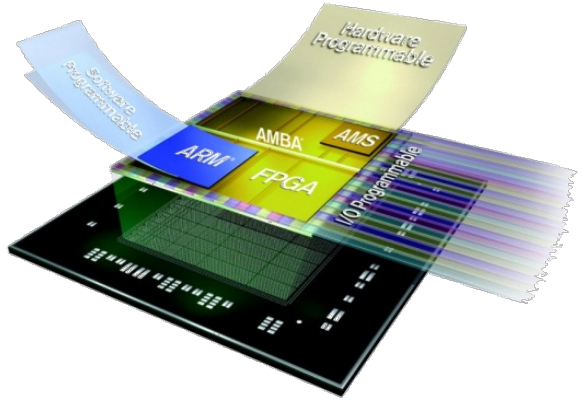
MATLAB EXPO 2016
KOREA

4월 28일 (목)

등록 하기 matlabexpo.co.kr

MATLAB과 Simulink를 이용한 프로그래머블 SoC 설계

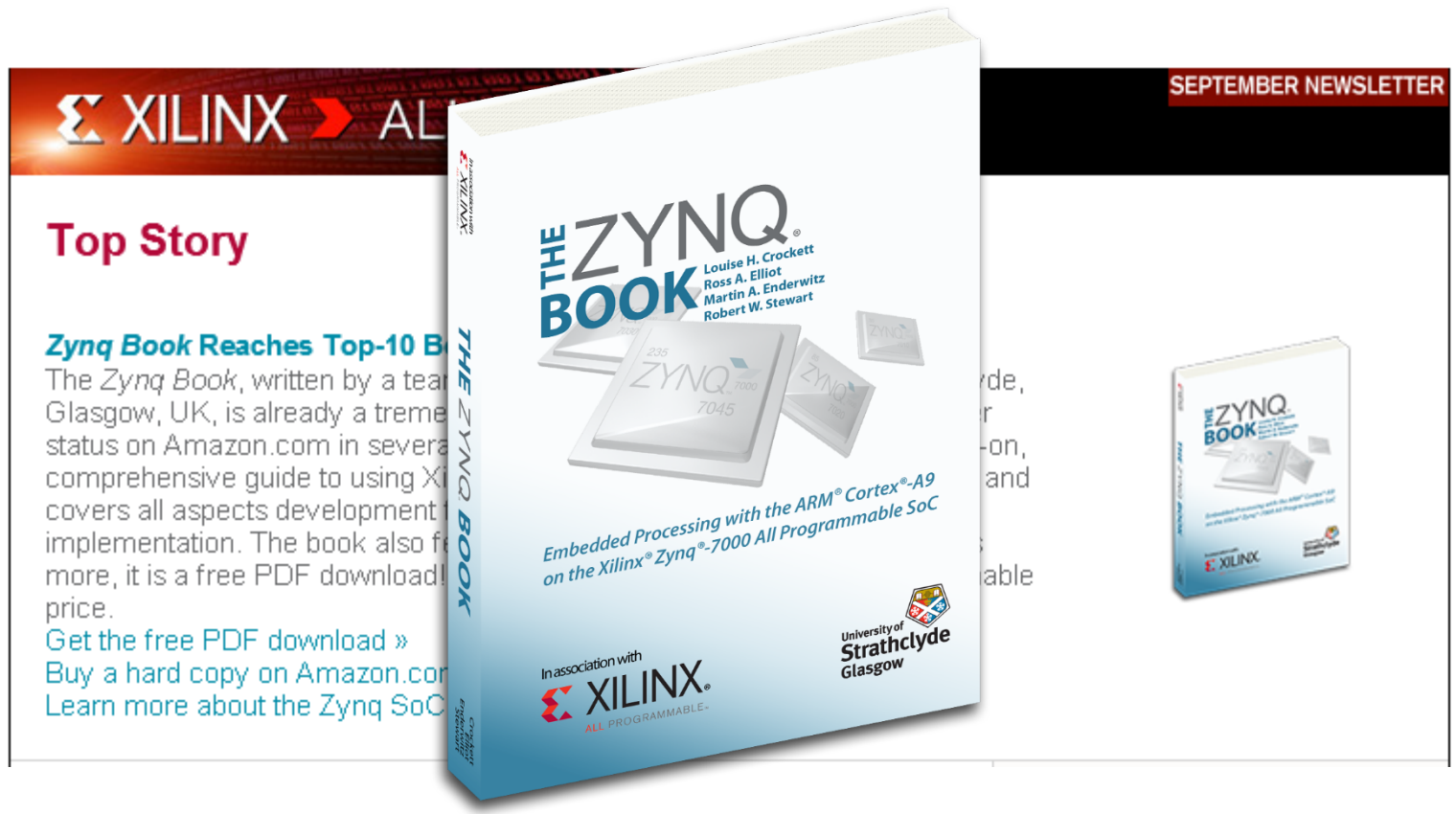


이웅재 부장
Application Engineering Group

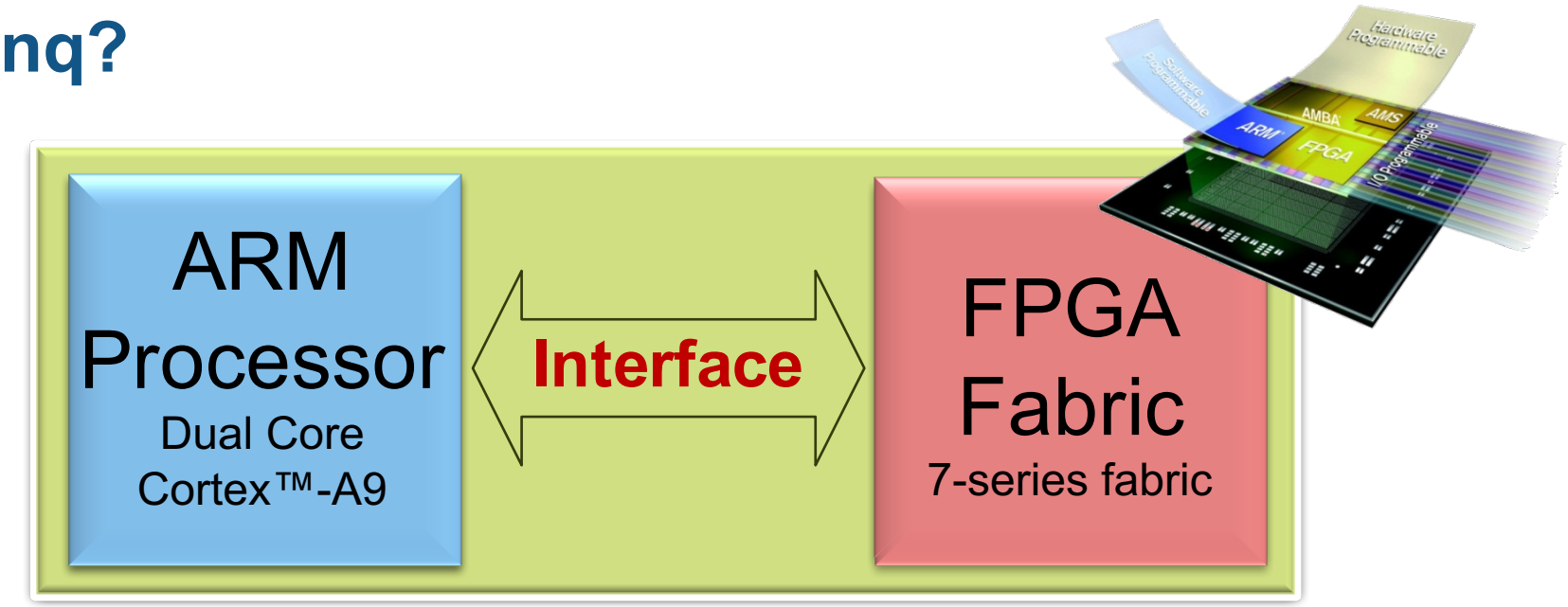
© 2016 The MathWorks, Inc.

