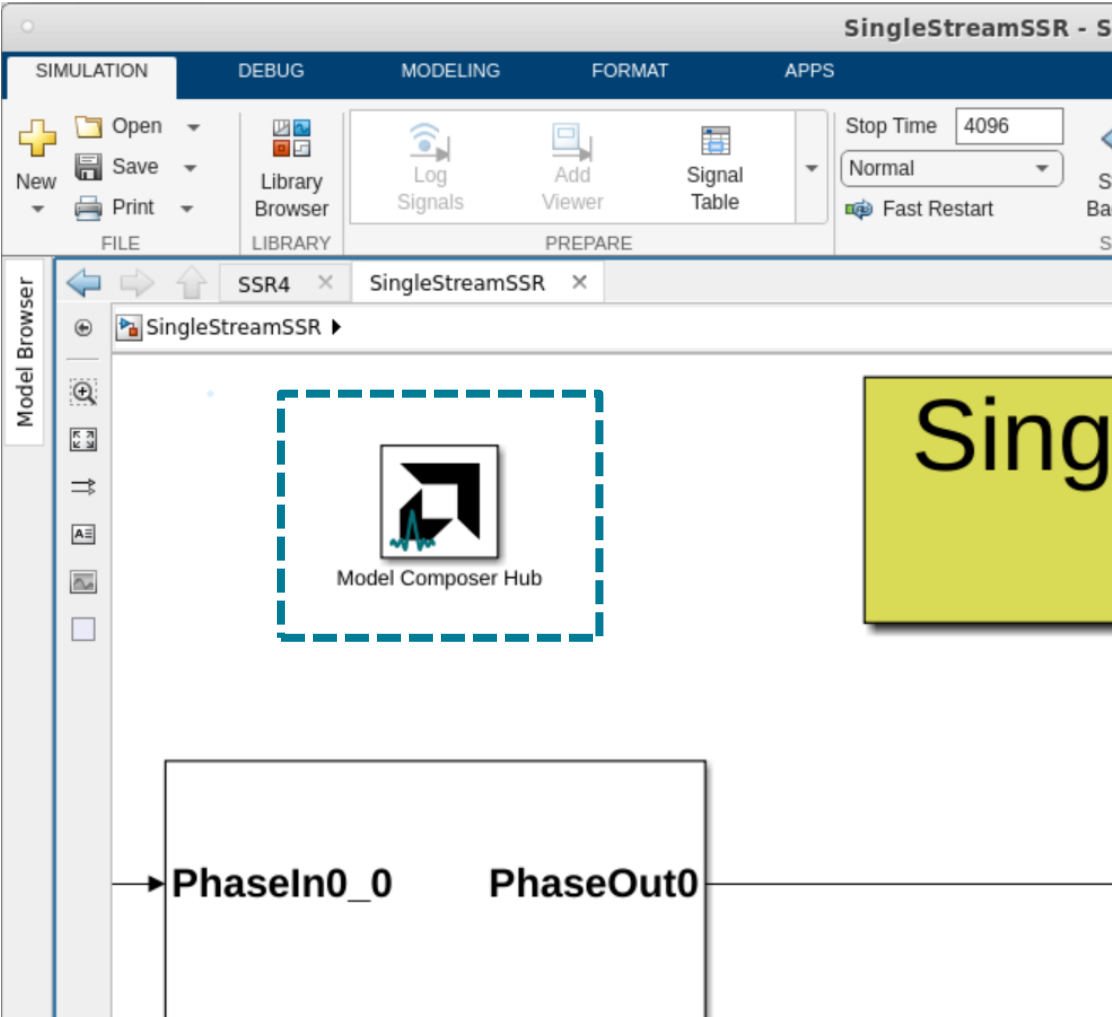
Summary

AMD Vitis™ Model Composer Hub



What is the Hub Block?

- Every design has one Hub block
- Allows for many design-level controls
- Hardware selection



AMD Vitis™ Model Composer Hub

AMD Vitis™ Model
Composer



Configures compilation and
generates outputs

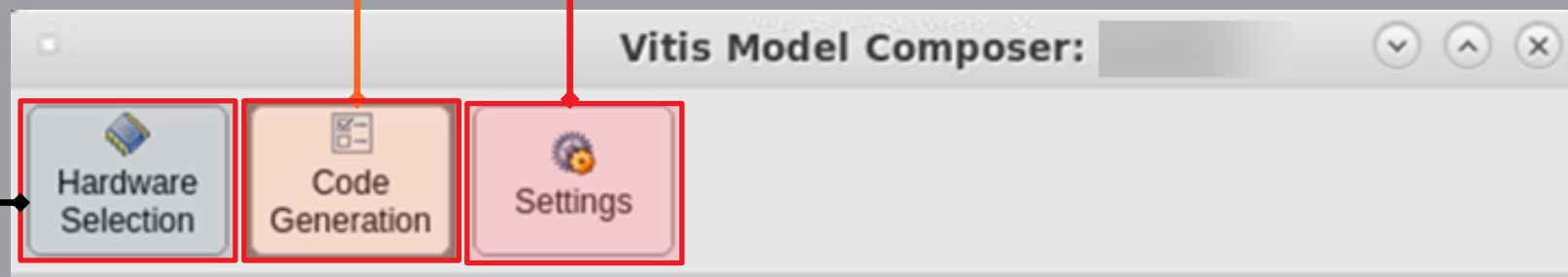


Controls the behavior of the tool

Provides options to select the output flow

Specifies whether the model should be
treated as a legacy
System Generator design

Helps with device, board, or
platform selection



Agenda

Introduction

Hardware Selection

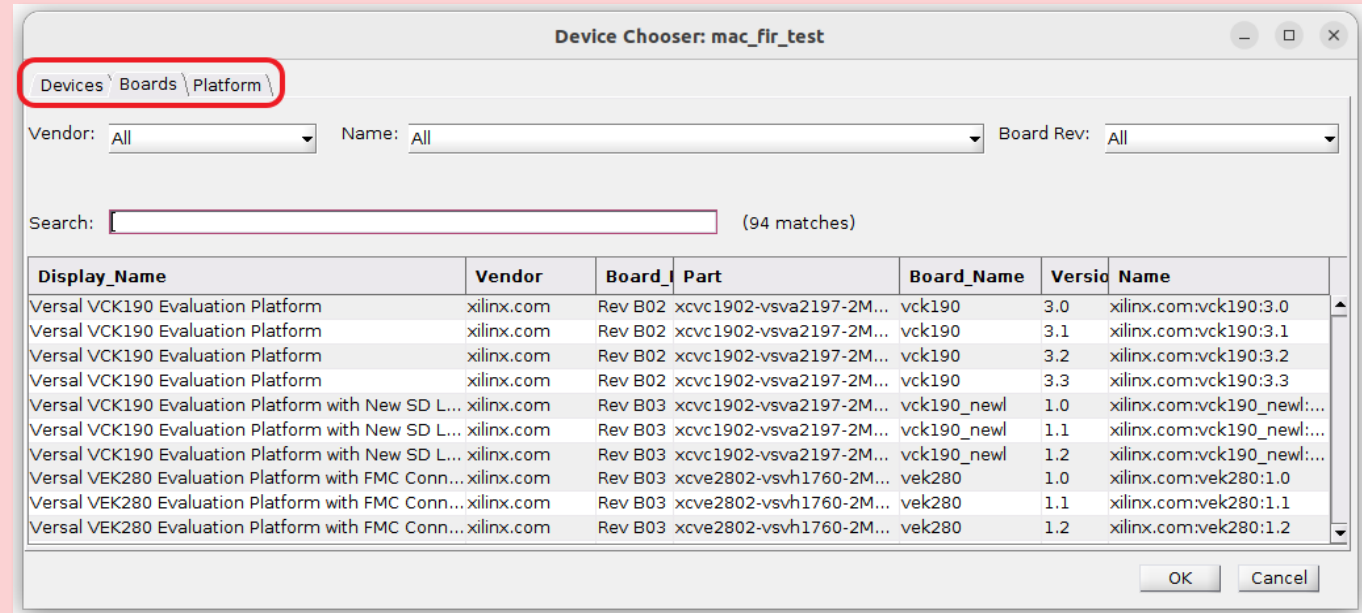
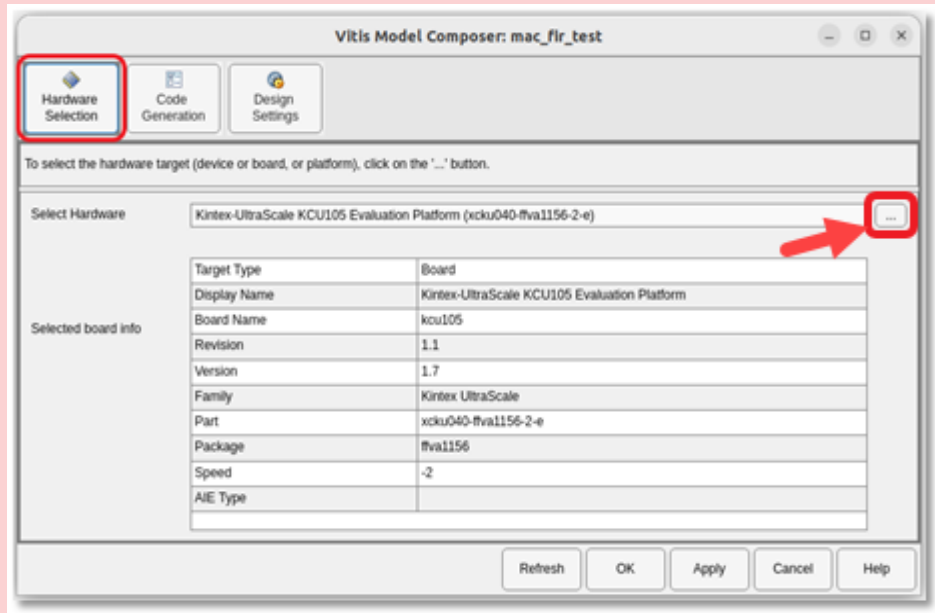
Code Generation

