MathWorks®

FPGA Meets Systems Engineering

Integrating Approaches for Space Applications

Stephan van Beek

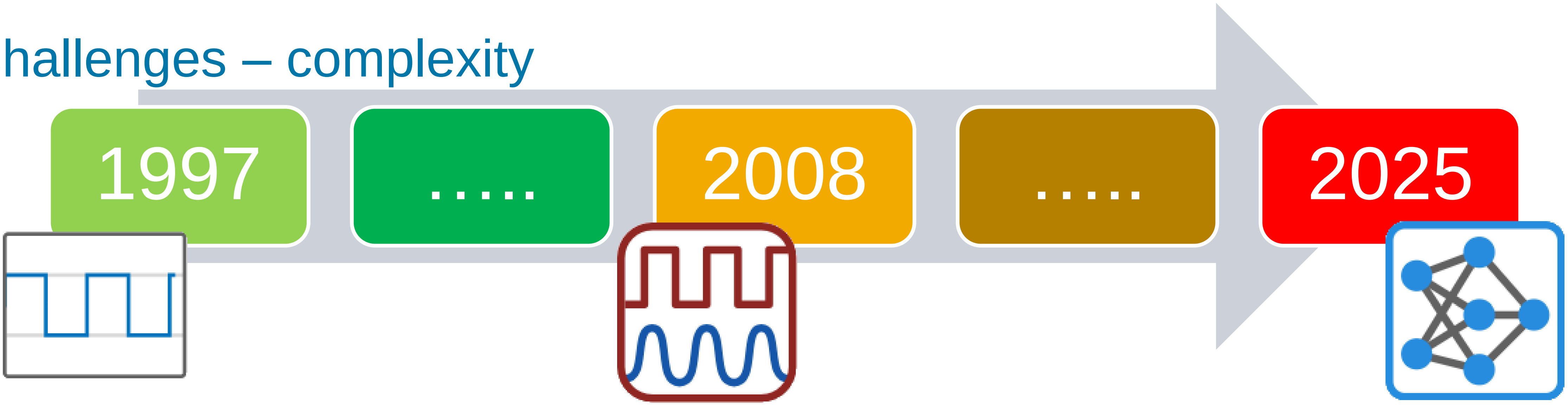
European Technical Specialist
SoC/FPGA Design Flows

The biggest problem was not the unit mismatch itself, but the failure to detect and correct this mistake

Mars Climate Orbiter

Image credit: NASA/JPL

Challenges – complexity



1997 → ..... → 2008 → ..... → 2025

1 FPGA engineer,
1 board with 4 FPGAs

- ○ Applications: logic interfaces, very simple algorithms
- ○ *Xilinx XC4000 resources* (**DSPs 0**, Logic Cells, RAM)
- ○ Schematic entry ➔ hybrid schematic entry + VHDL

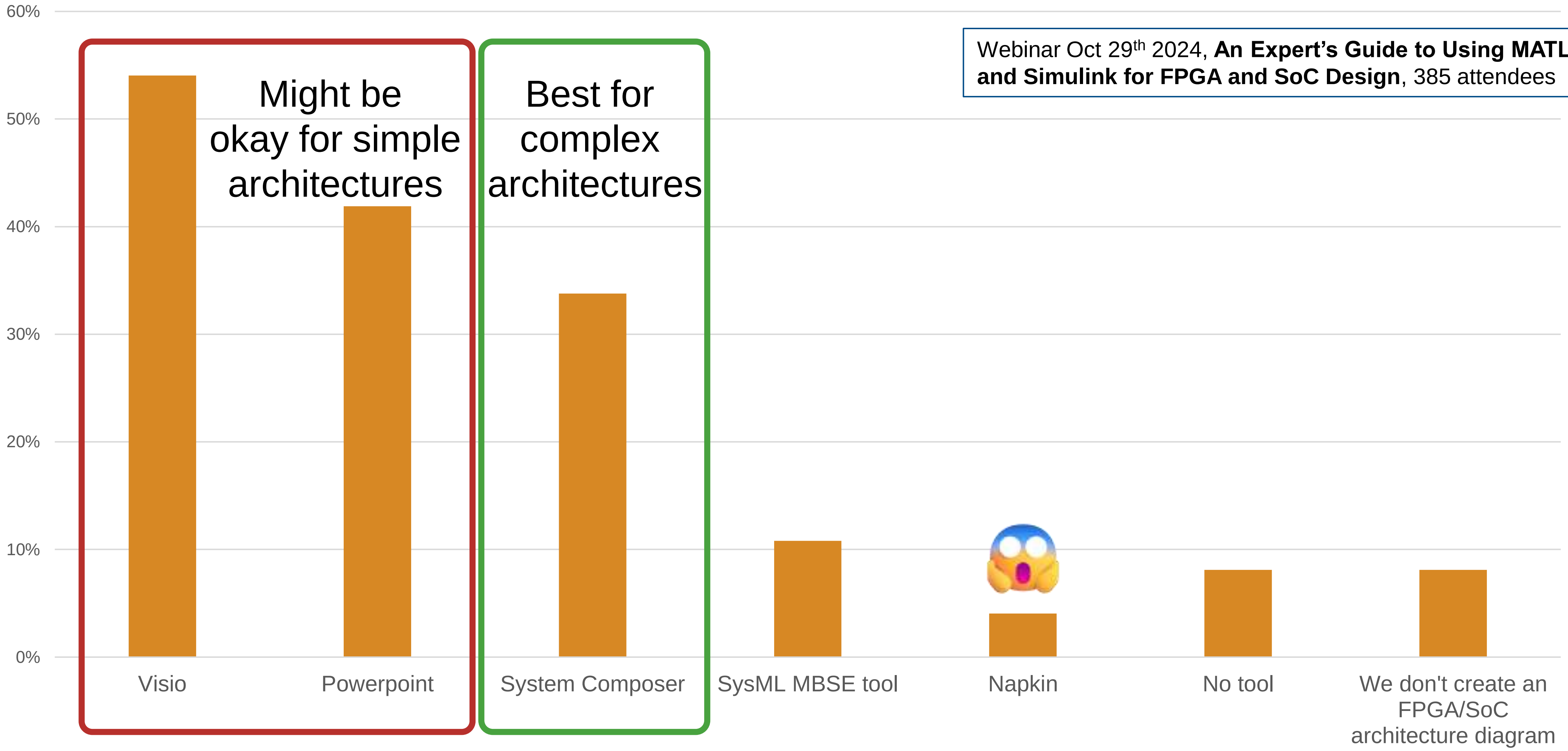
Multiple FPGA engineers,
1 board with 1 FPGA

- ○ Applications: more complex algorithms
- ○ *Xilinx Virtex-5 resources* (**DSPs**, Logic Cells, BRAM, **Embedded Processor**)
- ○ VHDL ➔ MBD (FPGA)