Agenda

- Introduction
  - What is Zynq?
  - Design Challenges
- ZYNQ Design Process
  - MBD on Programmable SoC
  - Code Generation
  - Workflow
- Verification & Partitioning
  - Parameter Tuning
  - UDP Interface
  - Processor In the Loop
- Advanced Features
- Conclusions

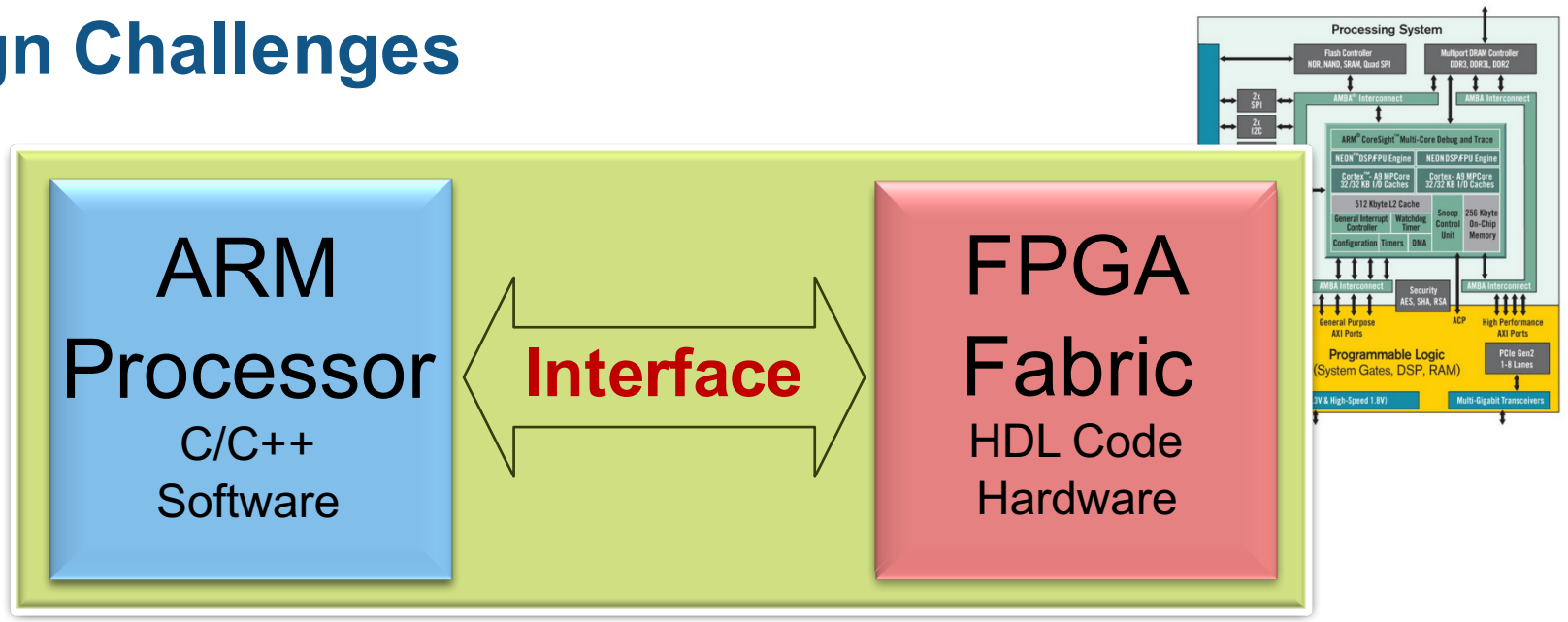

XILINX
SEPTEMBER NEWSLETTER
Top Story
Zynq Book Reaches Top-10 B
The Zynq Book, written by a tea
Glasgow, UK, is already a treme
status on Amazon.com in severa
comprehensive guide to using Xi
covers all aspects development
implementation. The book also f
more, it is a free PDF download!
price.
Get the free PDF download »
Buy a hard copy on Amazon.co
Learn more about the Zynq SoC
THE ZYNQ BOOK
Louise H. Crockett
Ross A. Elliot
Martin A. Enderwitz
Robert W. Stewart
Embedded Processing with the ARM® Cortex®-A9 on the Xilinx® Zynq®-7000 All Programmable SoC
In association with
XILINX
ALL PROGRAMMABLE
University of Strathclyde Glasgow

What is Zynq?



- New product family from Xilinx®
  - All Programmable System on Chip (SoC)
- FPGA Fabric + ARM® on one a single chip
  - Enables high-performance system development
  - Reduces BOM cost over multi-chip solutions

Zynq Design Challenges



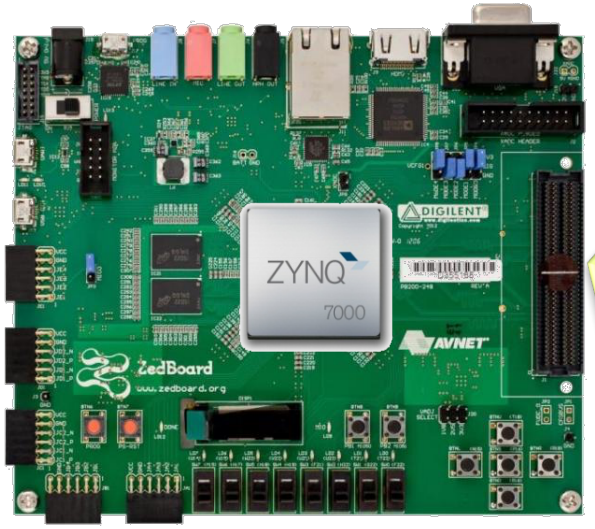
- **FPGA Designers** not familiar with programming processors
- **DSP/Processor programmers** not familiar with FPGAs
- **What should run on the FPGA vs. what should run on the ARM**?
- **No established rules for hooking up the interface** between FPGA and ARM processor