Analyzing and Verifying Designs

Hardware Validation

Summary

Hardware Selection



Device Chooser obtains data from Vivado™ IDE database

Agenda

Introduction

Hardware Selection

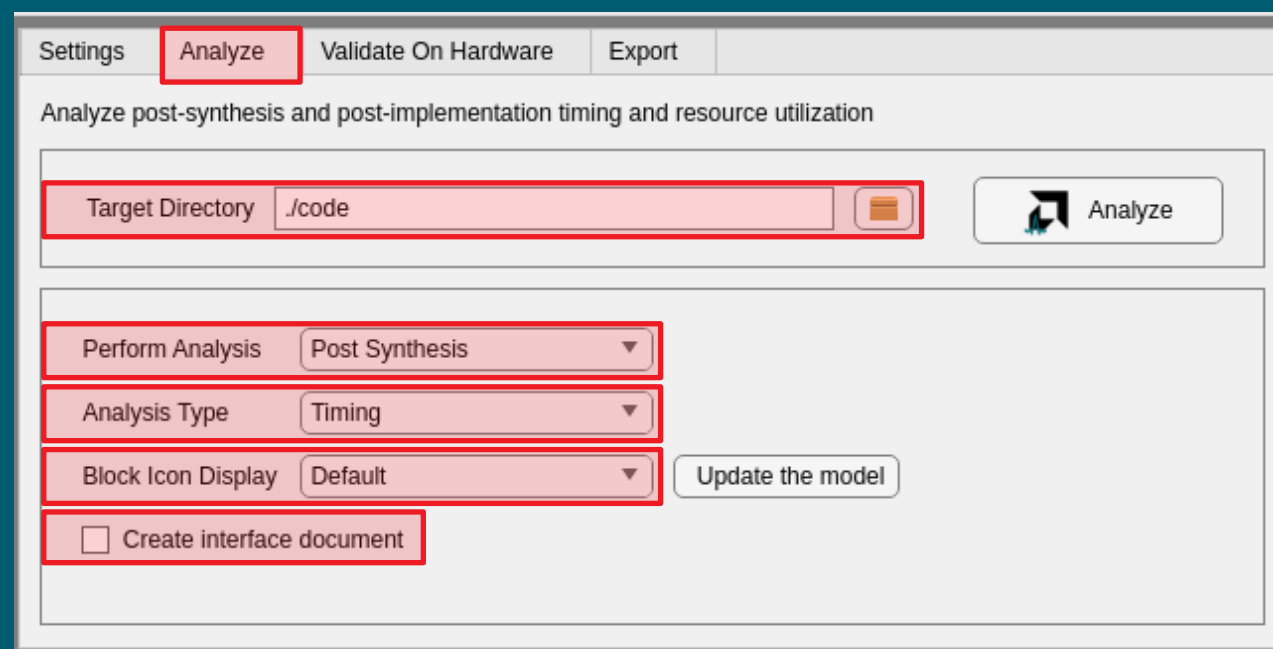
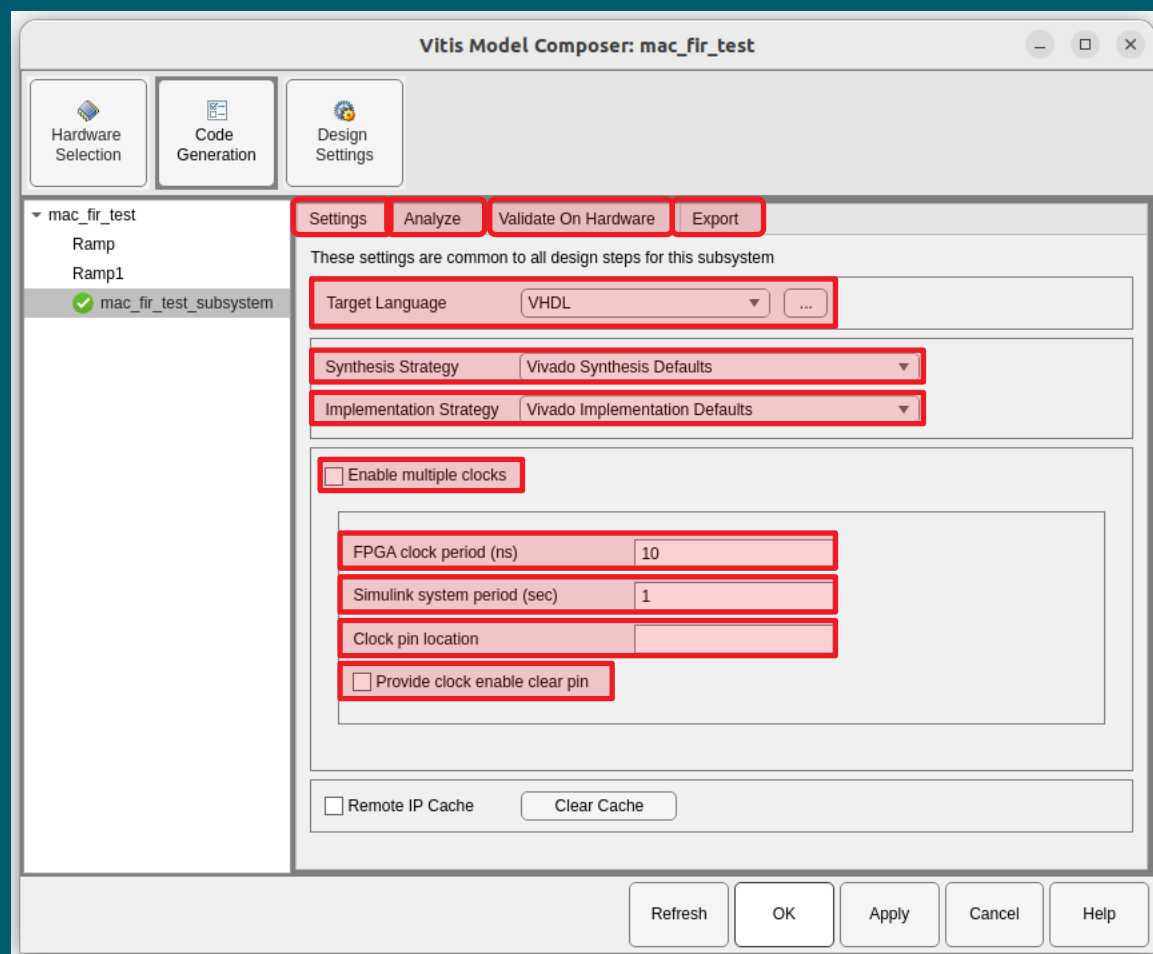
Code Generation

Analyzing and Verifying Designs

Hardware Validation

Summary



Code Generation – HDL Design



Code Generation – HDL Design

Settings Analyze **Validate On Hardware** Export

Generate hardware image to validate HDL subsystem in hardware or hardware emulation.

