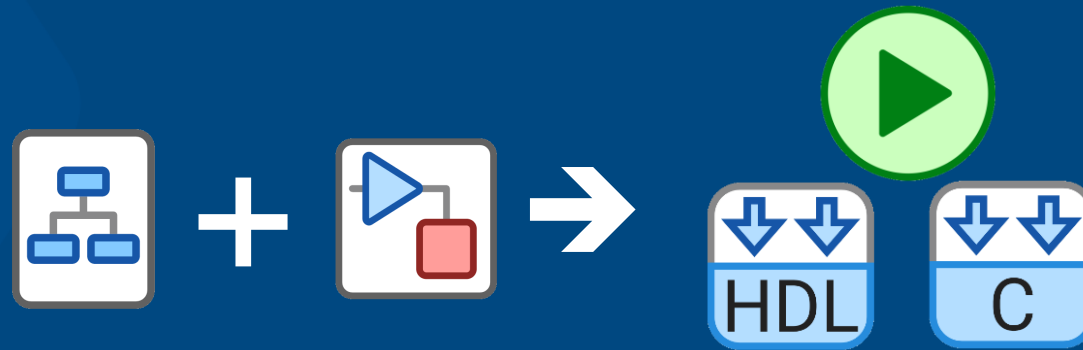




FPGA Meets Systems Engineering

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



1 FPGA engineer, 1 board with 4 FPGAs

- Applications: logic interfaces, very simple algorithms
- *Xilinx XC4000E* resources (**DSPs 0**, Logic Cells 85K, BRAM)
- Schematic entry → hybrid schematic entry + VHDL

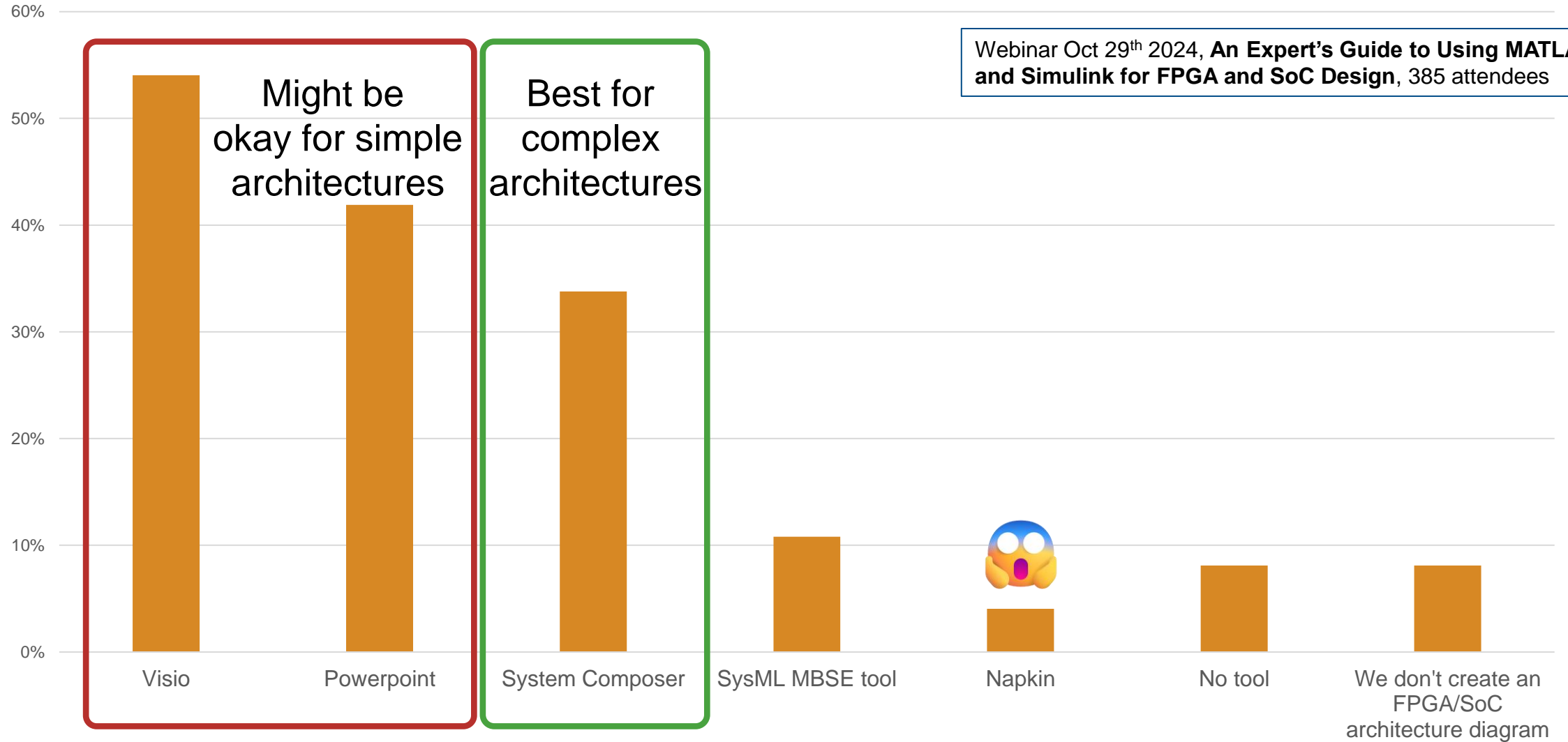
Multiple FPGA engineers, 1 board with 1 FPGA