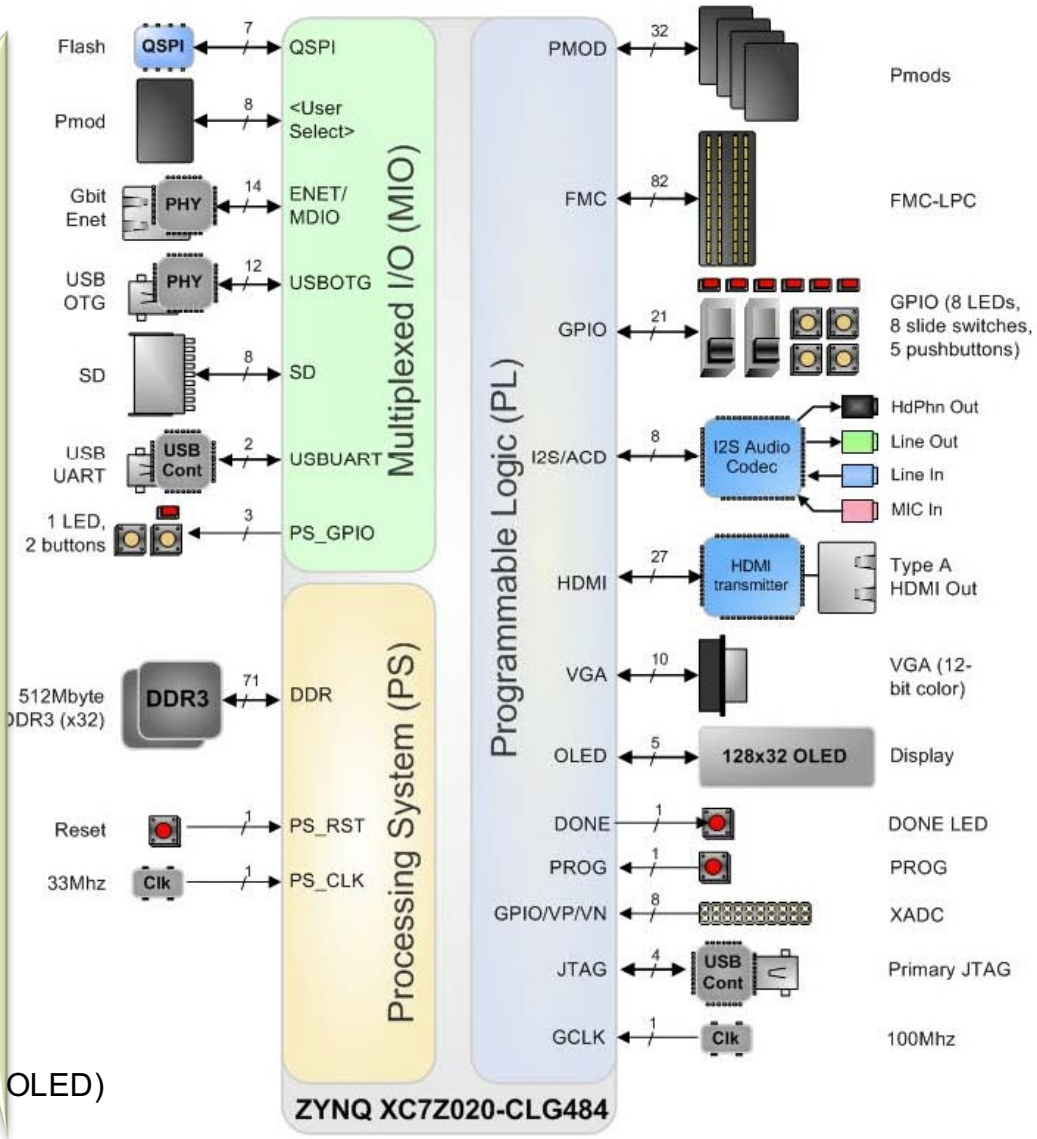
How can I address these challenges and get my project onto Zynq quickly?

- *Model-Based Design* provides a single environment from requirements to prototype *seamlessly*