Target Directory   Validate

HW System Type

☒ Baremetal ☐ Linux

Target



Common SW Directory ...

Target SDK Directory ...

☐ Generate BOOT.BIN after code generation

Settings Analyze Validate On Hardware **Export**

Export HDL subsystem for system integration.

Export Directory   Export

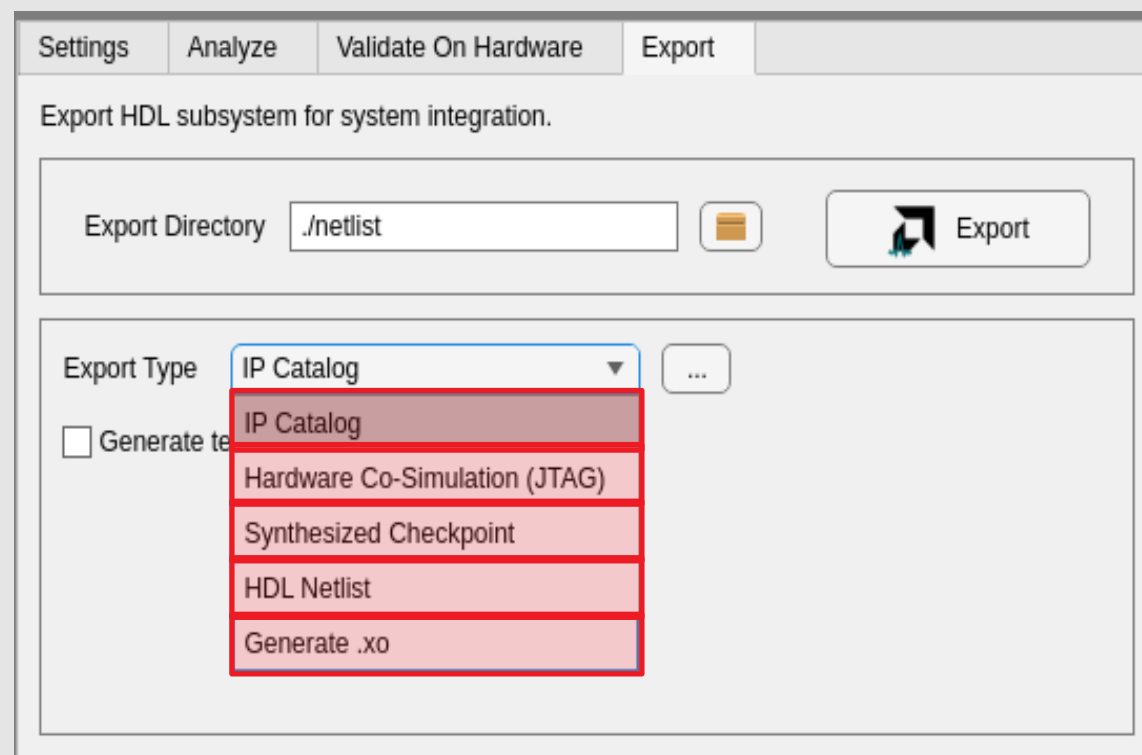
Export Type ...

☐ Generate testbench

* Testbench generation is not supported for designs that have gateways (Gateway In or Gateway Out) configured as an AXI4-Lite interface

Export Types for AMD Vitis™ Model Composer HDL Designs

- Can create following types of output from an HDL model:
 - IP Catalog
 - Hardware Co-Simulation
 - Synthesized Checkpoint
 - HDL Netlist
 - Generate .xo



Code Generation – AMD Vitis™ HLS Design

Settings Analyze Export

These settings are common to all design steps for this subsystem



FPGA clock frequency (MHz) 200

Throughput Factor 1

Testbench stack size (MBytes) 10



Settings Analyze Export


Run RTL co-simulation, verify output against Simulink, and analyze latency and initiation interval (II)

Target Directory   Analyze

Settings Analyze Export

Export HLS subsystem for system integration

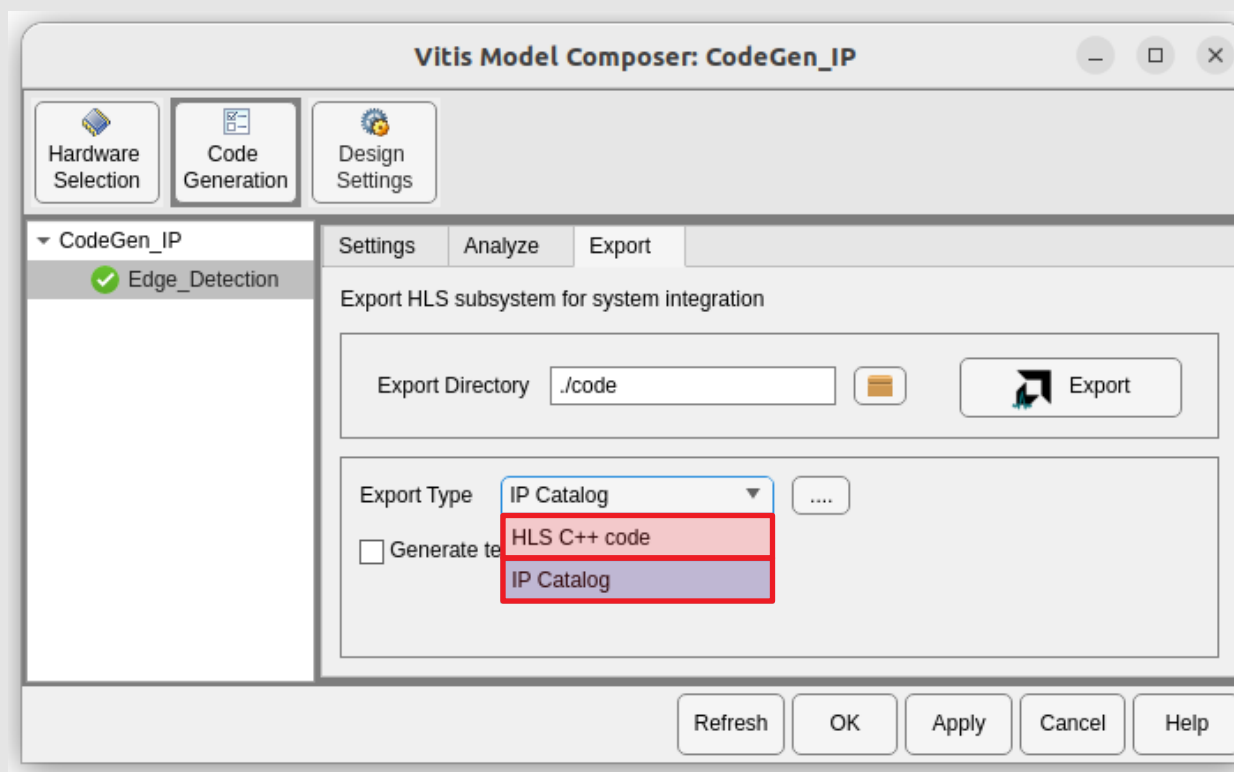
Export Directory ./code   Export

Export Type IP Catalog 

☐ Generate testbench

Export Types for AMD Vitis™ Model Composer HLS Designs

- Can create two different types of output from an HLS model:
 - HLS C++ code
 - IP Catalog



Code Generation – AI Engine Design

Settings Analyze Validate On Hardware Export

These settings are common to all design steps for this subsystem

AIE Compiler Options { }

Settings Analyze Validate On Hardware Export

Run System-C simulation, verify output against Simulink, and analyze throughput and latency

Target Directory .code { }

Analyze

AIE Simulator Options { }

Simulation timeout (cycles) 50000

☐ Collect profiling statistics and enable 'printf' for debugging



☐ Collect trace data for Vitis Analyzer, viewing internal signals, and latency

View AIE Simulation output and throughput Open Vitis Analyzer

Code Generation – AI Engine Design

Settings Analyze **Validate On Hardware** Export

Generate hardware image to validate AIE subsystem in hardware or hardware emulation

Target Directory   Validate

HW System Type

☒ Baremetal ☐ Linux

Target



Common SW Directory ...

Target SDK Directory ...