- Applications: more complex algorithms
- *Xilinx Virtex-5* resources (**DSPs 1K**, Logic Cells 330K, 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 9M**, **DSP engines 11K**, **AI engines 400**, BRAM, URAM, **ARM**)
- Will this be the next evolution? → MBD (SoC/FPGA) + MBSE??

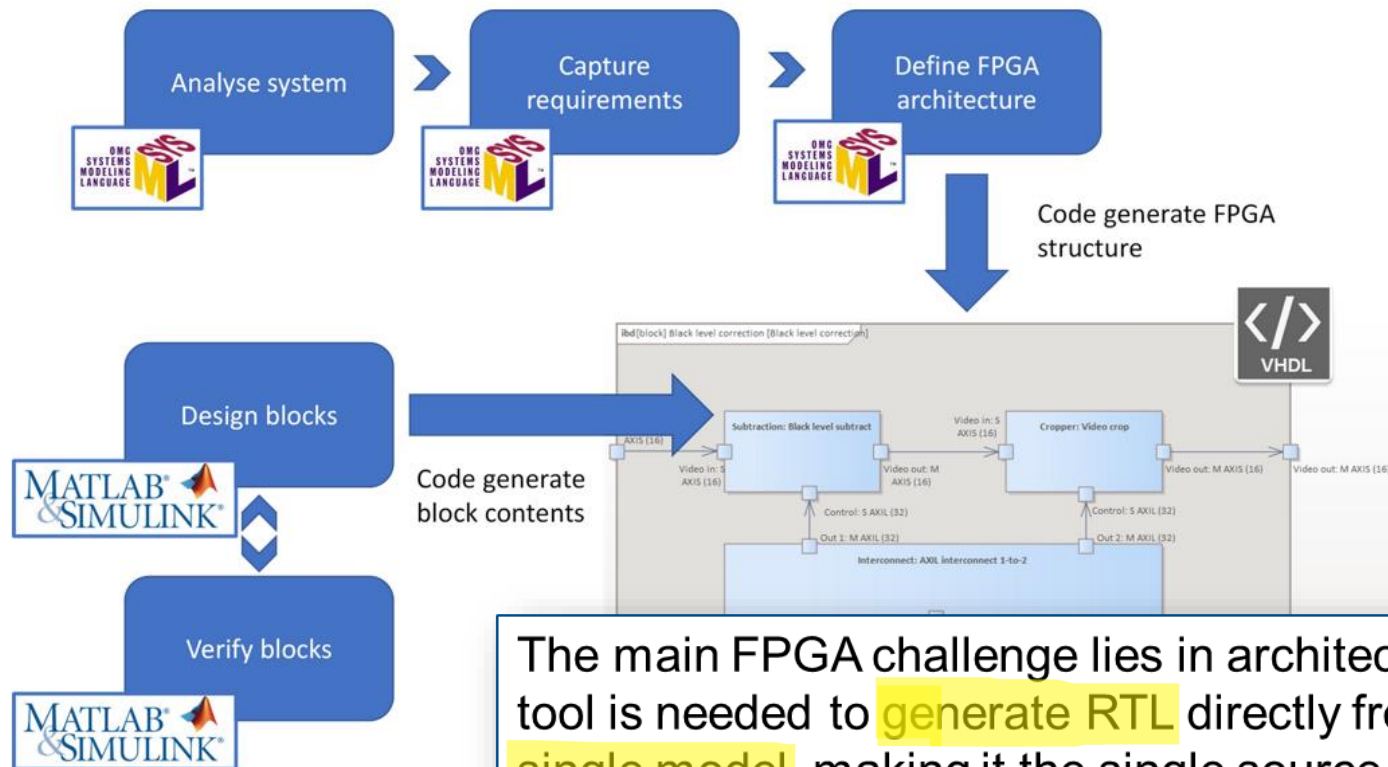
How do you create your FPGA architectural block diagrams?