- A ***guided workflow*** for hardware and software development

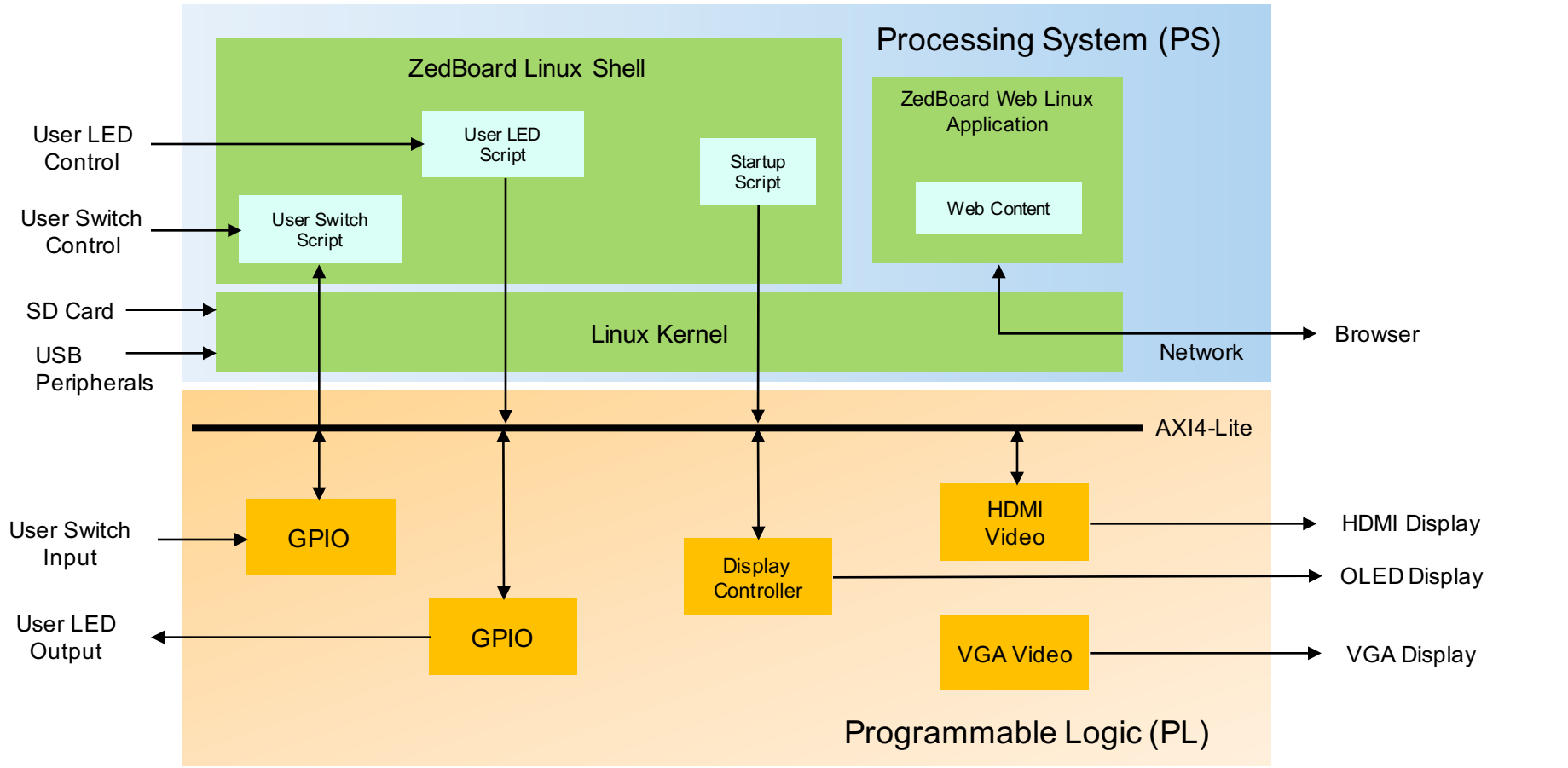
Hardware platform to prototyping



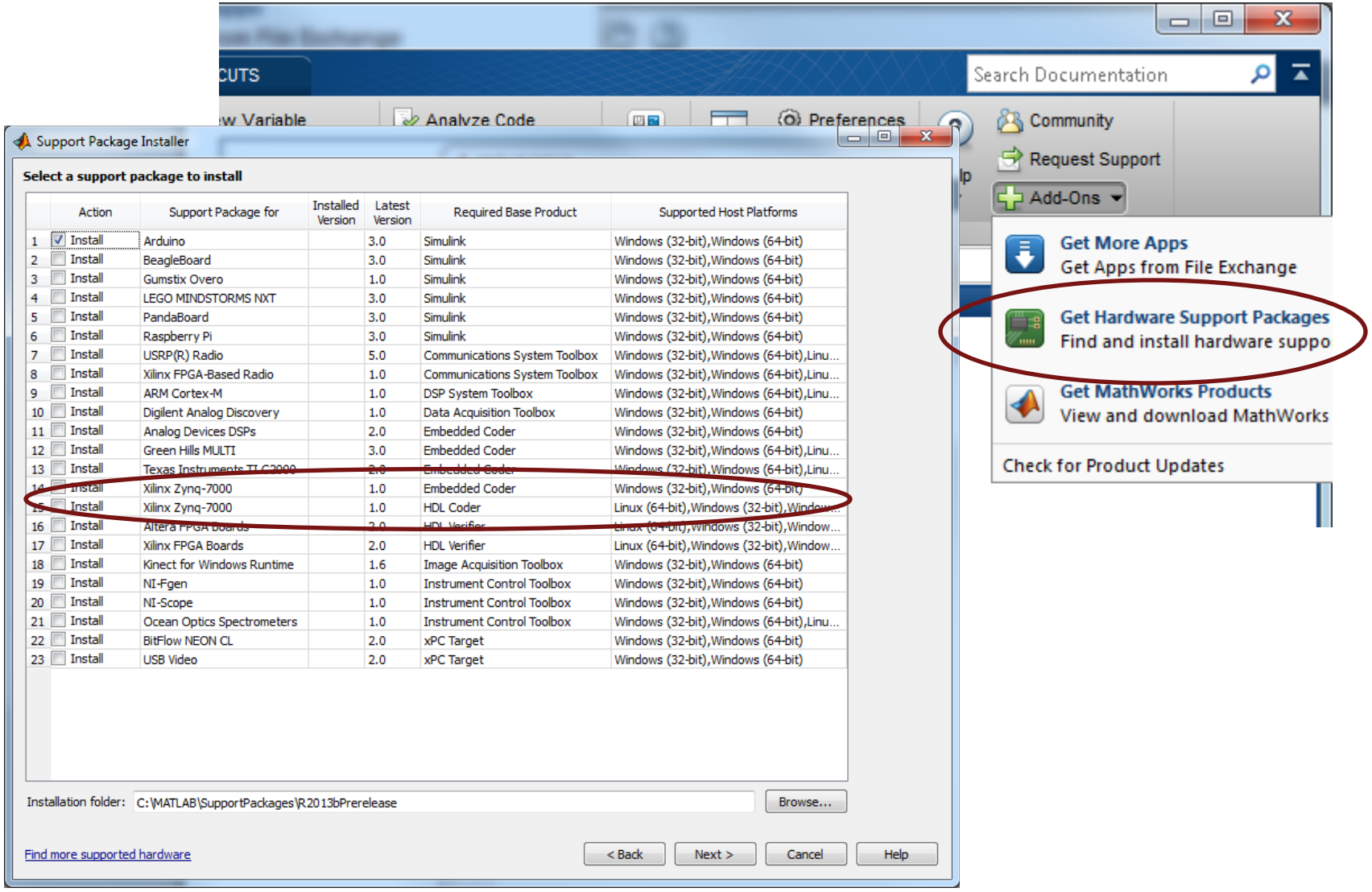


- Board Name: **ZedBoard™**
  Zynq-7000 AP SoC XC7Z020-CLG484
- Memory:
  - ✓ 512 MB DDR3
  - ✓ 256 Mb Quad-SPI Flash
  - ✓ 4 GB SD card
- Onboard USB-JTAG Programming
- 10/100/1000 Ethernet
- USB OTG 2.0 and USB-UART
- PS & PL I/O expansion (FMC, Pmod™, XADC)
- Multiple displays (1080p HDMI, 8-bit VGA, 128 x 32 OLED)
- I^2S Audio CODEC