☐ Generate BOOT.BIN after code generation

Settings Analyze Validate On Hardware **Export**

Export AIE subsystem for system integration

Export Directory   Export

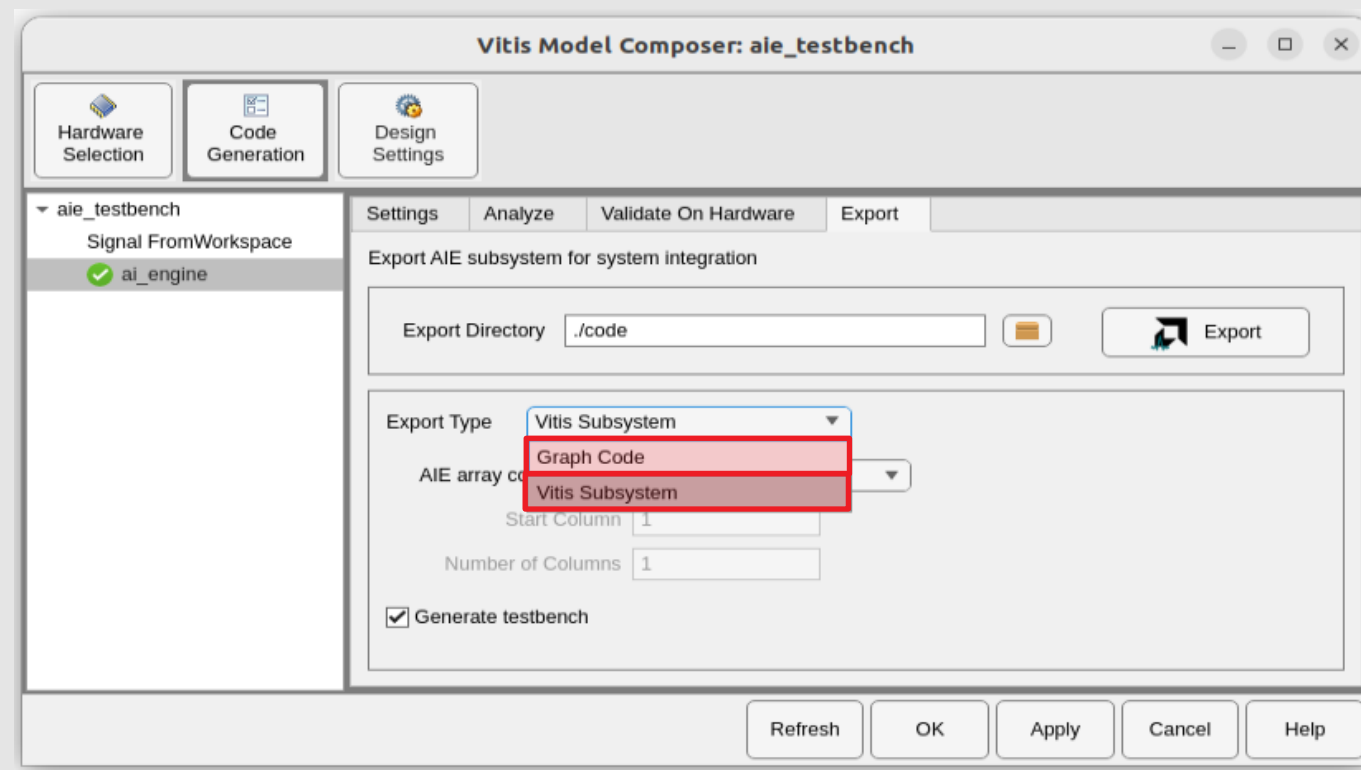
Export Type

☒ Generate testbench

Export Types for AMD Vitis™ Model Composer AI Engine Designs

Can create two different types of output from an AI Engine model:

- Graph Code
- Vitis Subsystem



Agenda

Introduction

Hardware Selection

Code Generation

Analyzing and Verifying Designs

Hardware Validation

Summary

Performing Analysis in HDL Design

Analysis tools integrated in AMD Vitis™ Model Composer to help verify your design works correctly on your target device

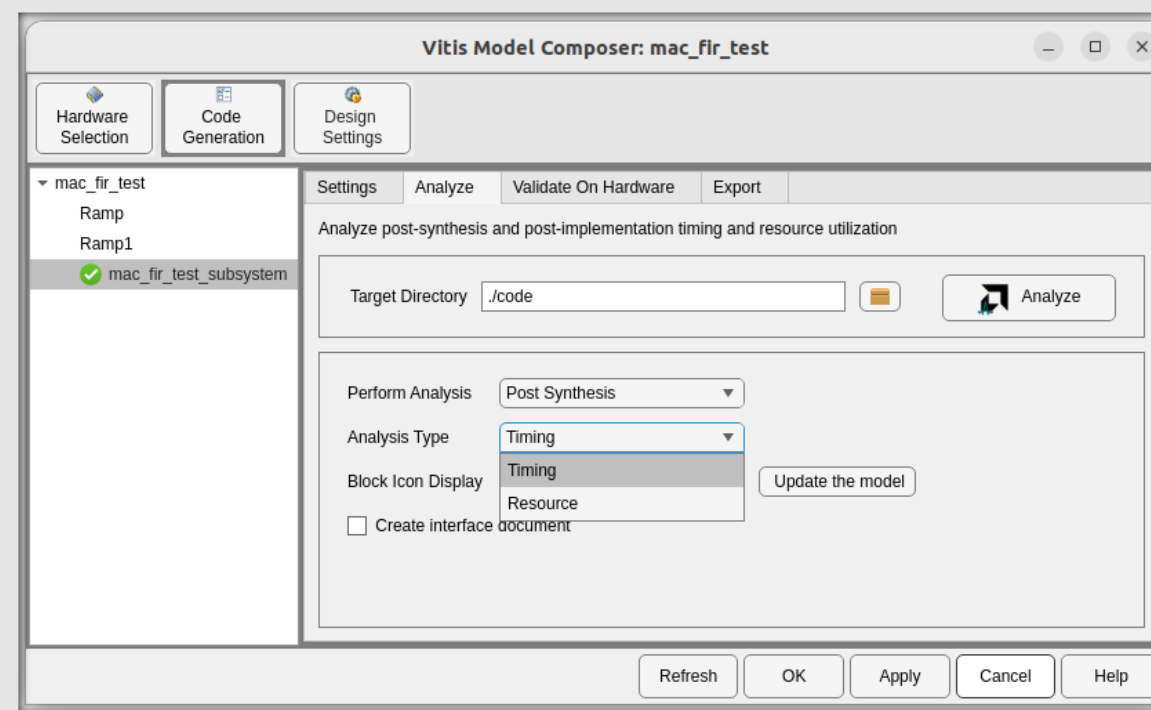
Performance Analysis options:

Timing Analysis

Closing timing to ensure the HDL files operate correctly in hardware

Resource Analysis

Analyzing the resources to ensure they fit into your target device

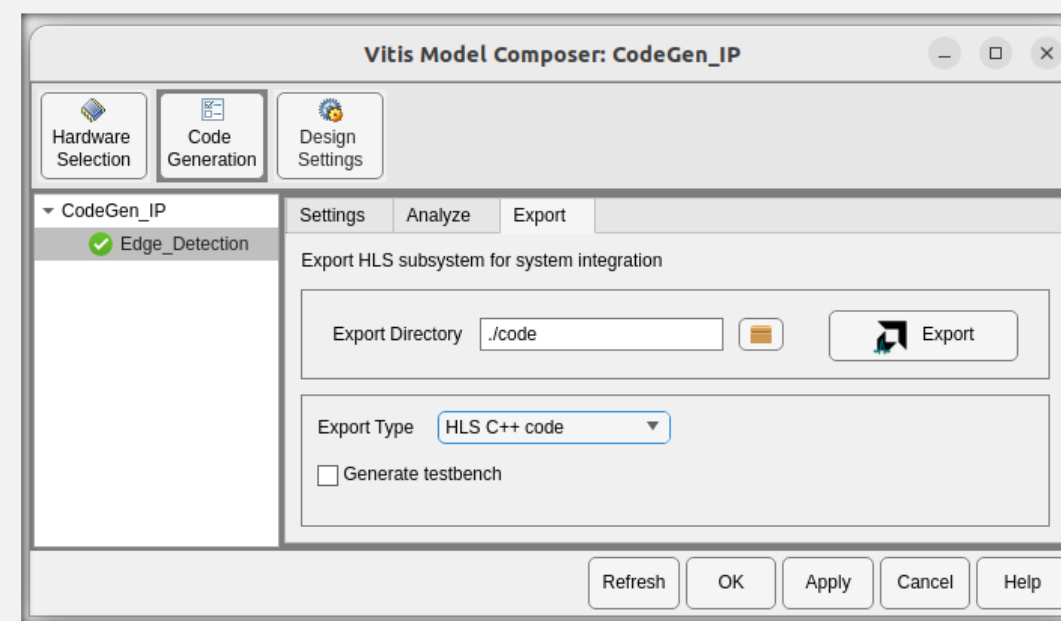


Verifying the C++ Code – AMD Vitis™ HLS Design

When the target specified is **HLS C++ code**, and the verification flow is enabled, Vitis™ Model Composer uses C simulation to verify the generated C++ code