Team of engineers (algorithm, architect, FPGA, software) for 1 board with 1 SoC

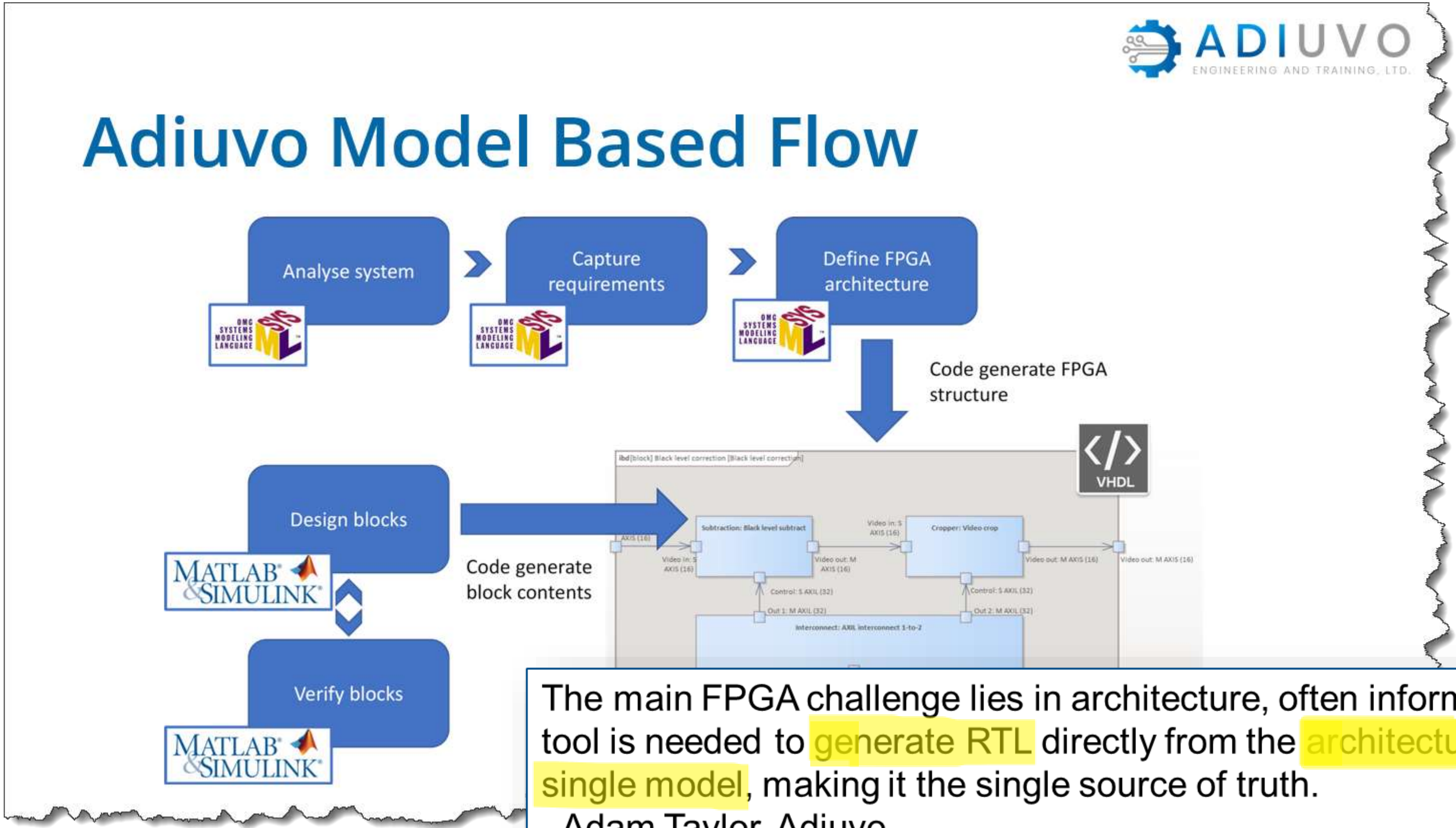
- ○ Applications: very complex algorithms (artificial intelligence, wireless, signal processing, vision, motor/power, etc.)
- ○ *AMD Versal resources* (**Logic Cells**, **DSP engines, AI engines,** BRAM, URAM, **ARM**)
- ○ Will this be the next evolution? ➔ MBD (SoC/FPGA) + MBSE??

How do you create your FPGA architectural block diagrams?



Webinar Oct 29th 2024, **An Expert's Guide to Using MATLAB and Simulink for FPGA and SoC Design**, 385 attendees

Adam Taylor is on a good path here ……



ADIUVO ENGINEERING AND TRAINING, LTD.

Adiuvo Model Based Flow

Analyse system

Capture requirements

Define FPGA architecture

Code generate FPGA structure

Design blocks

Verify blocks

Code generate block contents

VHDL

The main FPGA challenge lies in architecture, often informal and overlooked. A tool is needed to generate RTL directly from the architecture + design in a single model, making it the single source of truth.
- Adam Taylor, Adiuvo

Challenges – complexity ➔ needs

- Top-down design processes
  - Functional decomposition
  - No simulation needed early, but ….. later you will need simulation
- Go Beyond Textual Requirements
  - Use expressiveness of requirement models and trade studies
  - Use views to put stakeholder discussions in context
- Validate compliance to requirements through simulation
- Deployment
  - Generate RTL code from **architecture + design models**