System configuration for prototyping



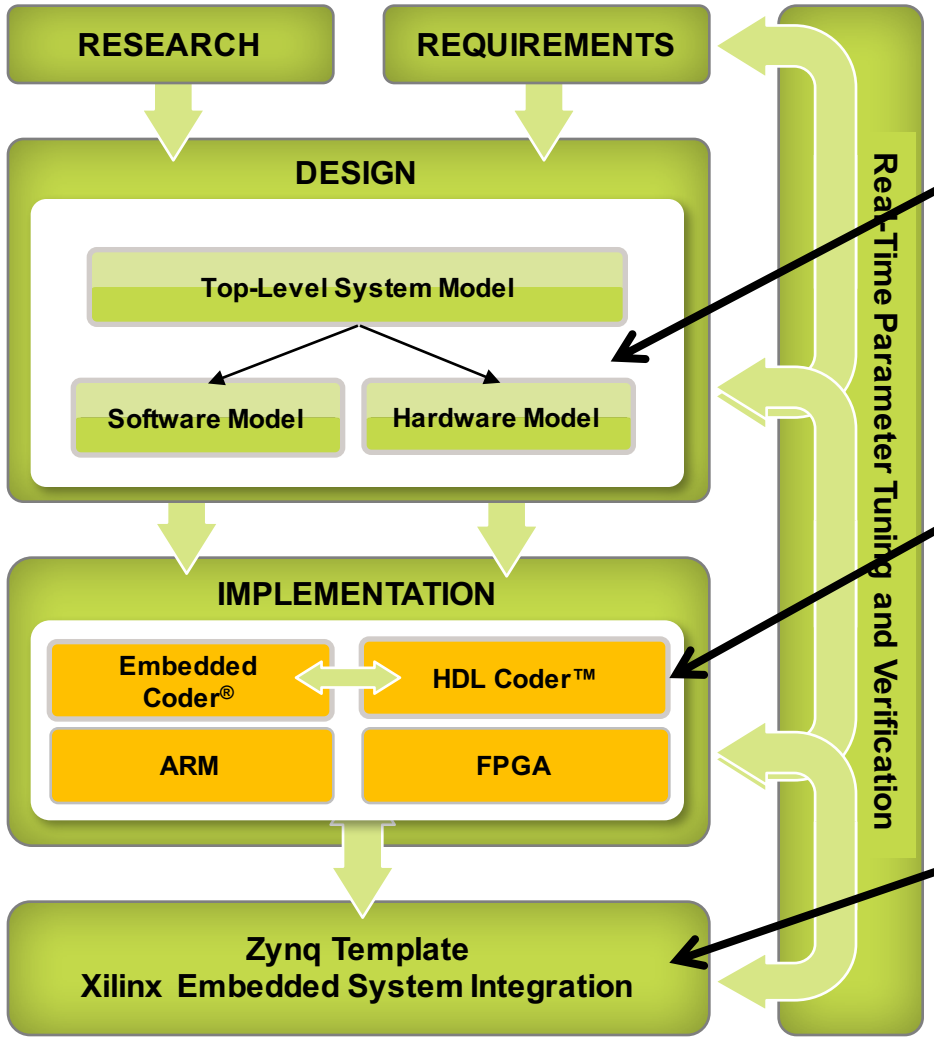
Support Package Installer



Agenda

- Introduction
  - What is Zynq?
  - Design Challenges
- ZYNQ Design Process
  - MBD on Programmable SoC
  - Code Generation
  - Workflow
- Verification & Partitioning
  - Parameter Tuning
  - UDP Interface
  - Processor In the Loop
- Advanced Features
- Conclusions

Model-Based Design for Zynq

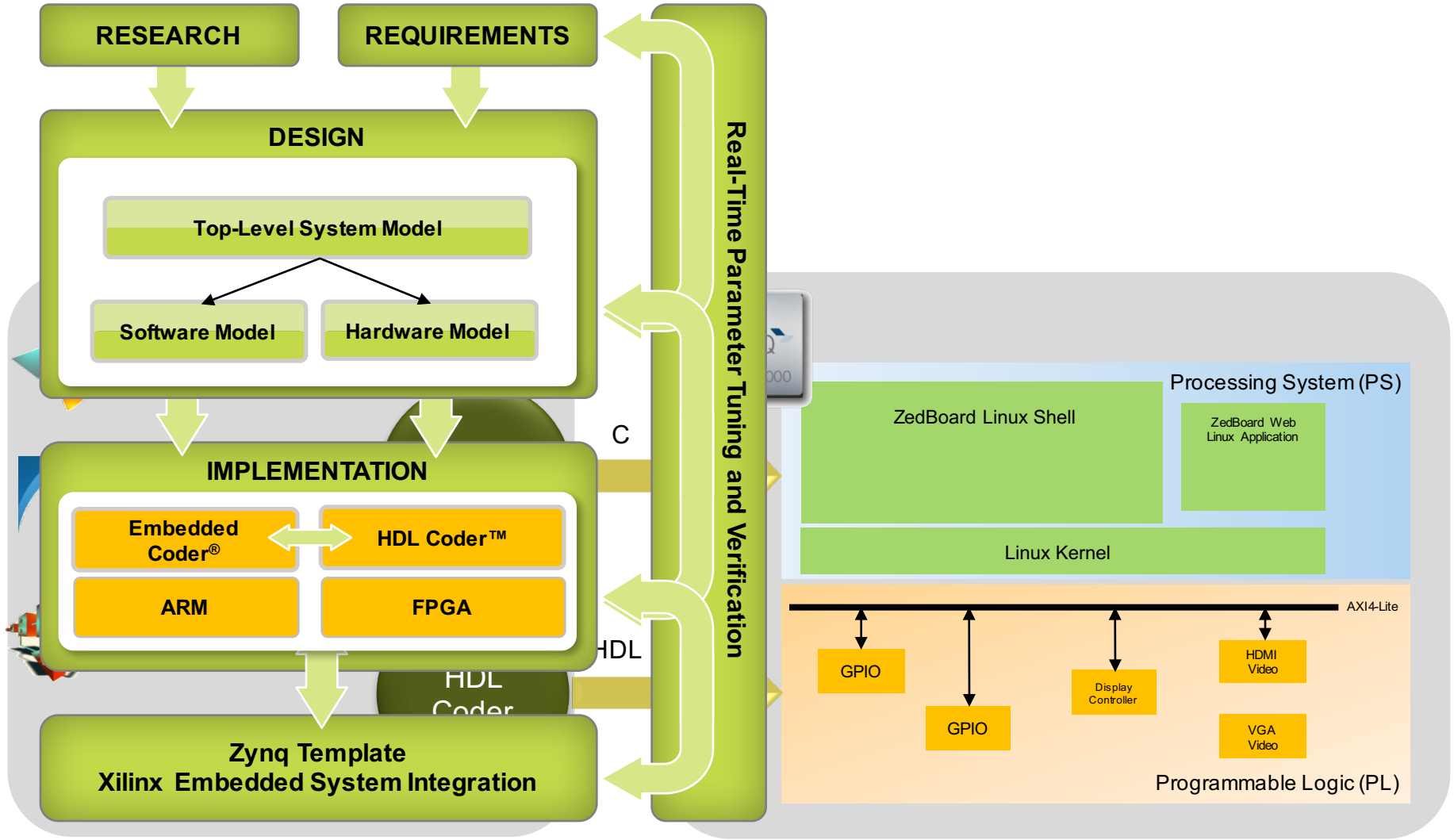


User defines partitioning

MathWorks automates code and **interface-model generation**

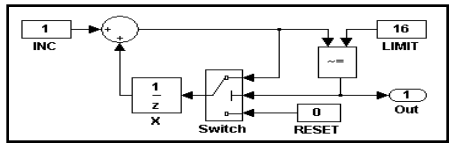
MathWorks automates the **build and download** through the Xilinx tools