C++ code verification flow:

- Model is simulated, and the input and outputs are logged into the signals.stim binary file
- C++ code and a testbench (tb.cpp, which contains a main() function) are generated
- Results are compared and verified from the generated C++ simulation and the output from the Simulink® simulation (signals.stim)
- Mismatched output signal name is reported, as well as the actual and expected values
- Result returns as a Pass/Fail



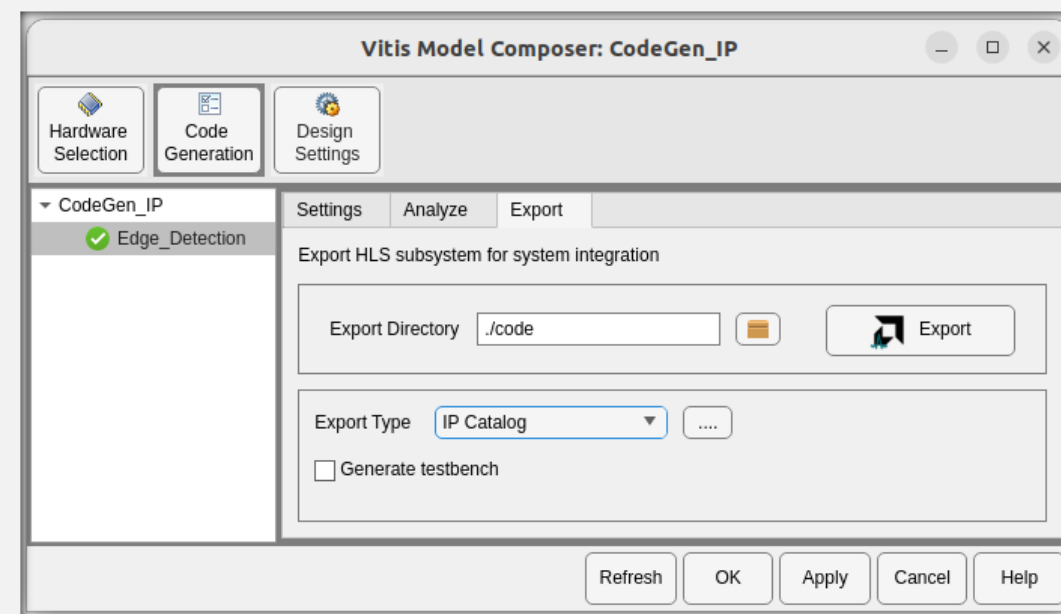
Verifying the C/RTL Code – AMD Vitis™ HLS Design

When the target specified is **IP Catalog**, and the verification flow is enabled, Vitis™ Model Composer runs C/RTL co-simulation

RTL code verification flow:

- Ensures that the C++ code generated is correct by comparing with the Simulink® simulation (signals.stim)
- Ensures that the RTL code generated is correct by comparing output stimulus from RTL with the C/C++ output

Result returns as a Pass/Fail

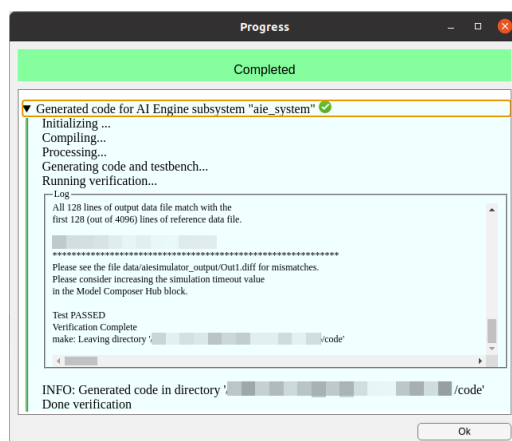


Verification of the AI Engine Code

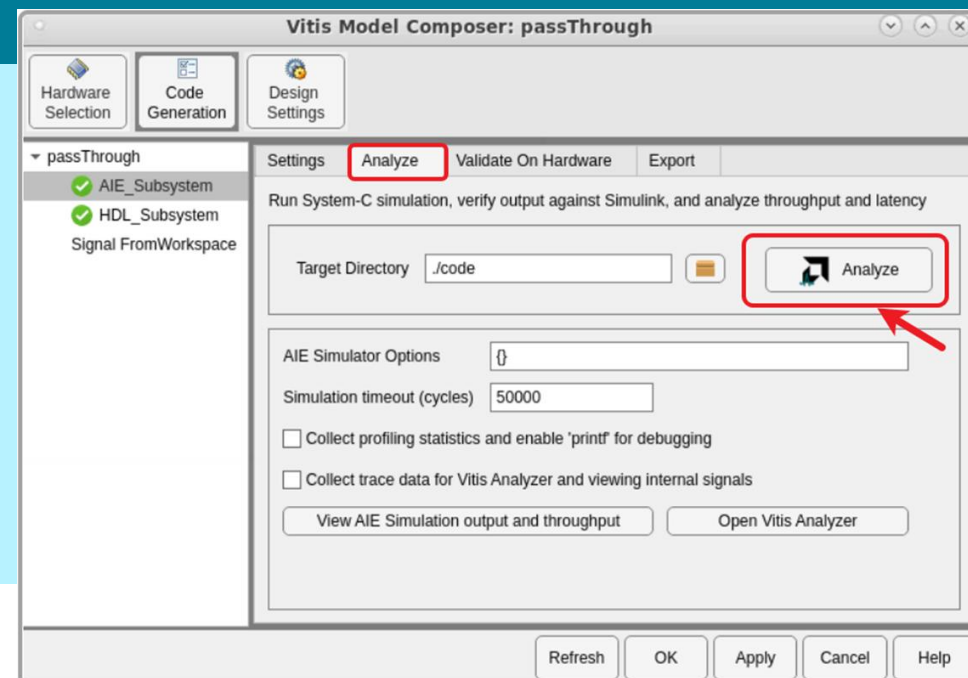
AMD Vitis™ Model Composer supports verification of the data flow graph using the AI Engine simulator

Vitis Model Composer Hub Block for Verification

- Analyze tab: Verifies and analyzes the AI Engine design; generates and compiles AI Engine code
- AI Engine code verification advances in three phases:
 - Compiling the AI Engine graph design
 - Running simulation using the AI Engine simulator
 - Verifying the simulation results by comparing the output with the golden reference output