Adam Taylor is on a good path here



Adiuvo Model Based Flow



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

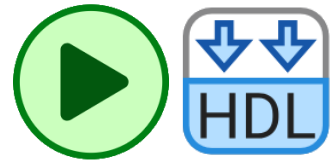
- Top-down design processes
 - Functional decomposition
 - No simulation needed early, but later you will need simulation

- 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

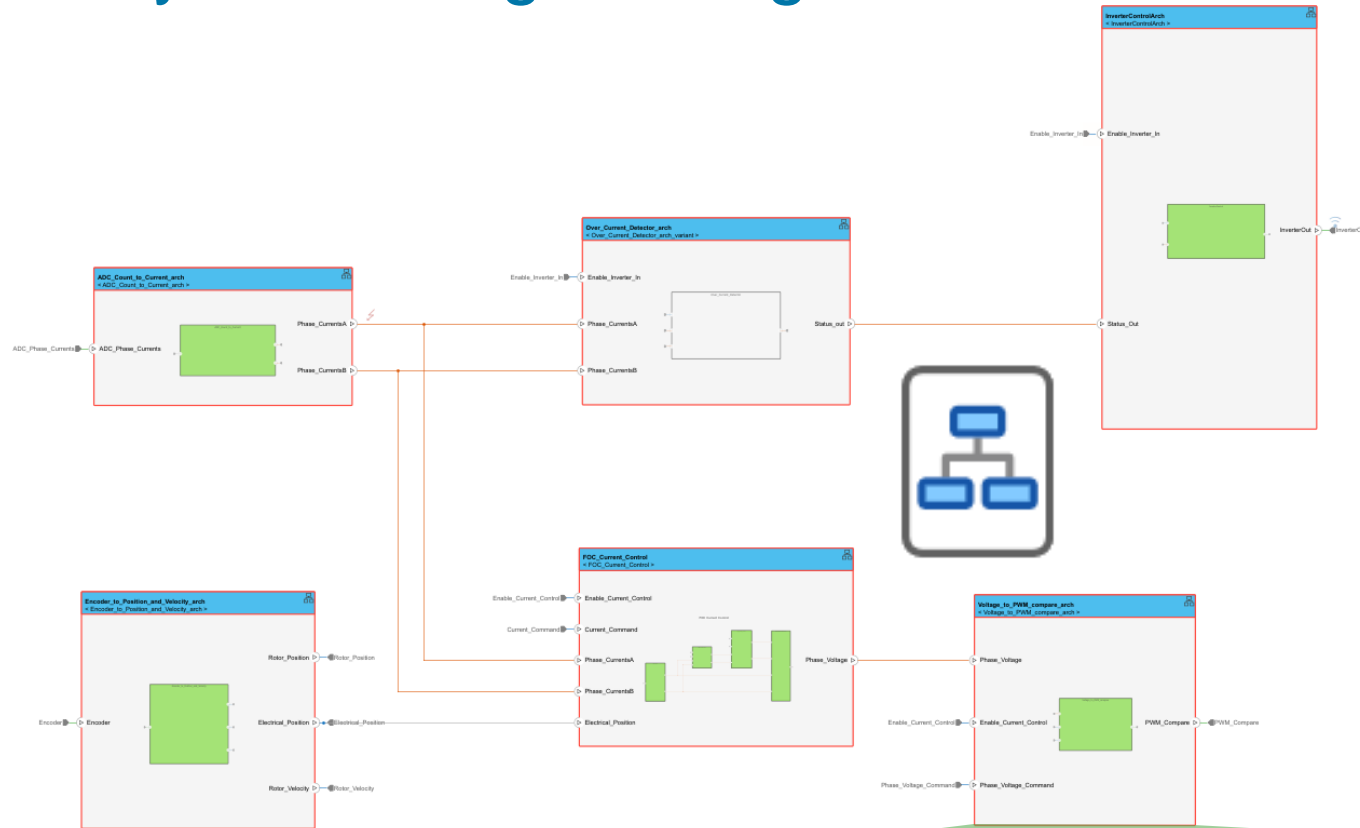


Top-down
Bottom-up

Requirements

Architecture

Design

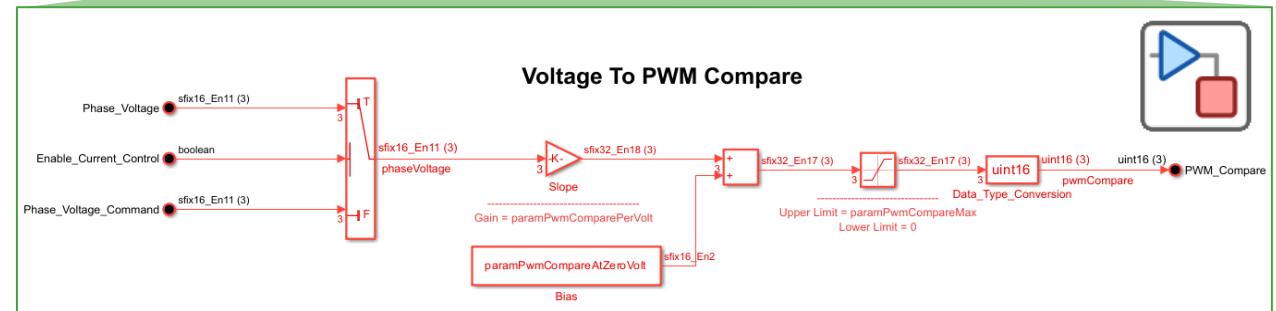


LogicalProfile

- AXI_interface
- ExternalInterface
- Function
 - Multiplier
 - Adder
 - Multiplexer
 - Register_1_bit
 - Criticality
- FunctionGroup
- InternalInterface