Model-Based Design for Zynq

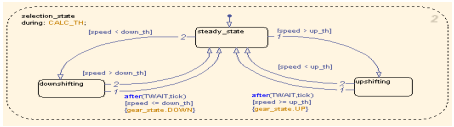


Code Generation Scheme

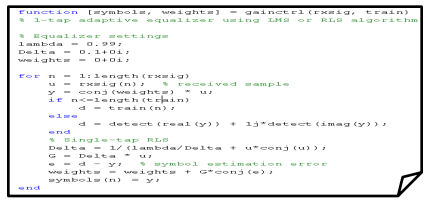
Multi-Domain, Multi-Target Technology



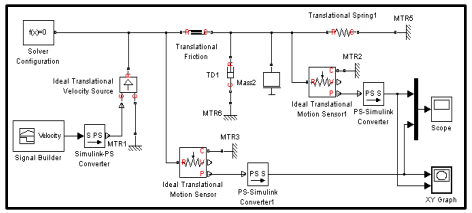
Simulink



Stateflow



MATLAB Function



Simscape

Multiple analyses and optimizations

IR

Analysis/ Transform

Analysis/ Transform

Intermediate Representation (IR)

Target Language Backend

C

C++

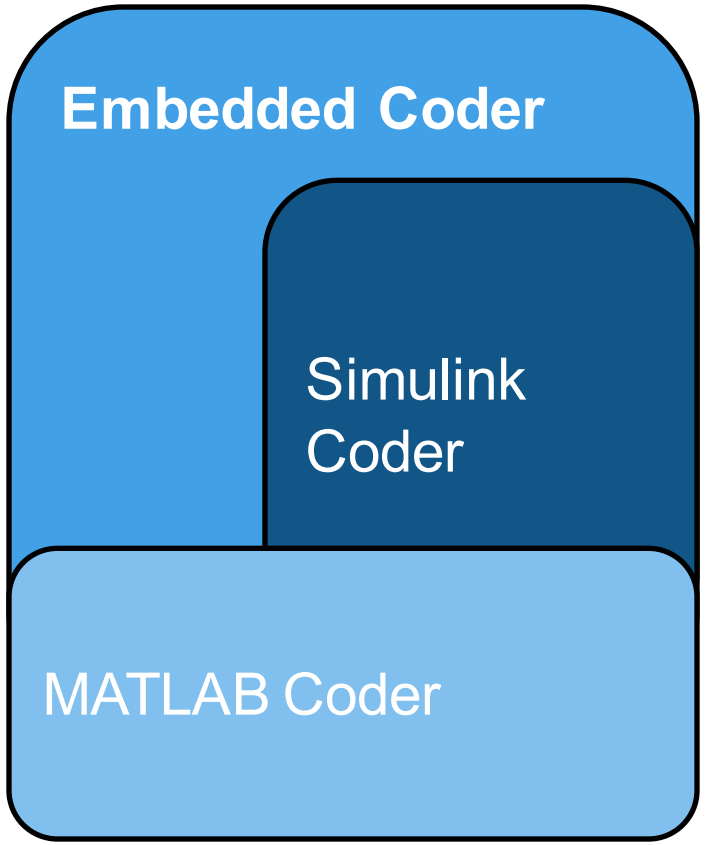
HDL

PLC

Multiple Domains

Multiple Targets

Code Generation Products for C/C++



Embedded Coder™
Automatically generate C and C++ optimized for embedded systems

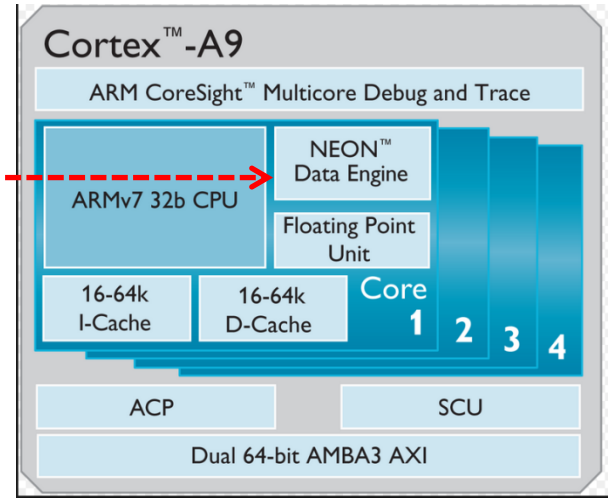
Simulink® Coder™
Automatically generate C and C++ from **Simulink** models and **Stateflow** charts

MATLAB® Coder™
Automatically generate C and C++ from **MATLAB** code

Optimize the C/C++ code for performance

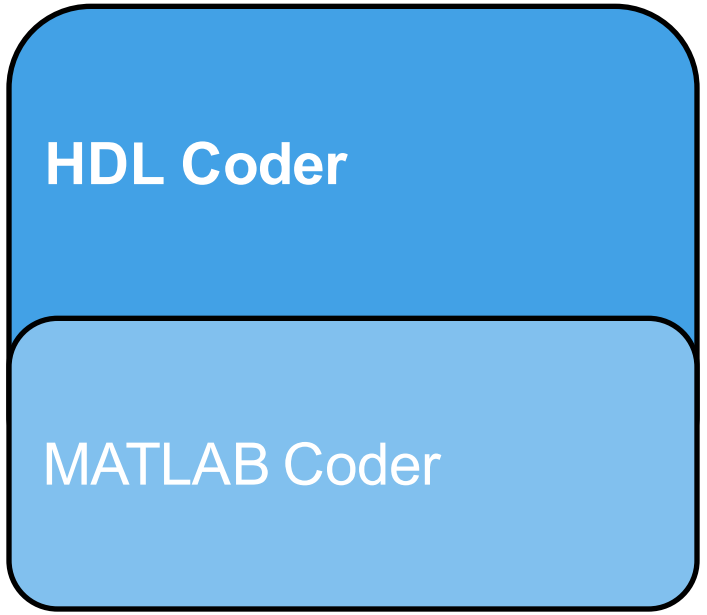
```
neon_a0 = vld1q_f32(A + j_T);
j_T+=Row;
neon_a1 = vld1q_f32(A + j_T);
j_T+=Row;
neon_a2 = vld1q_f32(A + j_T);
j_T+=Row;
neon_a3 = vld1q_f32(A + j_T);
```

- **SIMD intrinsics**
- **Fixed-point intrinsics**
- **Assembly**
- **Optimized libraries**