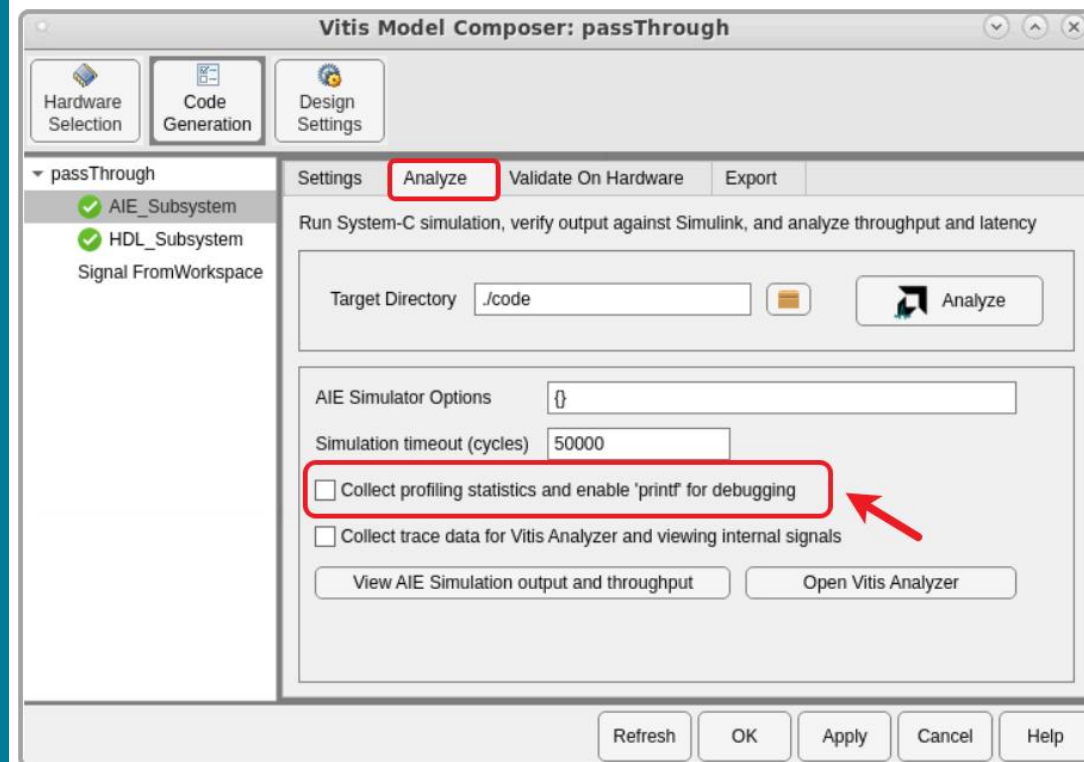


When the AI Engine code cannot be verified with a TEST PASSED message, need to debug it to arrive at the reference output



Profiling Statistics and Event Tracing

- Profiling data helps to gauge the efficiency of the kernels, the stall and active times of each AI Engine, and kernels whose performance may not be optimal
- Collect data on design latency, throughput, and bandwidth
- Event trace can be performed using a formatted printf statement in the code for printing debug messages
 - By enabling the ***Collect profiling statistics and enable 'printf' for debugging*** option in the Vitis™ Model Composer Hub block

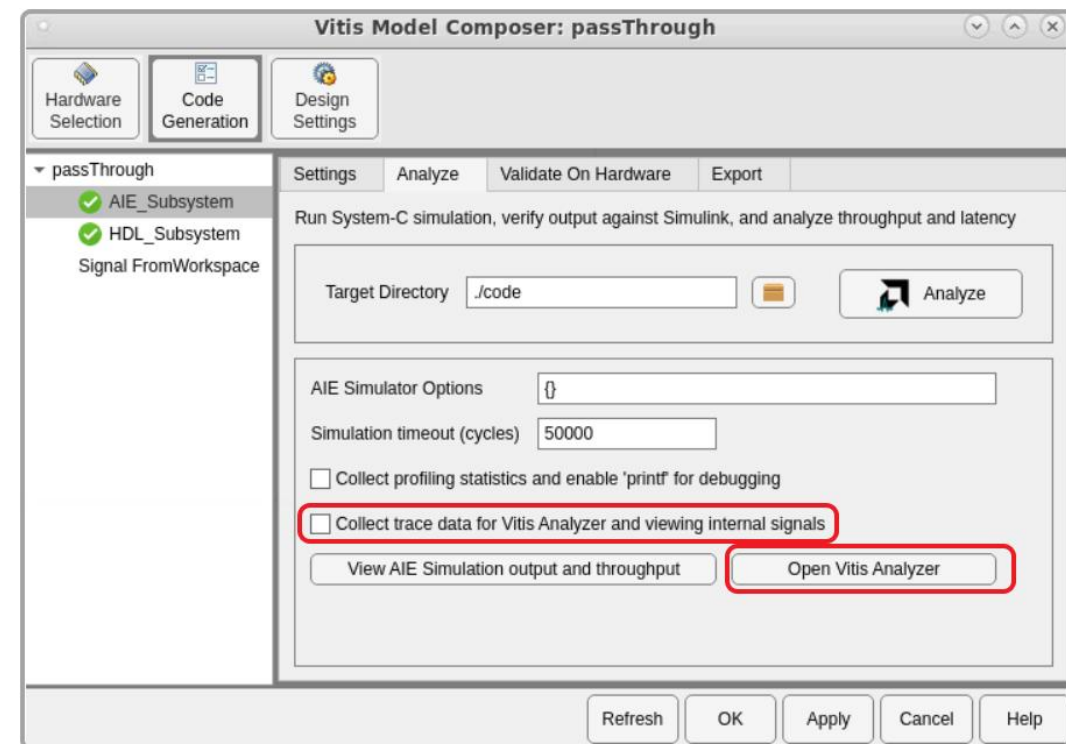


Viewing Results in the AMD Vitis™ Analyzer

AMD Vitis™ software platform analyzer is a utility to view and analyze the reports generated when building and running the application

During the simulation of the AI Engine graph, the AI Engine simulator writes a summary of the simulation results called `default.aierun_summary`

The Vitis analyzer utility can be invoked by enabling the Collect Data for Vitis Analyzer option from the Hub block



- From the report navigator, you can view other available reports, such as the Summary, Profile, Graph, Array, and Log

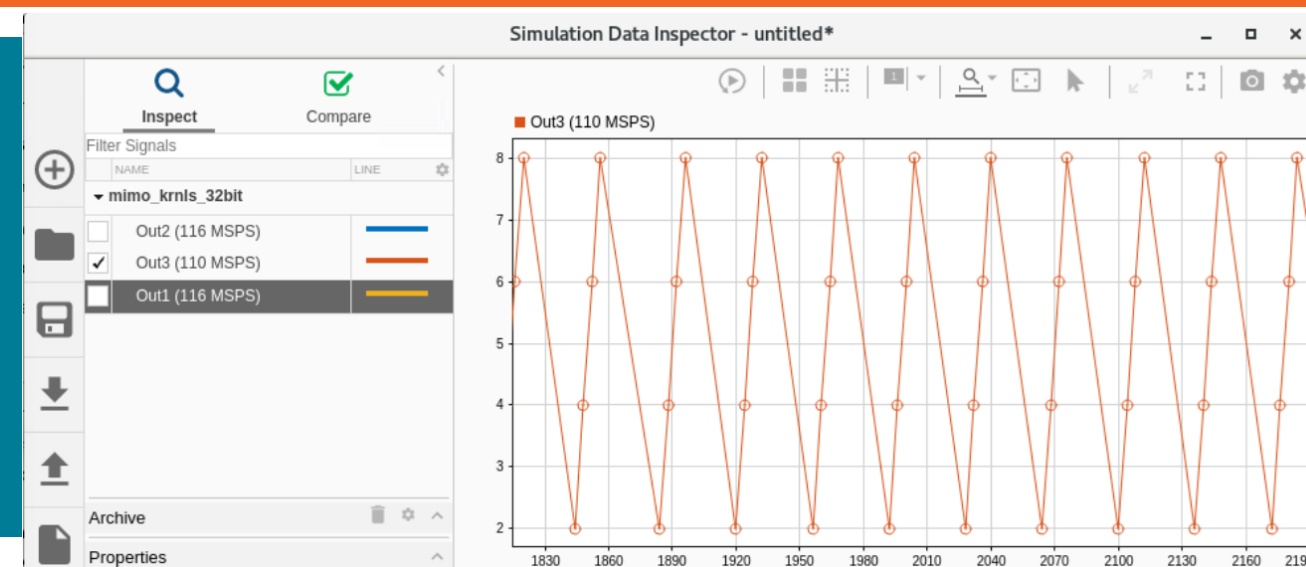
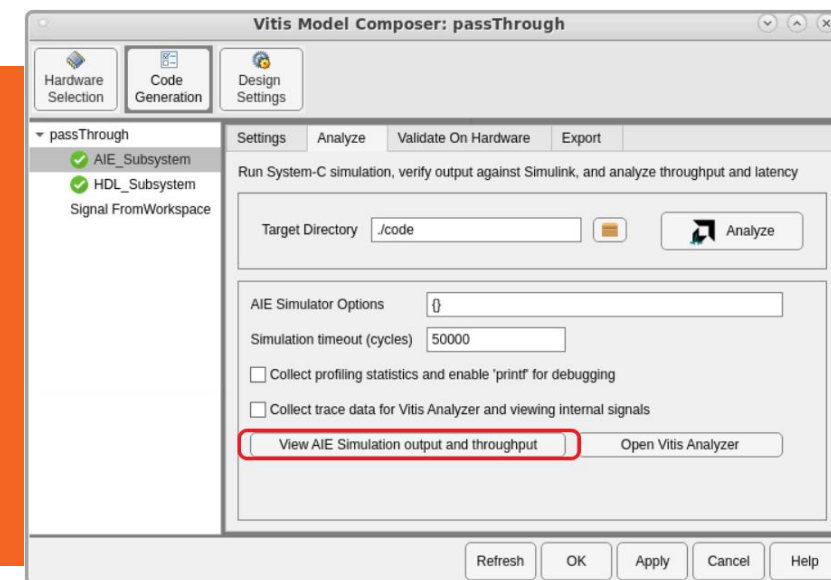
In Vitis Model Composer, you can launch the Vitis analyzer from the MATLAB® command window using the command:

```
xmcOpenVitisAnalyzer('file')
```