Model-Based Systems Engineering + Model-Based Design

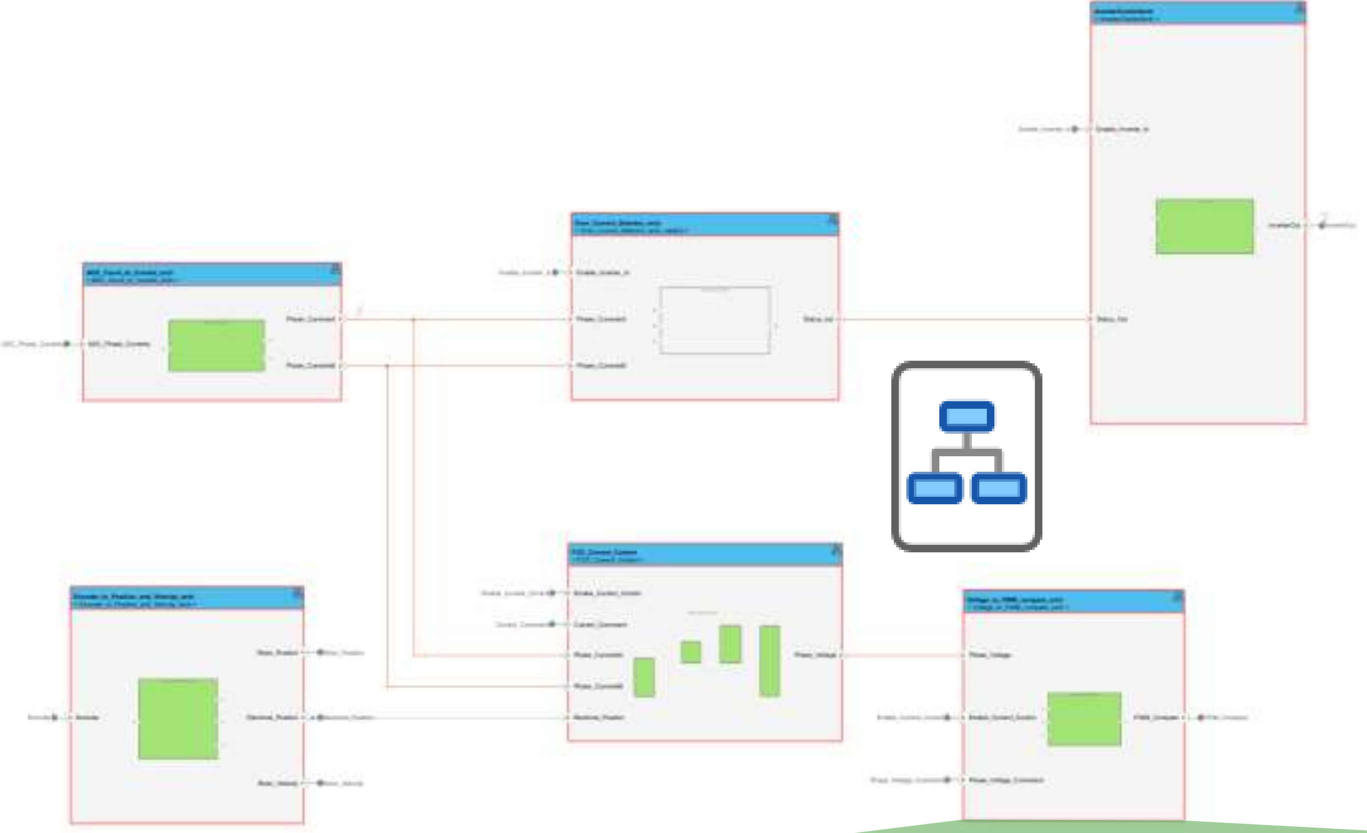
Top-down

Requirements

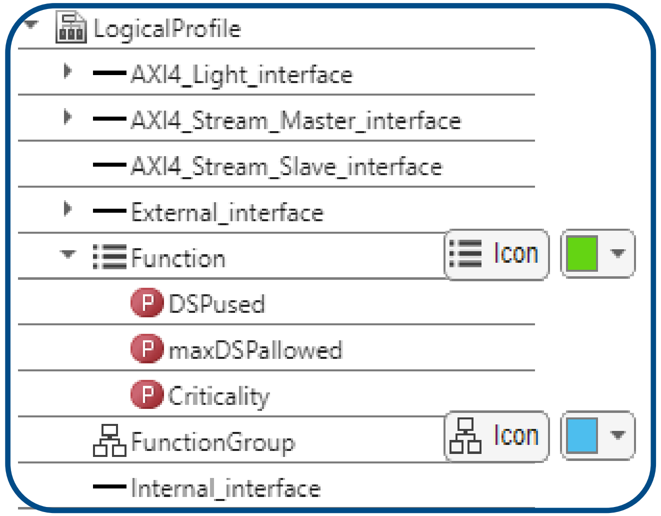
Architecture

Design

Bottom-up

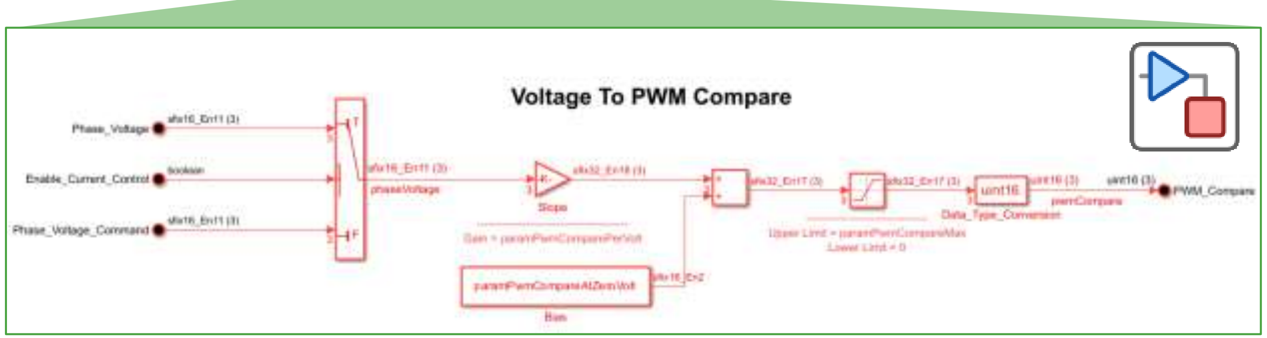


Custom Profiles

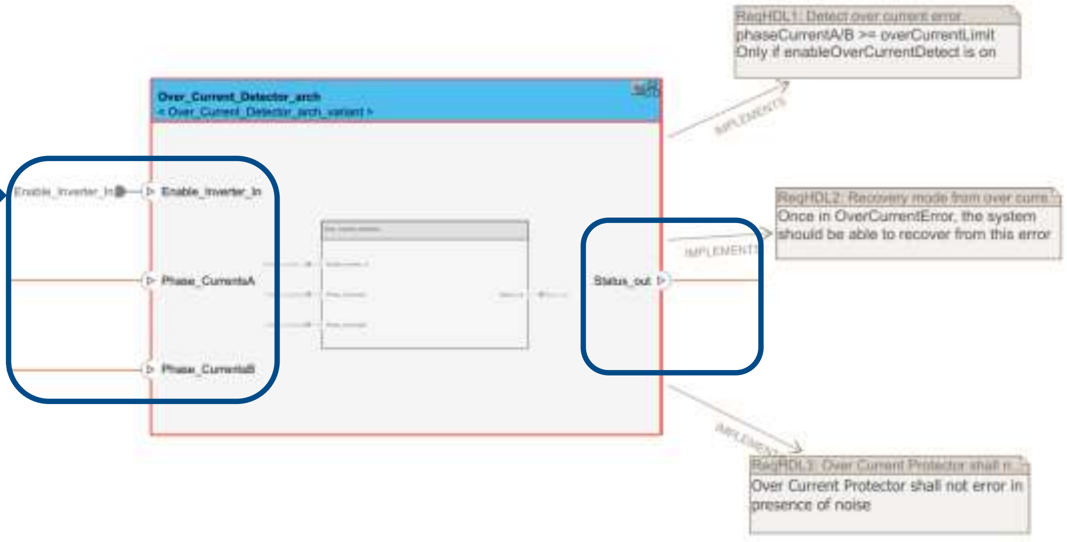


- LogicalProfile
  - AXI4_Light_interface
  - AXI4_Stream_Master_interface
  - AXI4_Stream_Slave_interface
  - External_interface
  - Function — Icon 🟩
    - DSPused
    - maxDSPallowed
    - Criticality
  - FunctionGroup — Icon 🟦
  - Internal_interface

Integrate MBD with MBSE

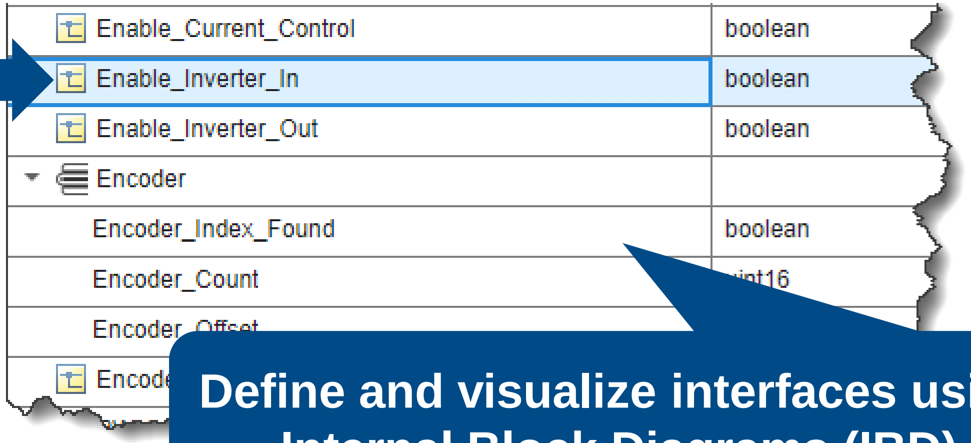


Requirements: Textual, Traceability, Interfaces, Interactions


Over_Current_Detector_arch
< Over_Current_Detector_arch_variant >
Enable_Inverter_In
Enable_Inverter_In
Phase_CurrentsA
Phase_CurrentsB
Status_out
IMPLEMENTS
IMPLEMENTS
IMPLEMENTS
ReqHDL1: Detect over current error
phaseCurrentA/B >= overCurrentLimit
Only if enableOverCurrentDetect is on
ReqHDL2: Recovery mode from over curre...
Once in OverCurrentError, the system should be able to recover from this error
ReqHDL3: Over Current Protector shall n...
Over Current Protector shall not error in presence of noise