Custom Profiles

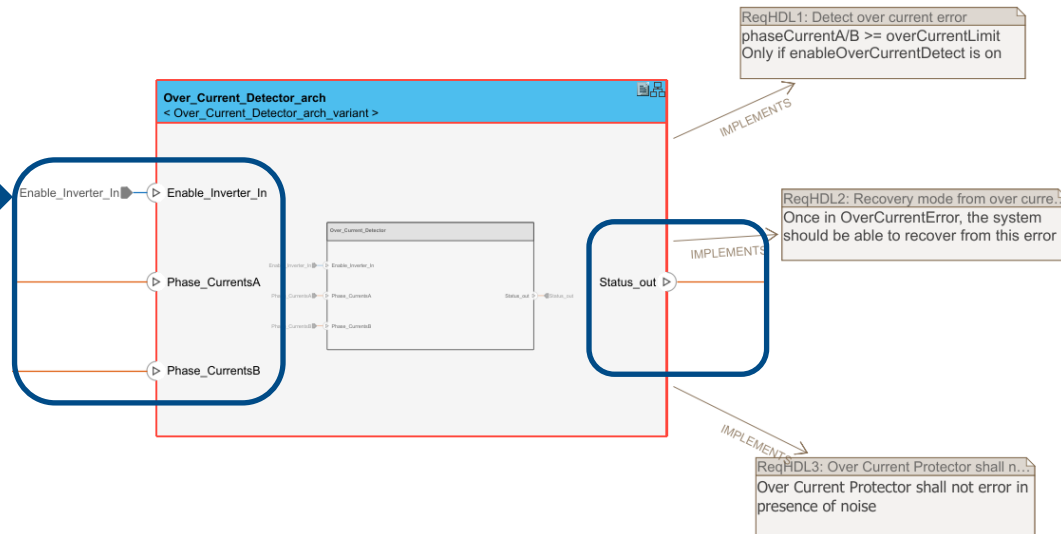
Integrate MBD with MBSE



Requirements: Textual, Traceability, Interfaces, Interactions

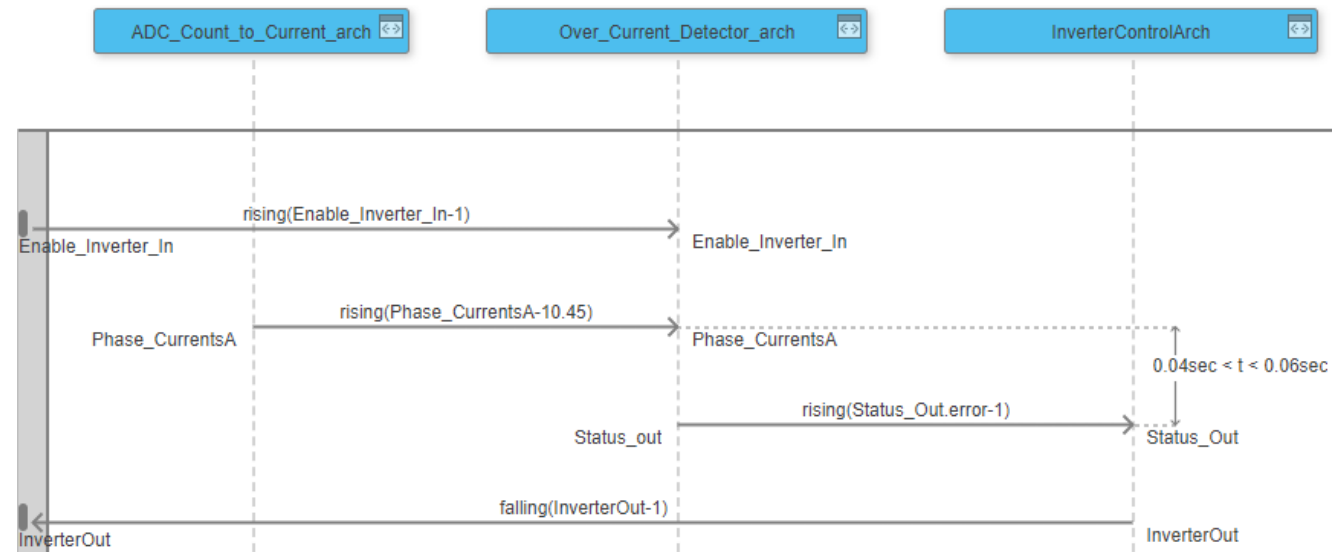
Establish traceability between architecture & design and textual requirements

Define interface behaviors using Sequence Diagrams (SD)