Viewing AI Engine Simulation Output and Throughput

Vitis™ Model Composer provides the capability to:

- Log the simulation data and visualize the output of an AI Engine subsystem by integrating the Simulink® tool's Simulation Data Inspector feature
- Calculate the throughput for each output port of the AI Engine subsystem (*View AIE Simulation output and throughput* option)



Simulation Data Inspector displays available data in the Inspect pane

- To plot a signal, select the check box next to the signal
- Modify the layout and add different visualizations to analyze the simulation data
- Obtain the throughput information for each port from the Inspect pane

Calculating Latency between AI Engine Ports

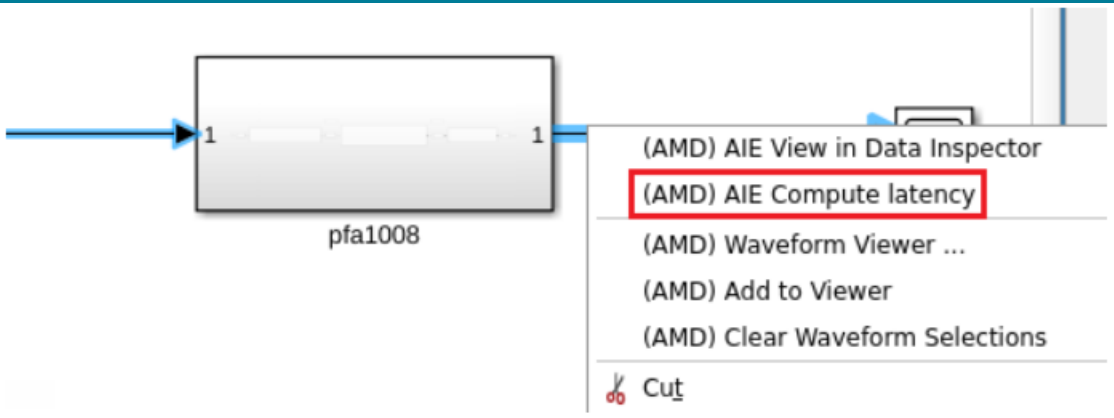
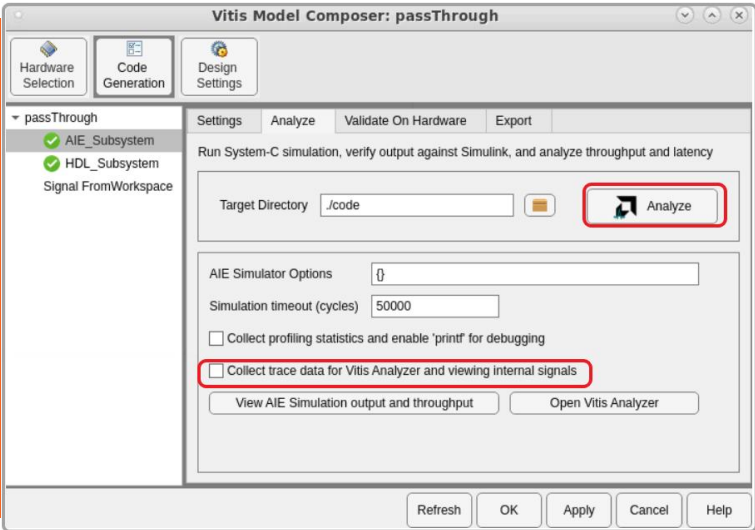
AIE simulation is a cycle-approximate

Estimate Latency between input-output ports of AI Engine subsystem

Visualize Latency from AMD Vitis™ Model Composer:

- Enable collect trace data option
- Run the AI Engine Simulation
- Select the AI Engine input/output signals
- Right-click and select “AIE Compute Latency”

Based on previous sample runs, the table shows first, last and avg. sample latency



(AMD) AIE Compute latency				
Latency computed between the input and output ports of "pfa1008" subsystem.				
"First Latency (ps)" is the latency between the first input sample and first output sample.				
"Last Latency (ps)" is the latency between the last input sample and last output sample.				
"Average Latency (ps)" is the difference between average output sample time and average input sample time.				
Input Port	Output Port	First Latency (ps)	Last Latency (ps)	Average Latency (ps)
In1	Out1	7193600	437600	3654698

Plotting AI Engine Simulation Internal Signals

Simulation Data Inspector can also be used for signals within an AI Engine subsystem

Collect Trace Data in Hub Block

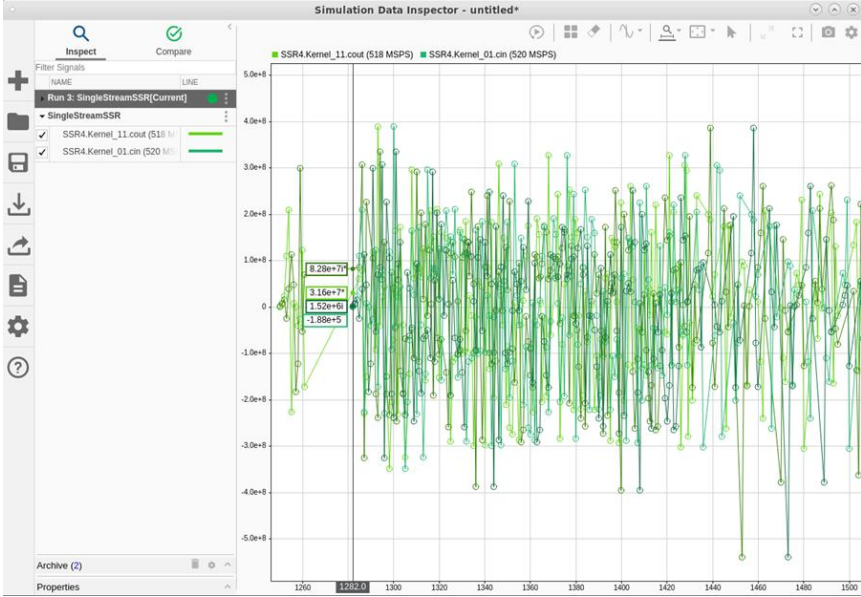
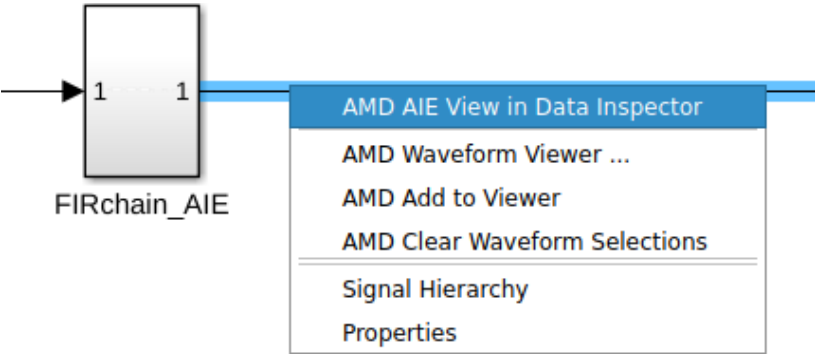
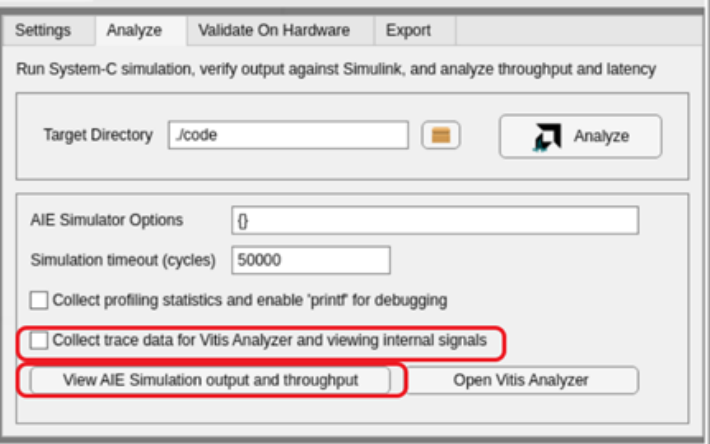
01