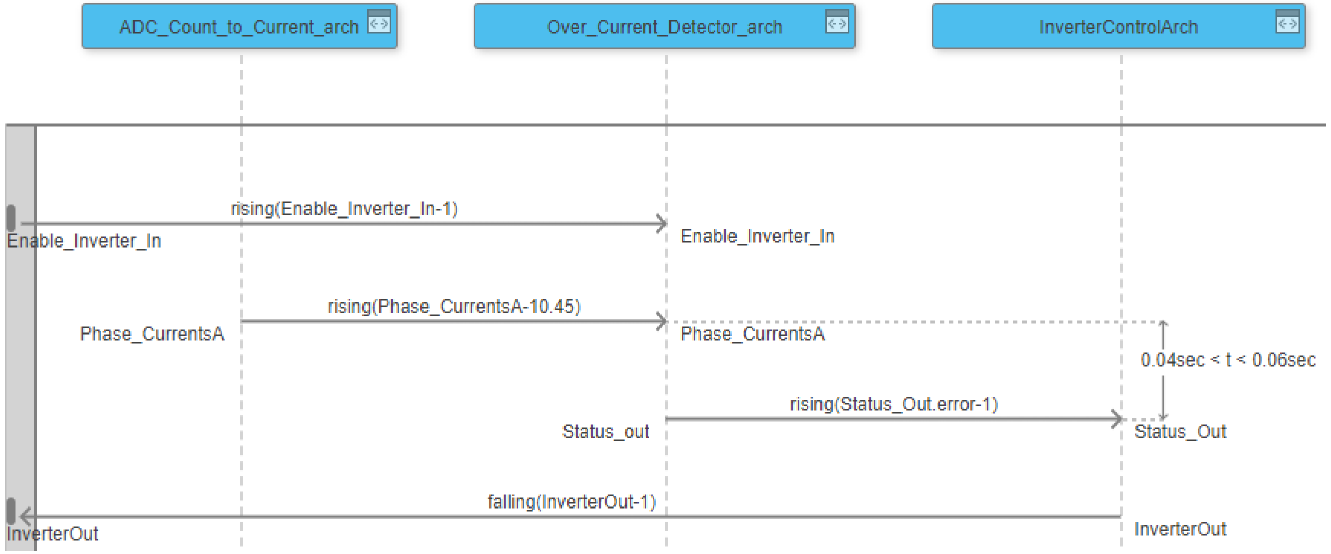
Establish traceability between architecture & design and textual requirements

Define interface behaviors using Sequence Diagrams (SD)

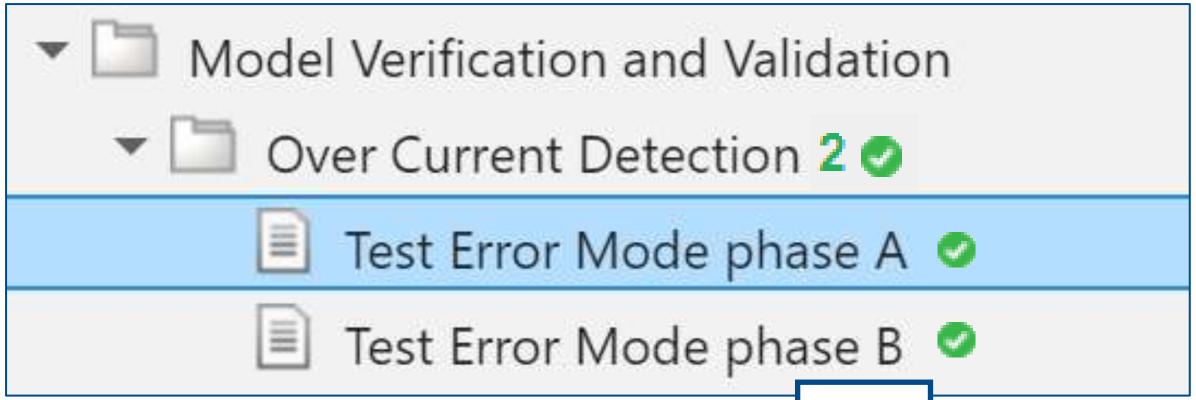


Enable_Current_Control	boolean
Enable_Inverter_In	boolean
Enable_Inverter_Out	boolean
Encoder	
Encoder_Index_Found	boolean
Encoder_Count	uint16
Encoder_Offset	
Encode	

Define and visualize interfaces using Internal Block Diagrams (IBD)