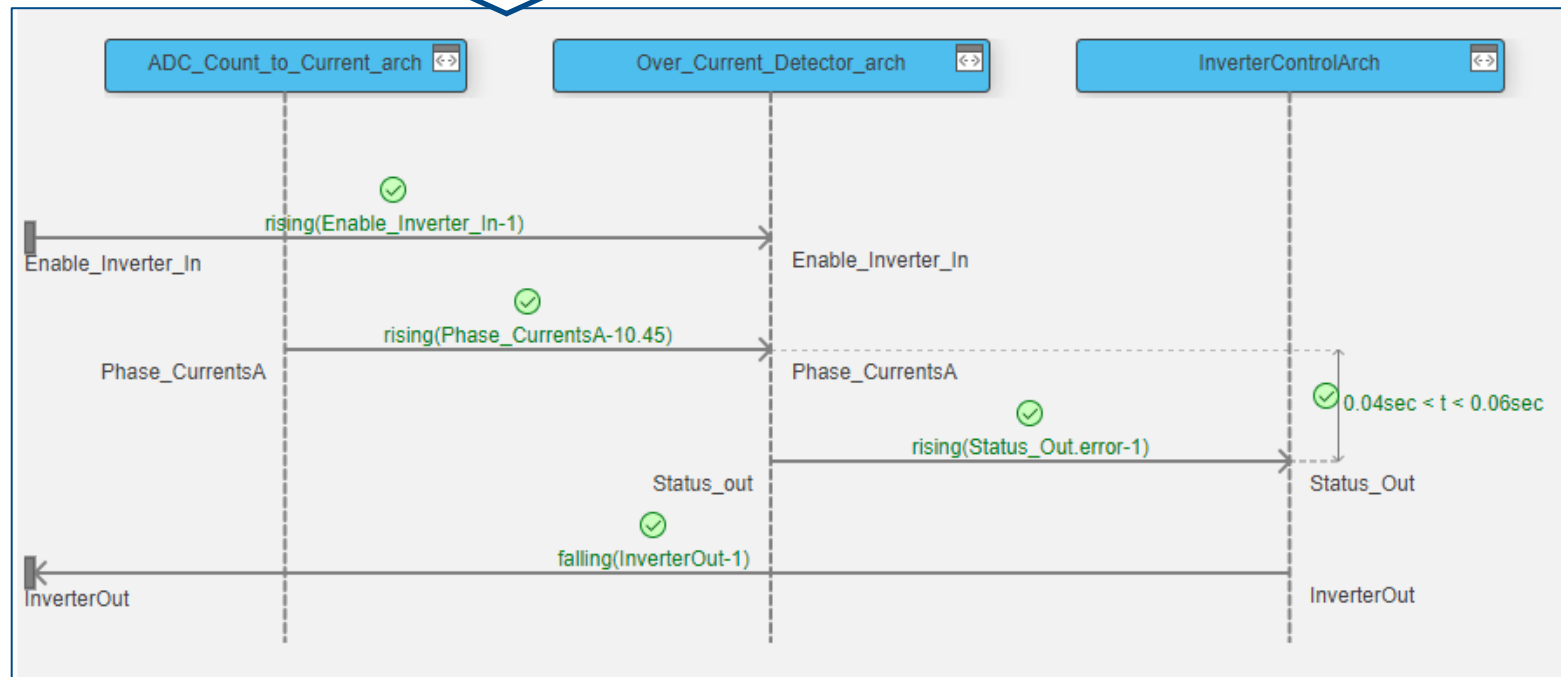
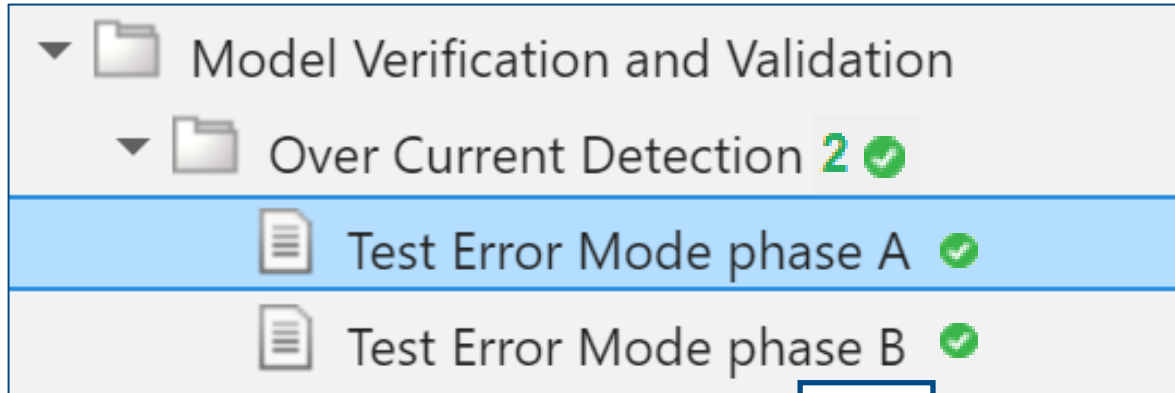
Enable_Current_Control	boolean
Enable_Inverter_In	boolean
Enable_Inverter_Out	boolean
Encoder	
Encoder_Index_Found	boolean
Encoder_Count	uint16
Encoder_Offset	
Encoder	

Define and visualize interfaces using Internal Block Diagrams (IBD)