1. Optimize the generated C/C++ code
2. Use the ARM NEON Media Processing Engine

Code Generation Products for VHDL/Verilog

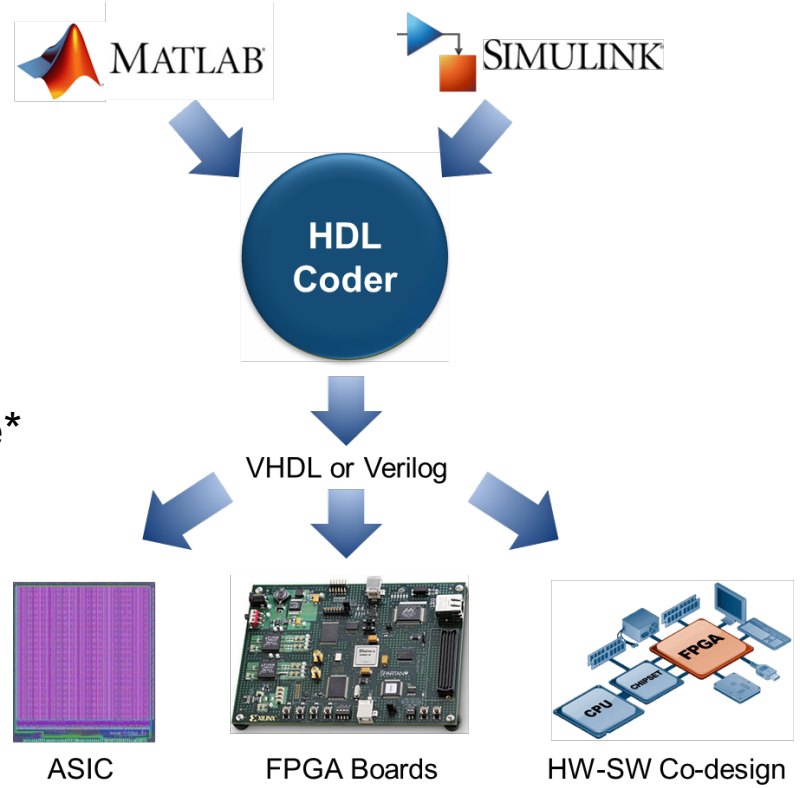


HDL Coder™
Automatically generate synthesizable RTL code (VHDL or Verilog) from **MATLAB** code and **Simulink** Model

MATLAB® Coder™
Automatically generate C and C++ from **MATLAB** code

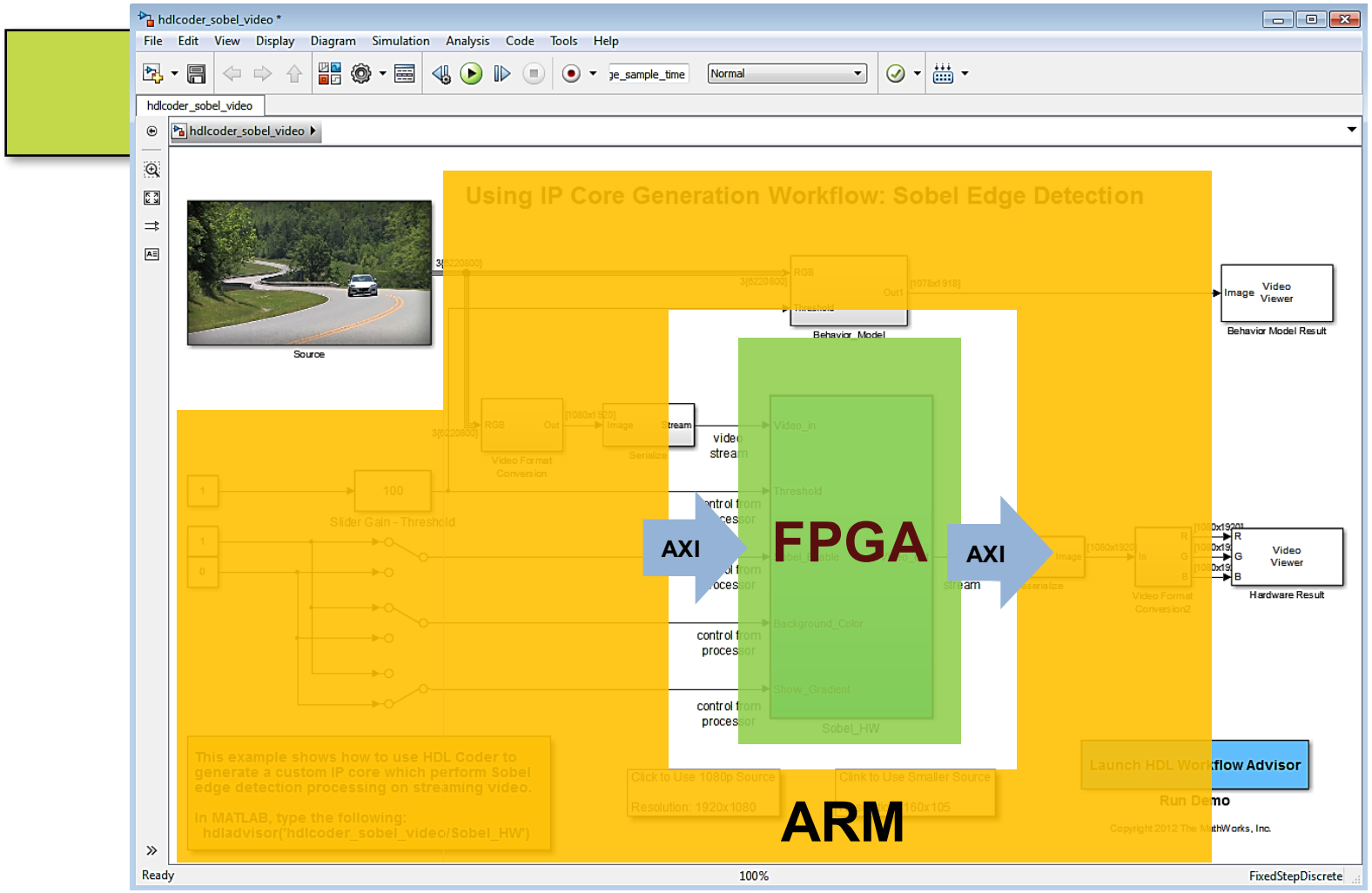
HDL Coder Key Features

- **Code Generation**
  - Target-independent Synthesizable RTL Code
  - IEEE 1376 compliant VHDL®
  - IEEE 1364-2001 compliant Verilog®
- **Verification**
  - Generate HDL test-bench
  - Co-simulate with ModelSim and Incisive*
- **Design automation**
  - Synthesize using integrated Xilinx and Altera synthesis tool interface
  - Optimize for area-speed
  - Program Xilinx and Altera boards