Validate Systems by Re-Using Requirements Models

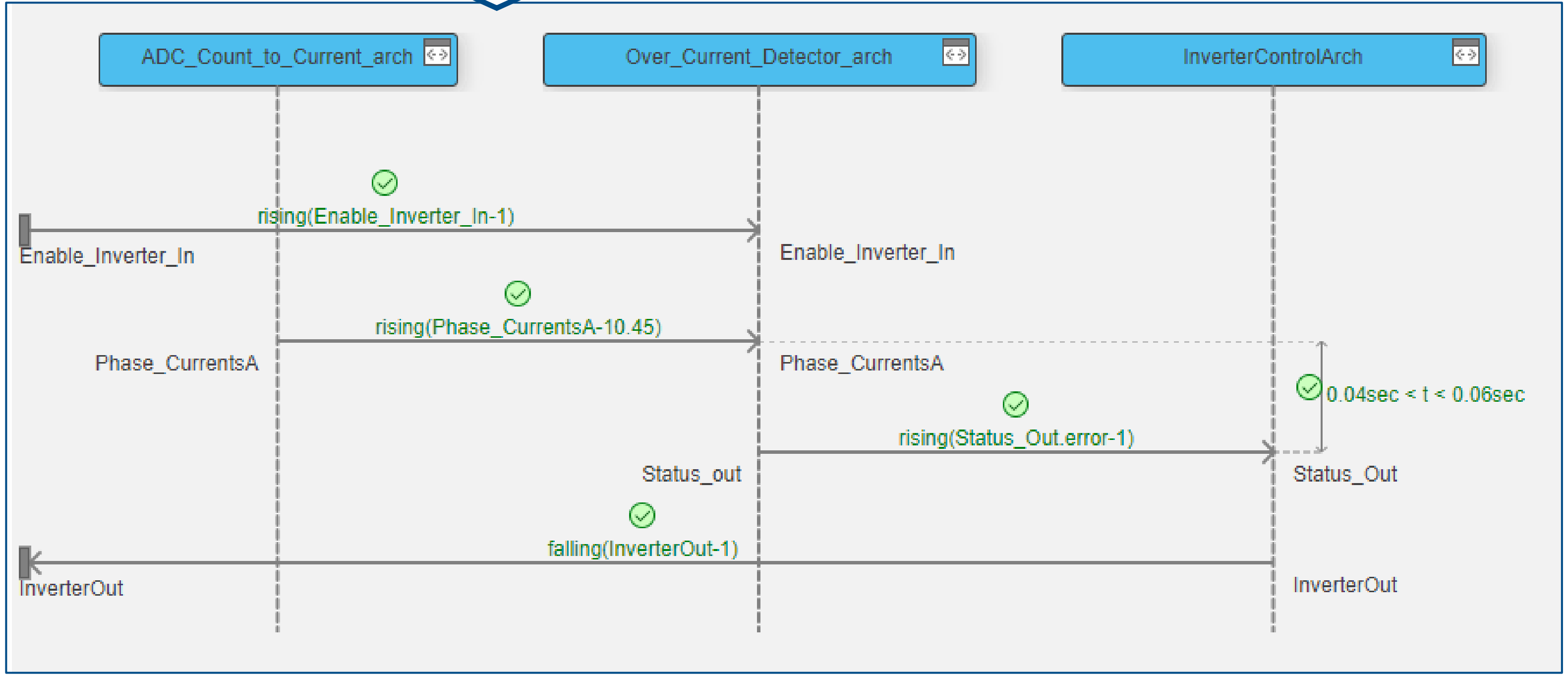


Model Verification and Validation

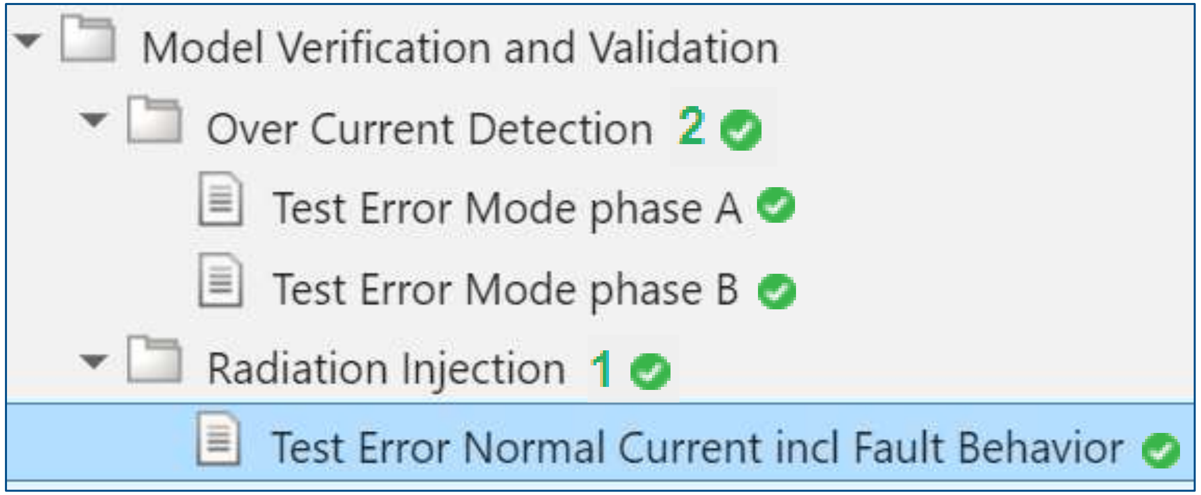
Over Current Detection 2 ✓

Test Error Mode phase A ✓

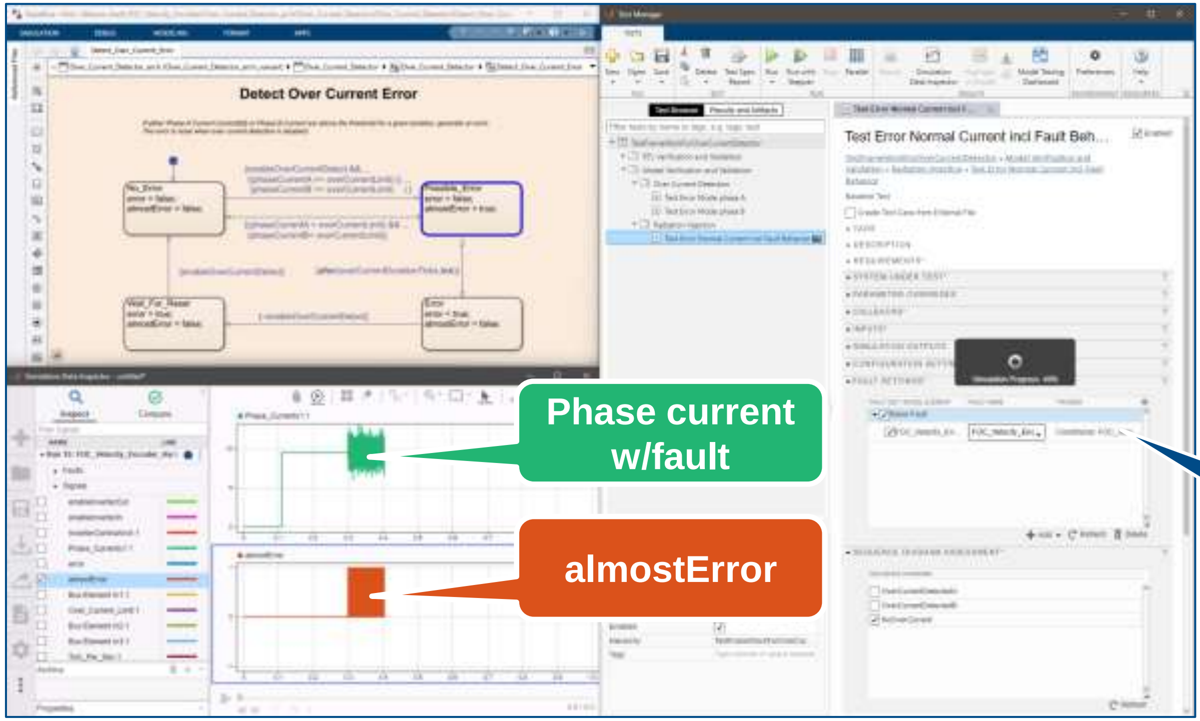
Test Error Mode phase B ✓



Validate System Behavior and Robustness with Fault Injection

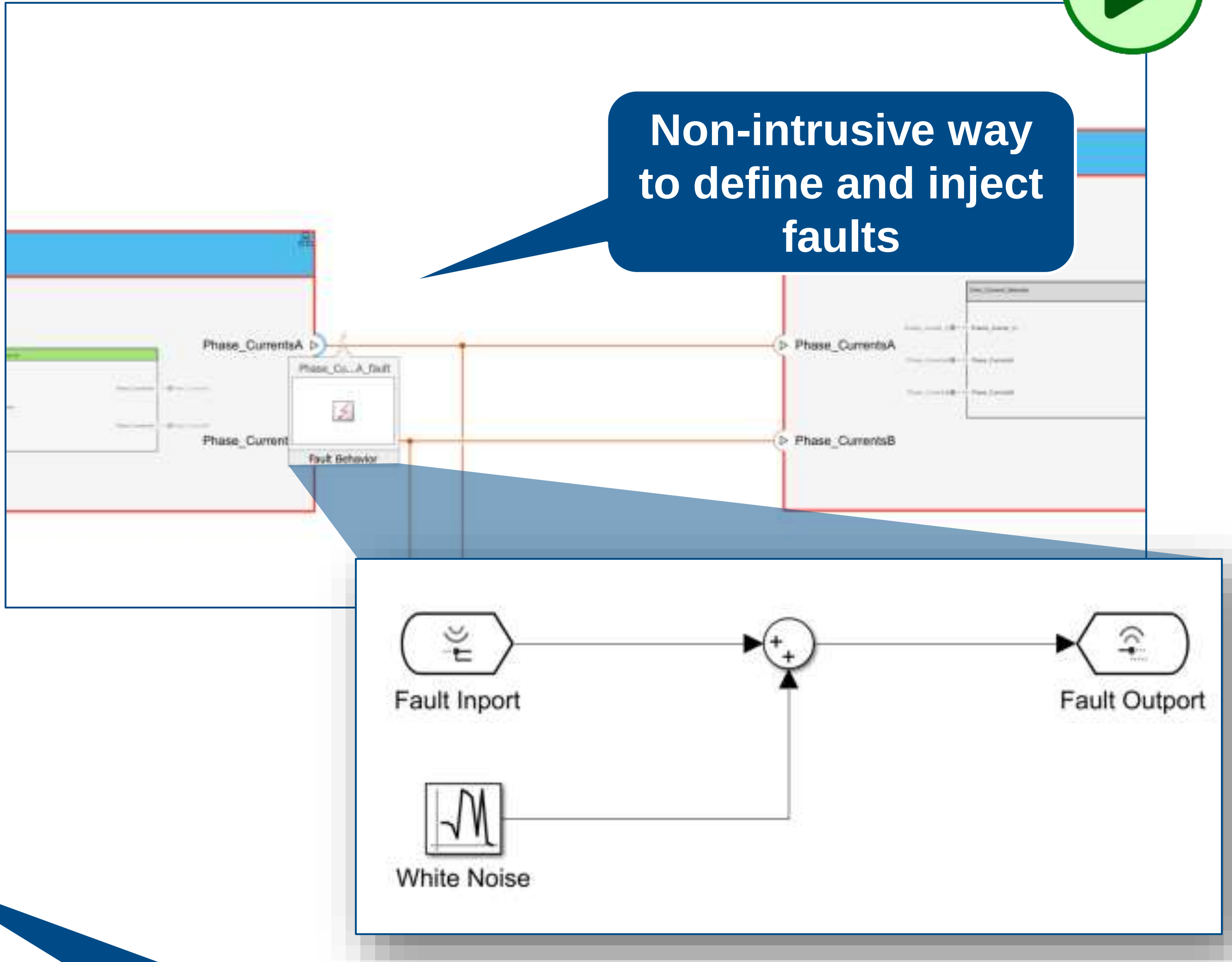


- Model Verification and Validation
  - Over Current Detection 2 ✓
    - Test Error Mode phase A ✓
    - Test Error Mode phase B ✓
  - Radiation Injection 1 ✓
    - Test Error Normal Current incl Fault Behavior ✓


Detect Over Current Error
Test Manager
Test Error Normal Current incl Fault Beh...
No_Error
Wait_For_Reset
Error
Phase_Currents1
almostError