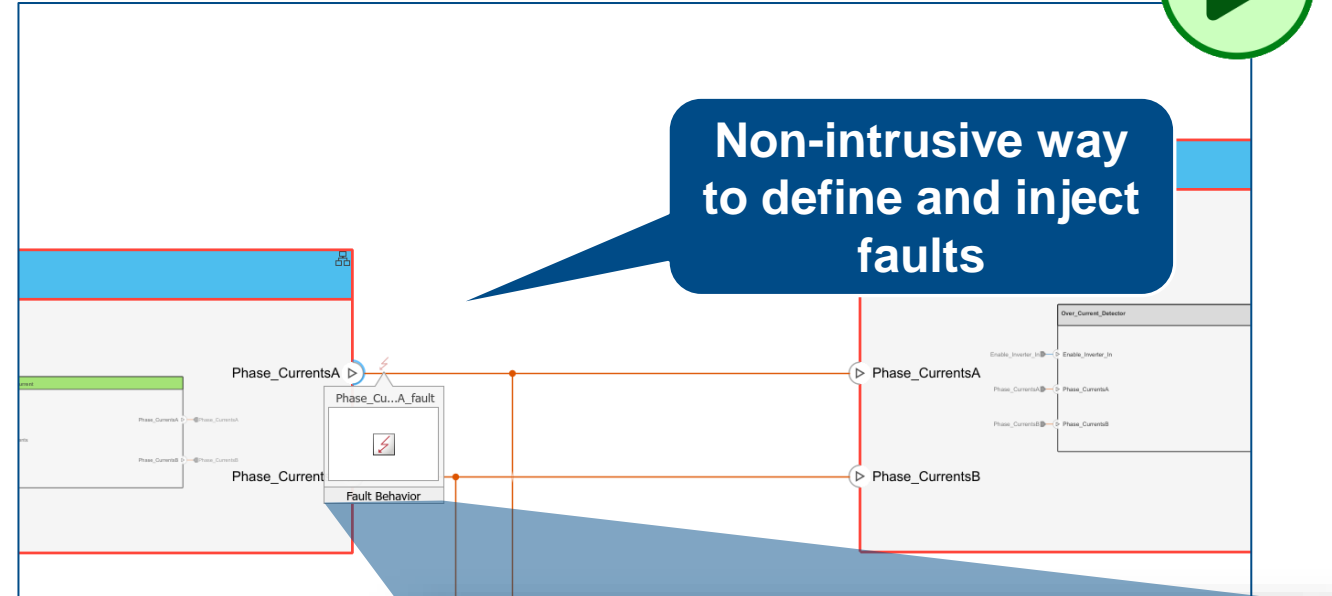
Validate Systems by Re-Using Requirements Models



Validate System Behavior and Robustness with Fault Injection

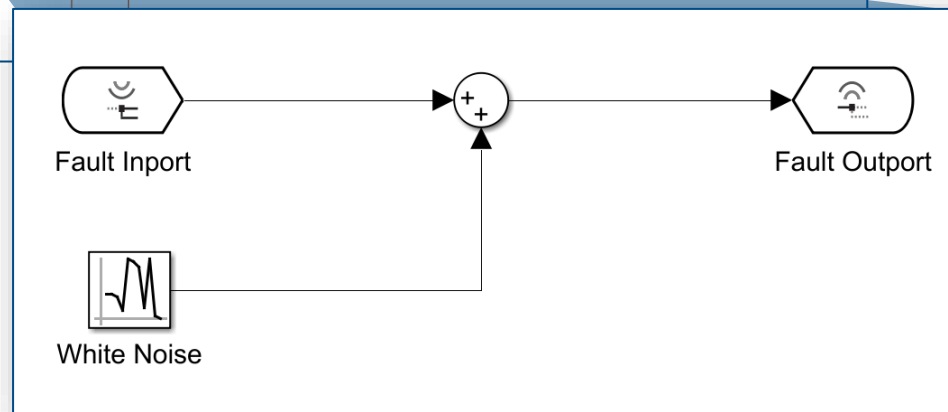


- Model Verification and Validation
 - Over Current Detection **2**
 - Test Error Mode phase A
 - Test Error Mode phase B
- Radiation Injection **1**
- Test Error Normal Current incl Fault Behavior



Phase current w/fault

almostError



Analyze impact of faults using simulation

Generate HDL code from System Architecture incl Detailed Models

The screenshot shows the HDL Code window in Simulink. On the left, a system architecture diagram is visible with several blocks connected. A callout box points to these blocks with the text "Interfaces in-sync with detailed design". On the right, the HDL code is displayed, showing Verilog-like code for an over-current detector. A green play button icon is overlaid on the code window. At the bottom left, there is a blue icon with two downward arrows and the text "HDL".

Validate architecture and design before deployment

Generate traceable/readable RTL code

The screenshot shows the Code Generation Report window for the model "FOC_Velocity_Encoder". It contains a table of resource usage and a list of report sections.