Select Internal AIE Signals

02

View Source and Destination Signal in Data Inspector (Simulink®)

03



Agenda

Introduction

Hardware Selection

Code Generation

Analyzing and Verifying Designs

Hardware Validation

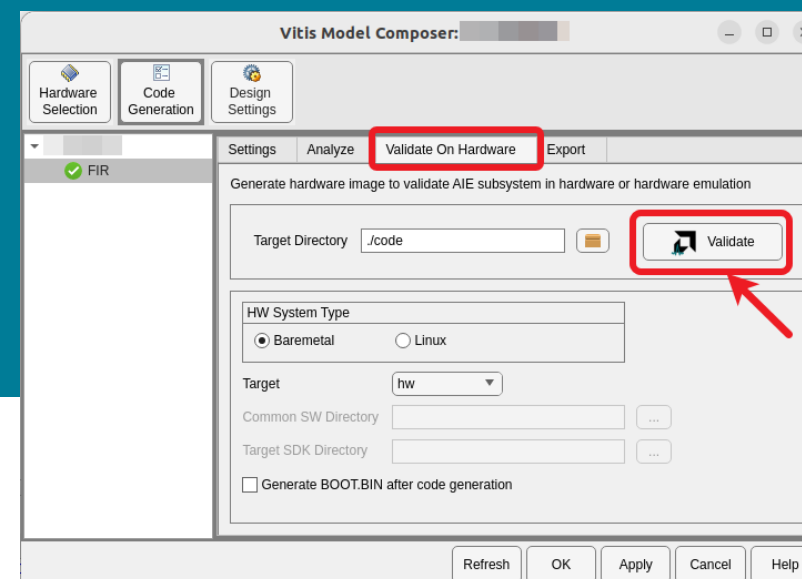
Summary

Hardware Validation Flow for AI Engines

- Provides a methodology to verify AI Engine-based applications on hardware (AMD Versal™ devices)
- Provides option to validate the generated hardware image targeting a specific platform for the Simulink® model
 - Hardware image can be:
 - Run on a board to verify whether the results from hardware match with the simulation output
 - Either bare-metal or Linux®-based

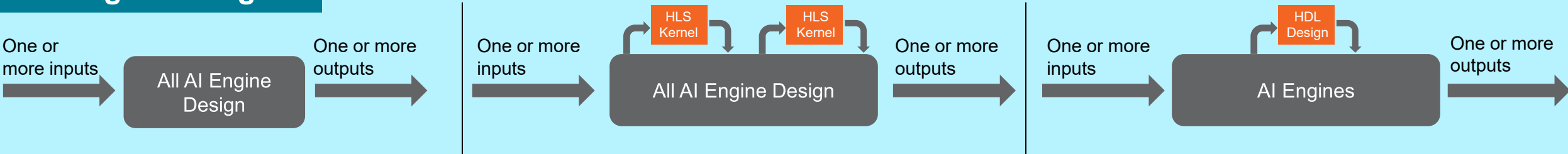
Hardware image can be generated for designs with:

- Only AI Engine blocks
- Both PL and AI Engines
- Only PL blocks

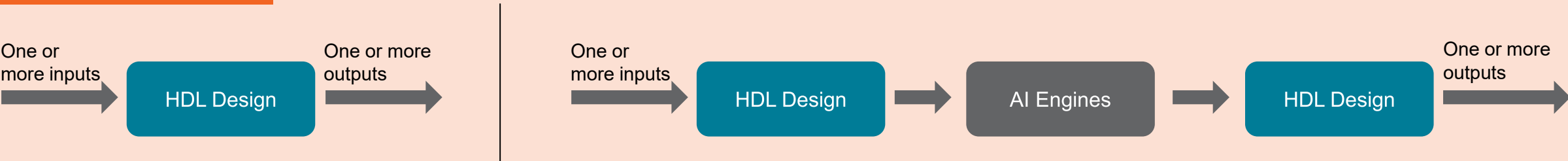


Topologies Supported for the Hardware Validation Flow

AI Engine Design



HDL Design



HLS Kernel



High-Level Flow for Generating a Hardware Image

Generate hardware image from the Simulink® tool:

- Requires an expandable platform (.xpfm) file
 - For hardware (e.g., VCK190), the platform files are shipped with the AMD Vitis™ software platform
- Expandable custom platform can be created for a custom board

Bare-metal applications:

- Tool generates a BOOT.BIN image file

Linux®-based applications:

- Tool generates an sd_card.img file

Hardware runs independently; no information is communicated between
Vitis Model Composer and the hardware

Design Considerations for Hardware Validation Flow

AXI4-Stream input and output ports of the subsystem must have a bit width in multiple of 8 bits, up to a maximum of 128 bits

Considerations for Designs with HLS Kernels

- HLS Kernels must have AXI4-Stream input and output ports
- HLS Kernel should be in free-running mode; accomplished by including the following pragma in the HLS function: `#pragma HLS INTERFACE ap_ctrl_none port=return`

Considerations for HLS-AI Engine Designs

- Only use HLS Kernel blocks to import C/C++ code (for PL) to connect with AI Engines
- Blocks from the HLS library are not allowed to connect to and co-simulate with the AI Engine
- Ensure the bit width of the HLS Kernel input or output that connects with the AI Engine matches the PLIO width of the AIE
- Ensure that there are no extra outputs from the subsystem that will not be in the hardware implementation, such as debug outputs to monitor internal signals
- If multiple HLS Kernel inputs are being driven by the same signal, the signal multiplexing must occur outside the hardware subsystem, so there are subsystem inputs for each HLS Kernel input

Agenda

Introduction

Hardware Selection

Code Generation

Analyzing and Verifying Designs

Hardware Validation

Summary

Summary

- 01** AMD Vitis™ Model Composer automatically compiles designs into low-level representations using the Vitis Model Composer Hub block
- 02** Vitis Model Composer Hub block controls the behavior of the Vitis Model Composer tool
- 03** Support for exporting different compilation types provides the freedom to choose a suitable representation for a design's environment
- 04** Enable the verification flow for HLS design by selecting the create testbench and run C simulation or C/RTL co-simulation options from the Vitis Model Composer Hub block
- 05** Verification of AI Engine code can be done via the Vitis Model Composer Hub block, profiling, and event tracing, as well as using the Vitis Analyzer tool to calculate throughput
- 06** Hardware validation flow can be performed by generating a hardware image targeting a specific platform for the Simulink® environment

GENERAL DISCLOSURE AND ATTRIBUTION STATEMENT

The information contained herein is for informational purposes only and is subject to change without notice. While every precaution has been taken in the preparation of this document, it may contain technical inaccuracies, omissions and typographical errors, and AMD is under no obligation to update or otherwise correct this information. Advanced Micro Devices, Inc. makes no representations or warranties with respect to the accuracy or completeness of the contents of this document, and assumes no liability of any kind, including the implied warranties of noninfringement, merchantability or fitness for particular purposes, with respect to the operation or use of AMD hardware, software or other products described herein. No license, including implied or arising by estoppel, to any intellectual property rights is granted by this document. Terms and limitations applicable to the purchase or use of AMD products are as set forth in a signed agreement between the parties or in AMD's Standard Terms and Conditions of Sale. GD-18u.

©2025 Advanced Micro Devices, Inc. All rights reserved. AMD, the AMD Arrow logo, UltraScale+, Versal, Vitis, Vivado, Zynq, and combinations thereof are trademarks of Advanced Micro Devices, Inc. Other product names used in this publication are for identification purposes only and may be trademarks of their respective owners. Certain AMD technologies may require third-party enablement or activation. Supported features may vary by operating system. Please confirm with the system manufacturer for specific features. No technology or product can be completely secure.