Phase current w/fault

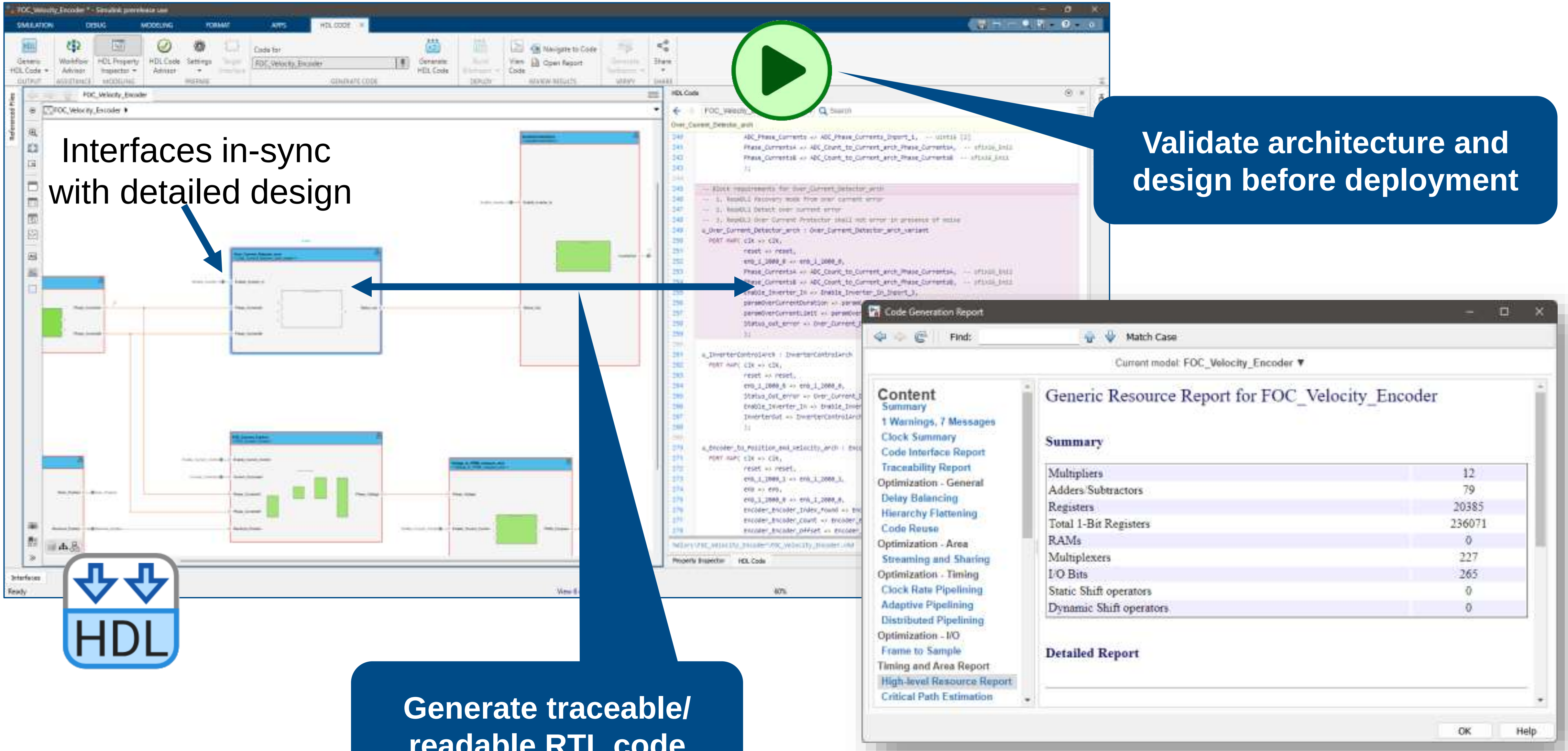
almostError



Non-intrusive way to define and inject faults

Analyze impact of faults using simulation

Generate HDL code from System Architecture incl Detailed Models



Interfaces in-sync with detailed design

Validate architecture and design before deployment

Generate traceable/ readable RTL code

Code Generation Report

Find: Match Case

Current model: FOC_Velocity_Encoder

Content

- Summary
- 1 Warnings, 7 Messages
- Clock Summary
- Code Interface Report
- Traceability Report
- Optimization - General
- Delay Balancing
- Hierarchy Flattening
- Code Reuse
- Optimization - Area
- Streaming and Sharing
- Optimization - Timing
- Clock Rate Pipelining
- Adaptive Pipelining
- Distributed Pipelining
- Optimization - I/O
- Frame to Sample
- Timing and Area Report
- High-level Resource Report
- Critical Path Estimation

Generic Resource Report for FOC_Velocity_Encoder

Summary

Multipliers	12
Adders/Subtractors	79
Registers	20385
Total 1-Bit Registers	236071
RAMs	0
Multiplexers	227
I/O Bits	265
Static Shift operators	0
Dynamic Shift operators	0

Detailed Report

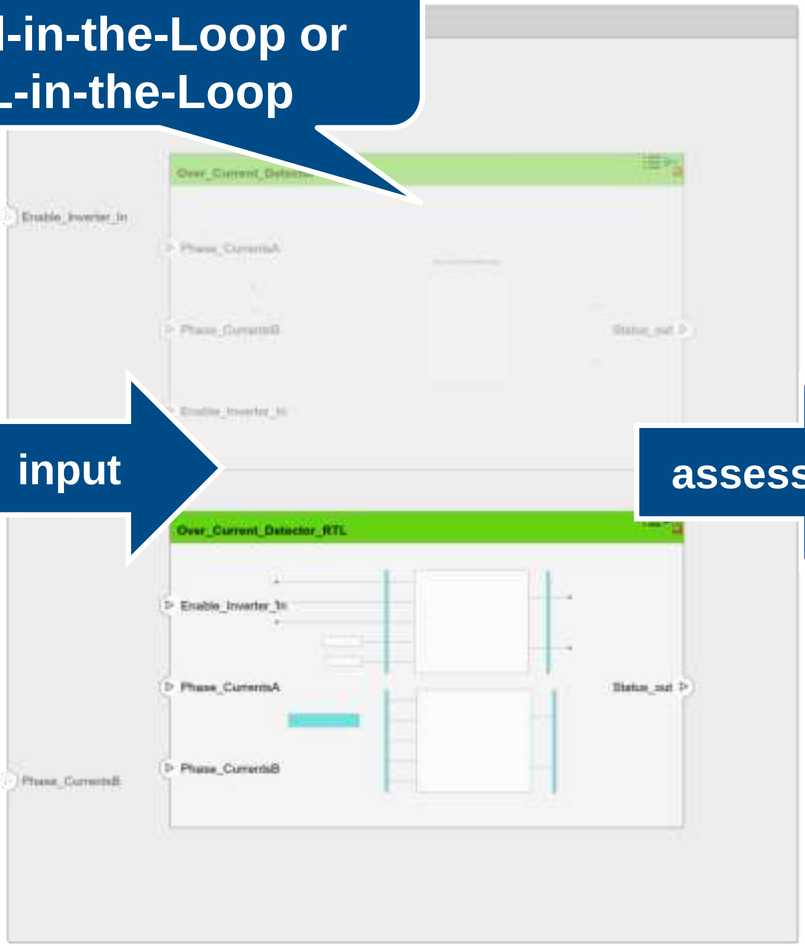
OK Help

Re-use System Architecture and Design Models for RTL Verification & Validation

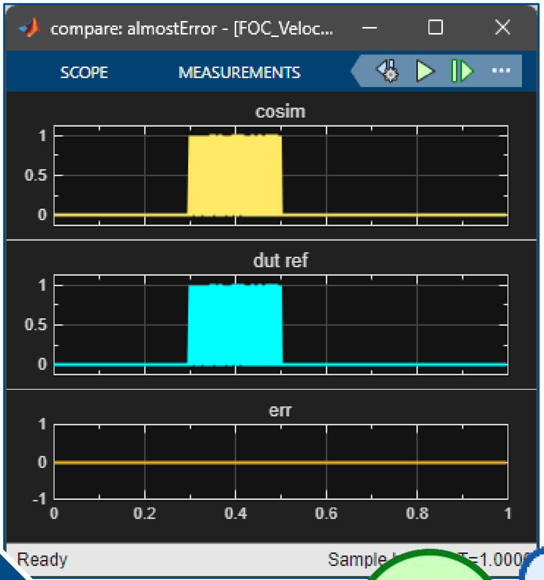
Variants to select Model-in-the-Loop or RTL-in-the-Loop