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

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

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)

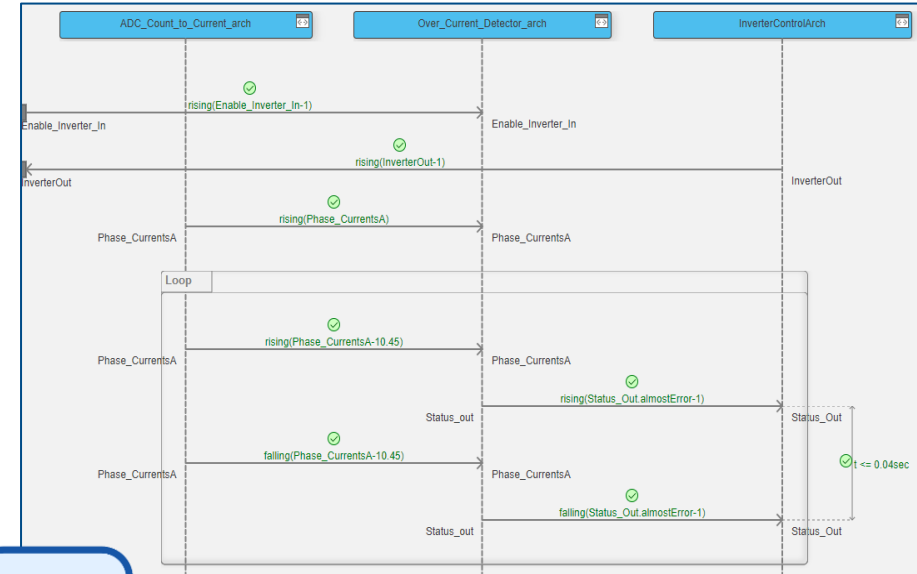
input



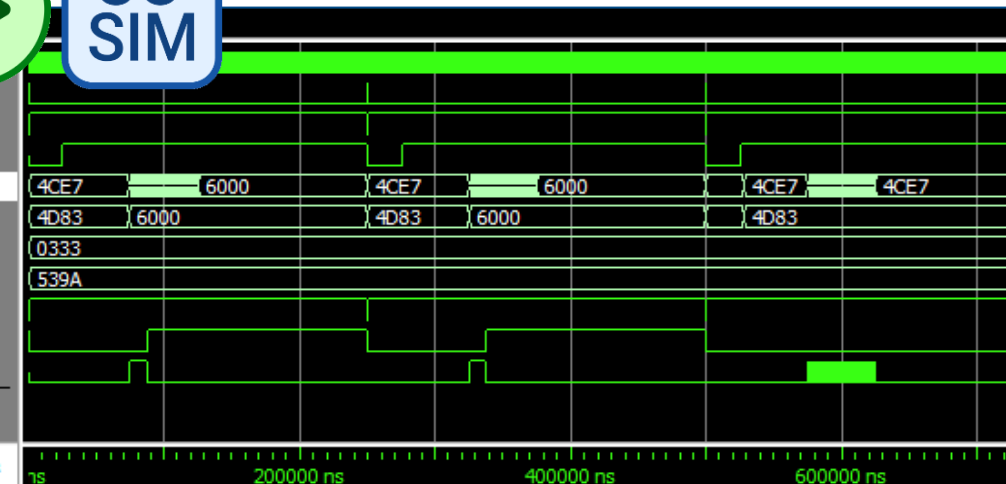
assess



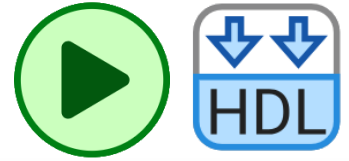
CO SIM



ck	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



Integration with other IPs or Software Components



Choose target platform and reference design

Direct integration with FPGA architecture and design

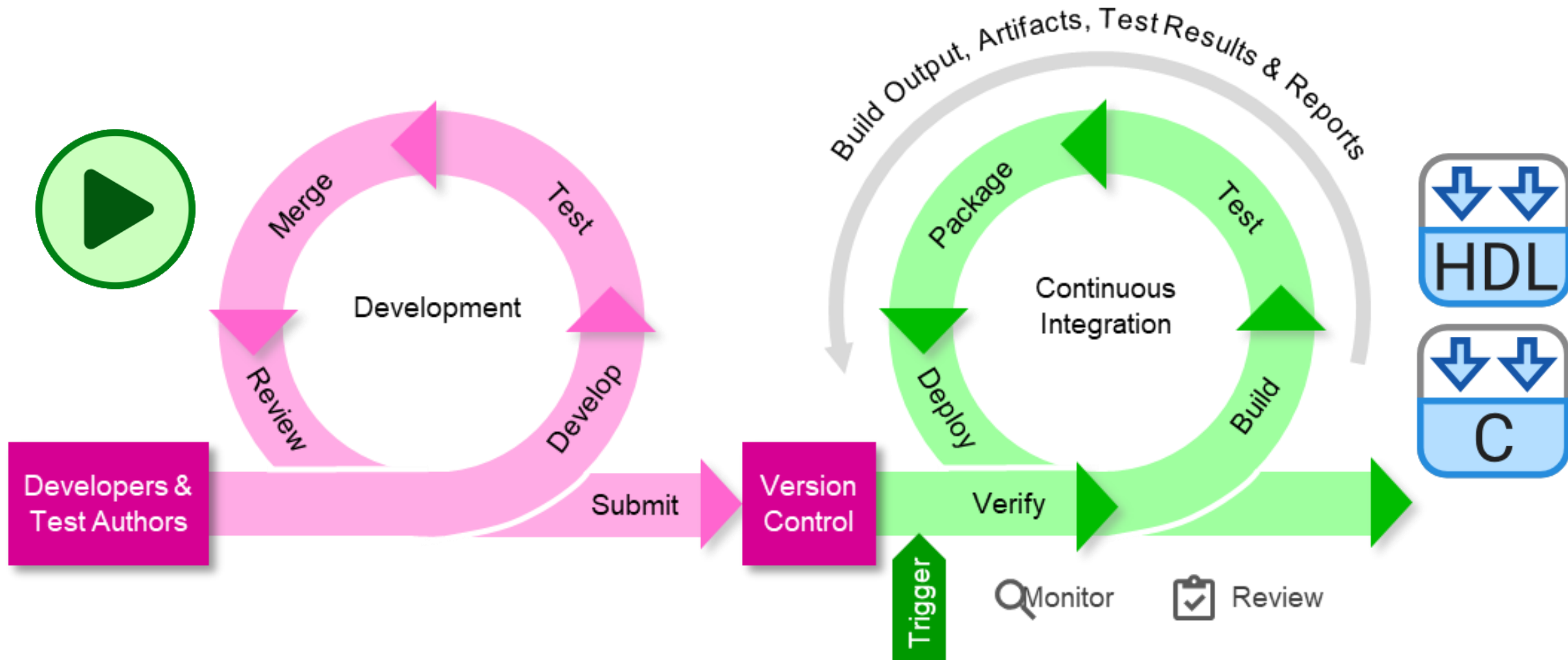
Map interfaces to AXI or external interfaces

Configuration Parameters: IntegrationFPGAmodel/Configuration (Active)

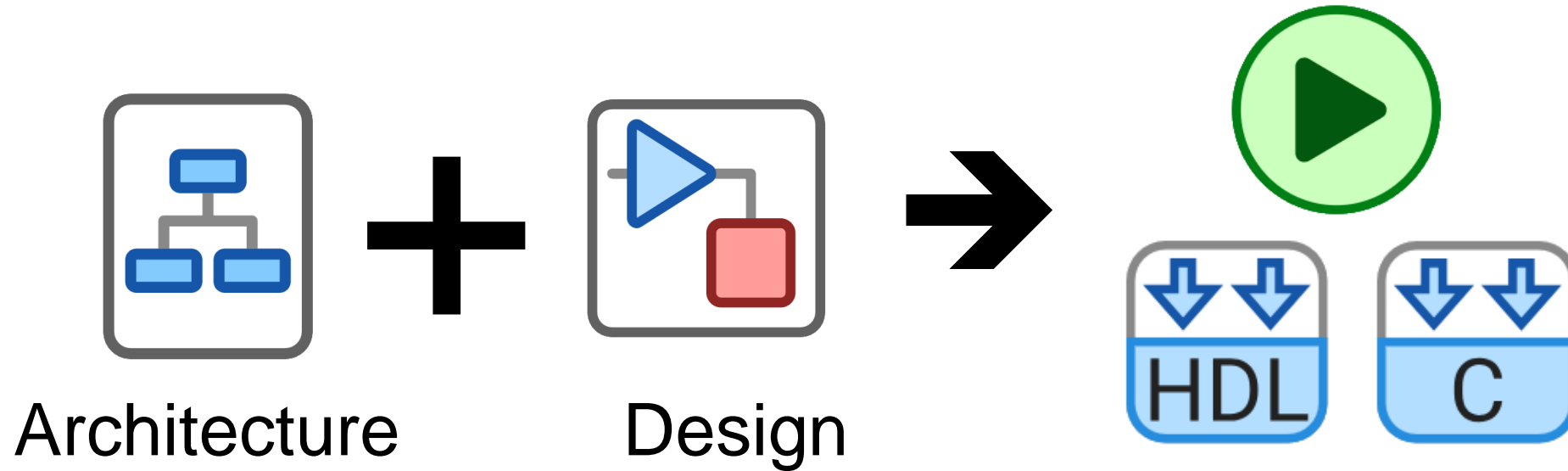
- Target Platform: **Trenz TE0820 with CR00140**
- Synthesis Tool: Generic Platform
- Family: Intel Arria 10 SoC development kit
- Package: Microchip PolarFire SoC Icicle Kit
- Reference Design: **Trenz TE0820 with CR00140**

Source	Port Type	Data Type	Interface
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
AXI_Current_Command	Inport	sfix16_En11	AXI4-Lite

Continuous Integration for Model-Based Design



Concluding remark



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, Aduvo