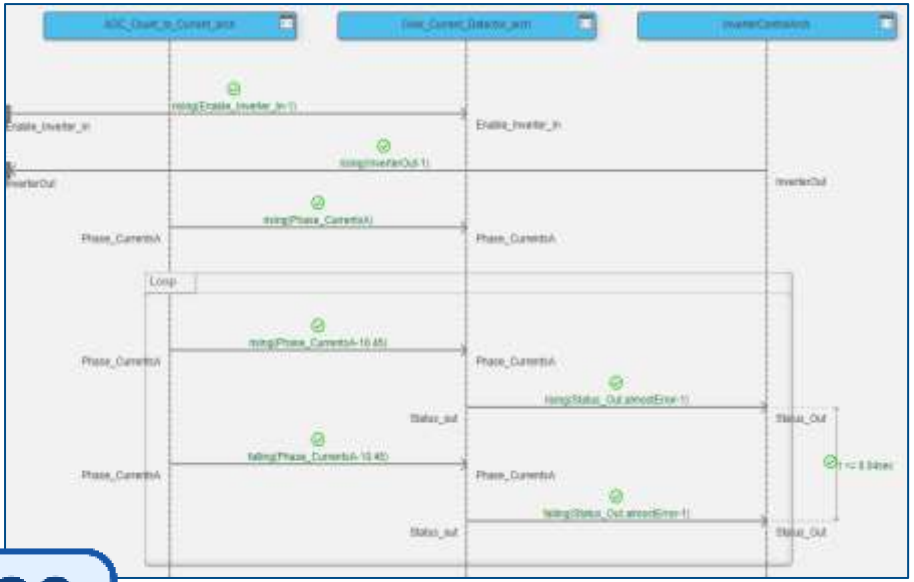
Excel file (input)

input

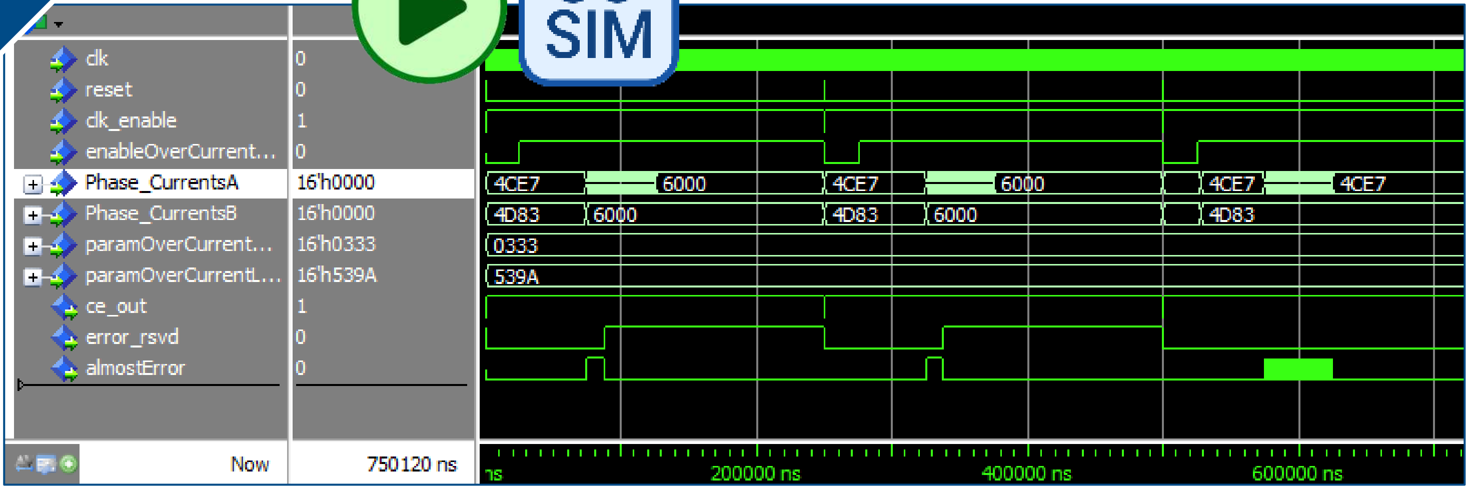


assess



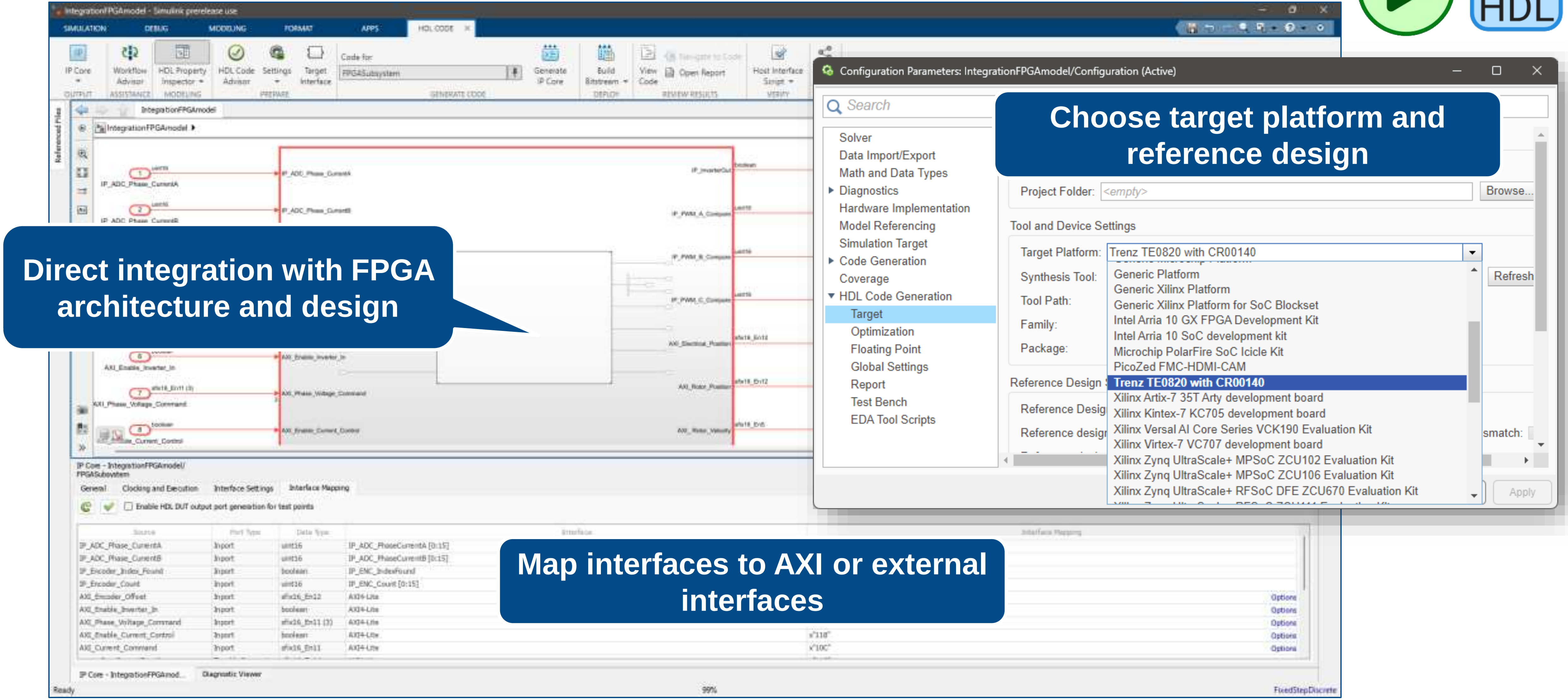


CO SIM



clk	0
reset	0
clk_enable	1
enableOverCurrent...	0
Phase_CurrentsA	16'h0000
Phase_CurrentsB	16'h0000
paramOverCurrent...	16'h0333
paramOverCurrentL...	16'h539A
ce_out	1
error_rsvd	0
almostError	0
Now	750120 ns

Integration with other IPs or Software Components



☐ Enable HDL DUT output port generation for test points

Source	Port Type	Data Type	Interface	Interface Mapping
IP_ADC_Phase_CurrentA	Inport	uint16	IP_ADC_PhaseCurrentA [0:15]	
IP_ADC_Phase_CurrentB	Inport	uint16	IP_ADC_PhaseCurrentB [0:15]	
IP_Encoder_Index_Found	Inport	boolean	IP_ENC_IndexFound	
IP_Encoder_Count	Inport	uint16	IP_ENC_Count [0:15]	
AXI_Encoder_Offset	Inport	sfix16_En12	AXI4-Lite	
AXI_Enable_Inverter_In	Inport	boolean	AXI4-Lite	
AXI_Phase_Voltage_Command	Inport	sfix16_En11 (3)	AXI4-Lite	
AXI_Enable_Current_Control	Inport	boolean	AXI4-Lite	x"118"
AXI_Current_Command	Inport	sfix16_En11	AXI4-Lite	x"10C"

Configuration Parameters: IntegrationFPGAmodel/Configuration (Active)

- Solver
- Data Import/Export
- Math and Data Types
- Diagnostics
- Hardware Implementation
- Model Referencing
- Simulation Target
- Code Generation
- Coverage
- HDL Code Generation
  - Target
  - Optimization
  - Floating Point
  - Global Settings
  - Report
  - Test Bench
  - EDA Tool Scripts

Project Folder: <empty> Browse...

Tool and Device Settings

Target Platform: Trenz TE0820 with CR00140

Synthesis Tool: Refresh

Tool Path:

Family:

Package:

Reference Design

- Generic Platform
- Generic Xilinx Platform
- Generic Xilinx Platform for SoC Blockset
- Intel Arria 10 GX FPGA Development Kit
- Intel Arria 10 SoC development kit
- Microchip PolarFire SoC Icicle Kit
- PicoZed FMC-HDMI-CAM
- **Trenz TE0820 with CR00140**
- Xilinx Artix-7 35T Arty development board
- Xilinx Kintex-7 KC705 development board
- Xilinx Versal AI Core Series VCK190 Evaluation Kit
- Xilinx Virtex-7 VC707 development board
- Xilinx Zynq UltraScale+ MPSoC ZCU102 Evaluation Kit
- Xilinx Zynq UltraScale+ MPSoC ZCU106 Evaluation Kit
- Xilinx Zynq UltraScale+ RFSoC DFE ZCU670 Evaluation Kit

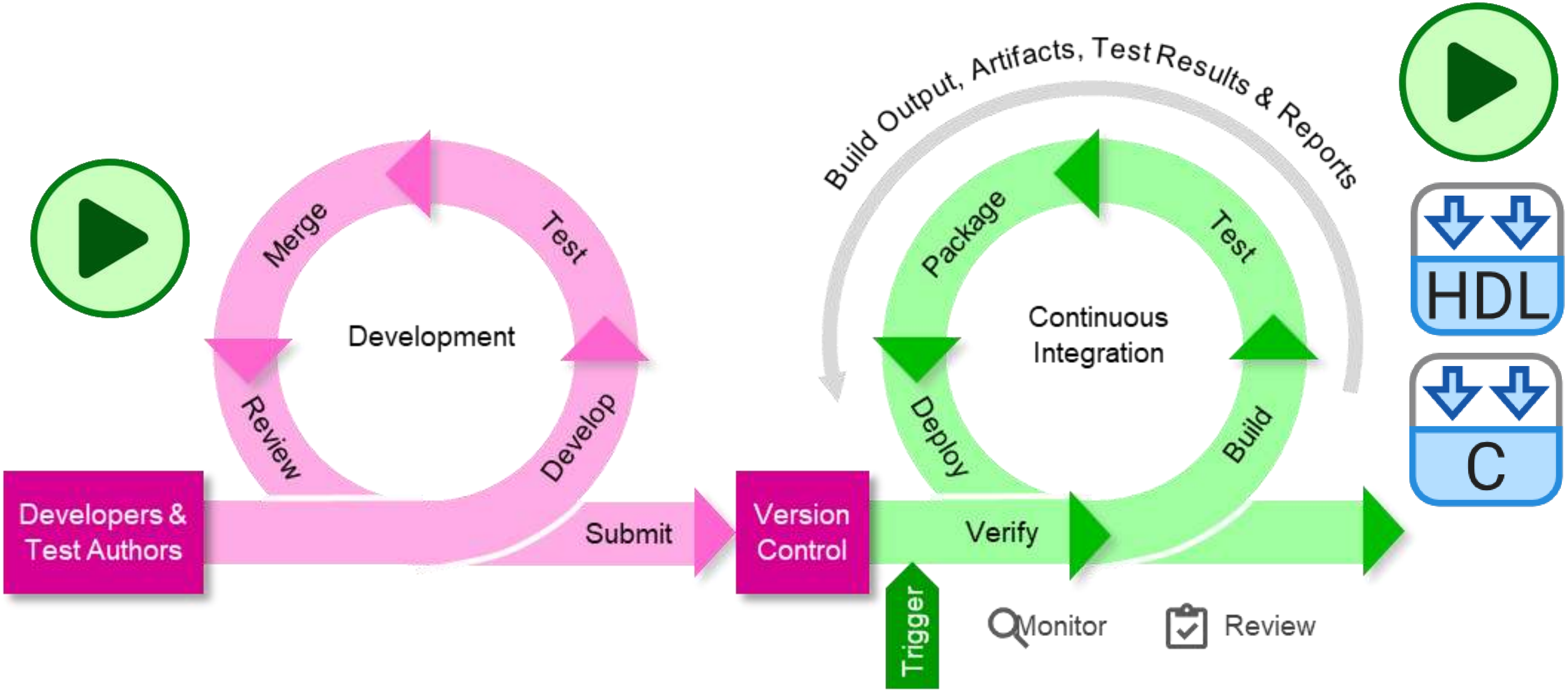
Apply

Choose target platform and reference design

Direct integration with FPGA architecture and design

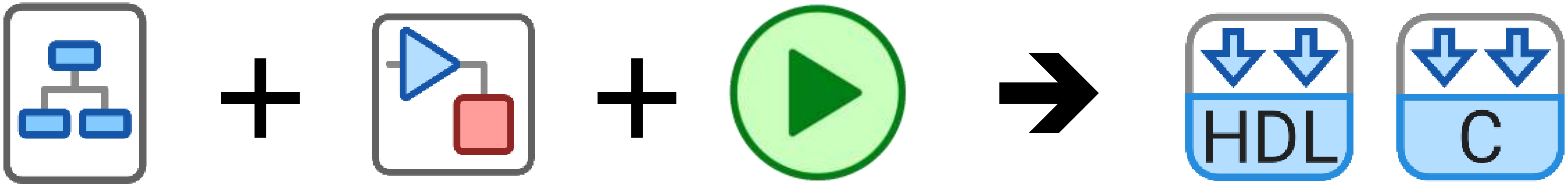
Map interfaces to AXI or external interfaces

Continuous Integration for Model-Based Design



Continuous Integration: CI/CD Automation for Model-Based Design (support package)
CI/CD Automation for Simulink Check (reference book)

Concluding remark



Architecture Design Simulation

> The main FPGA challenge lies in architecture, often informal and overlooked. A tool is needed to generate RTL directly from the architecture + design in a single model, making it the single source of truth.
> - Adam Taylor, Adiuvo

Simulated