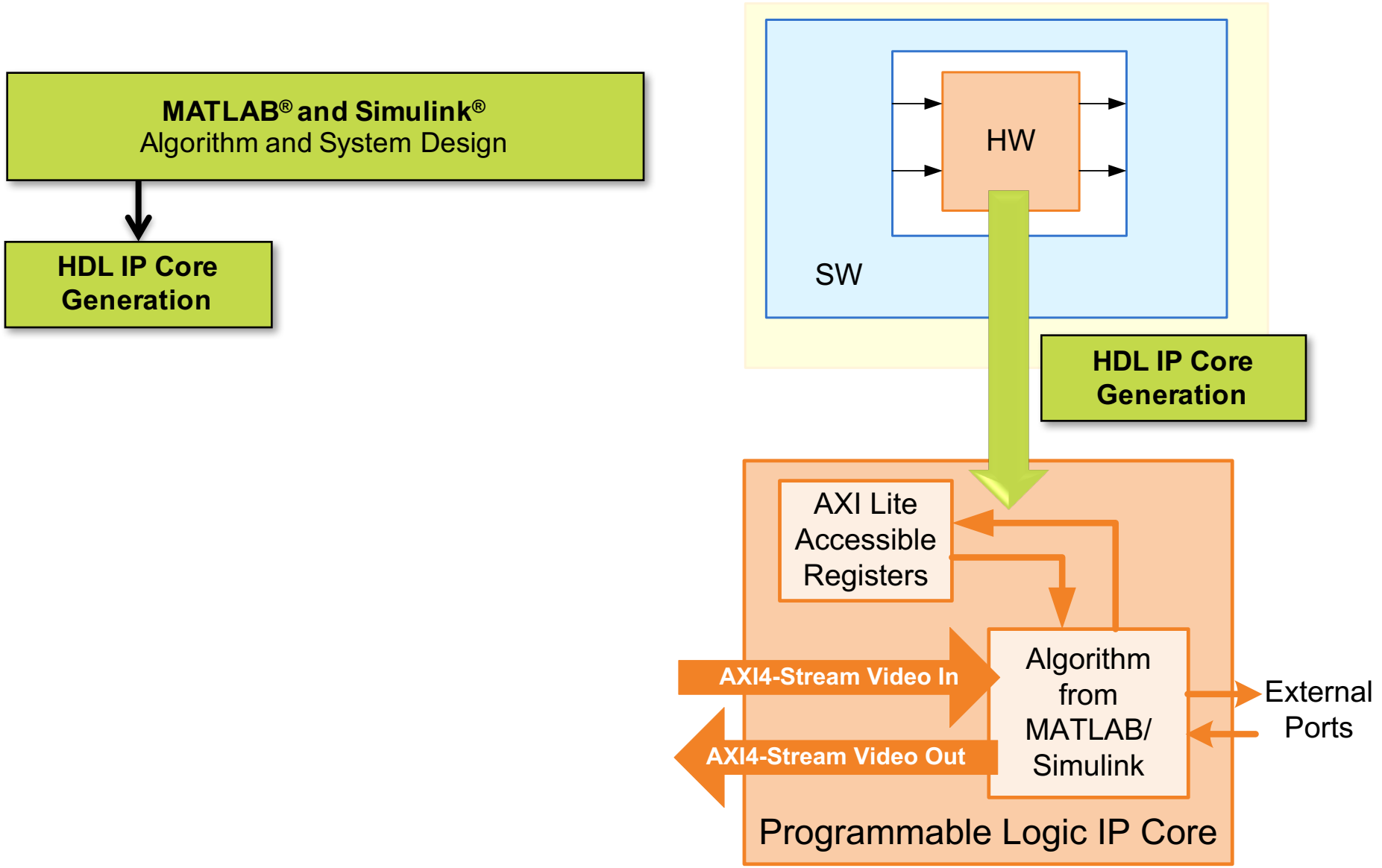


* HDL Verifier required for co-simulation and FPGA-in-the-loop verification

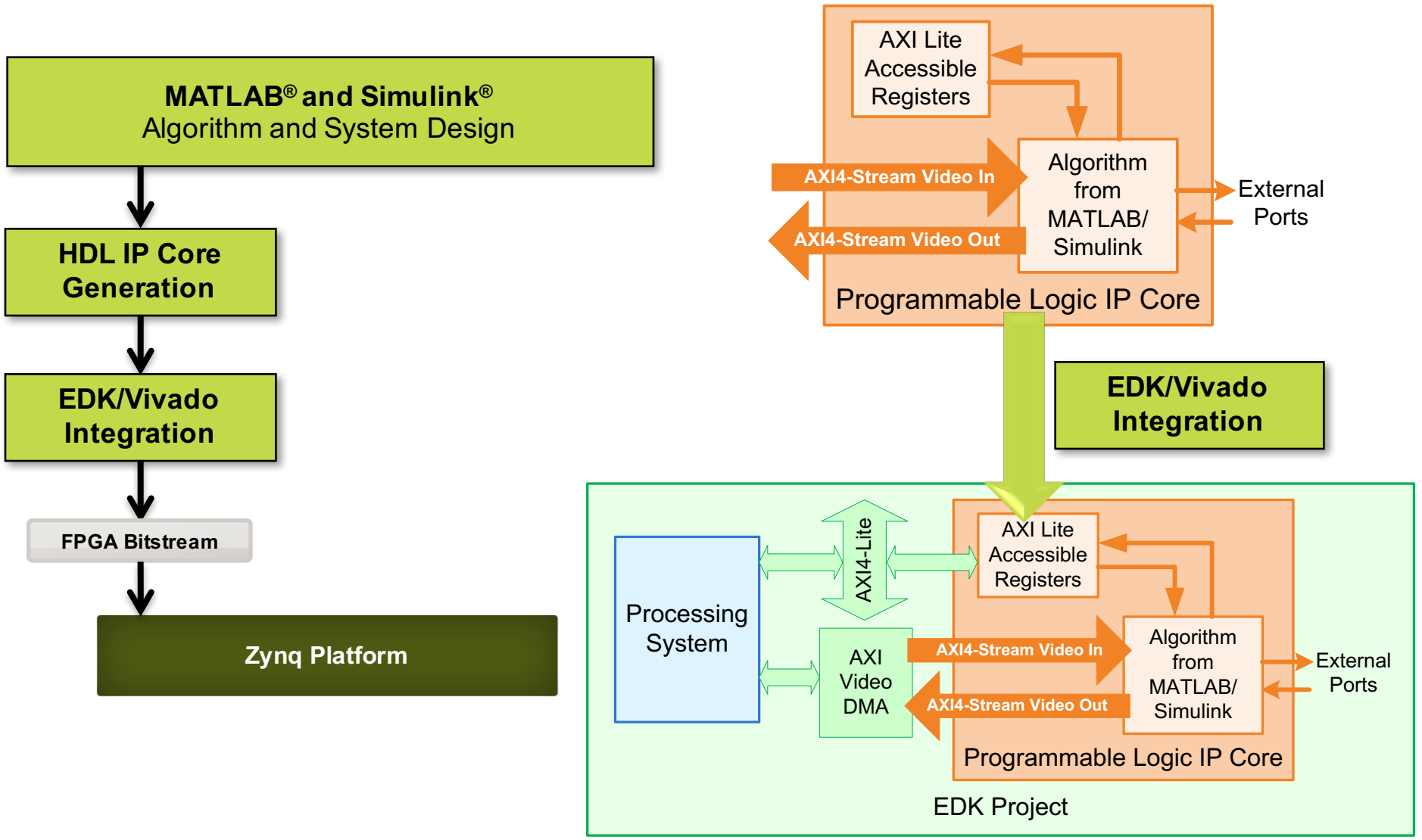
Zynq Model-Based Design Workflow



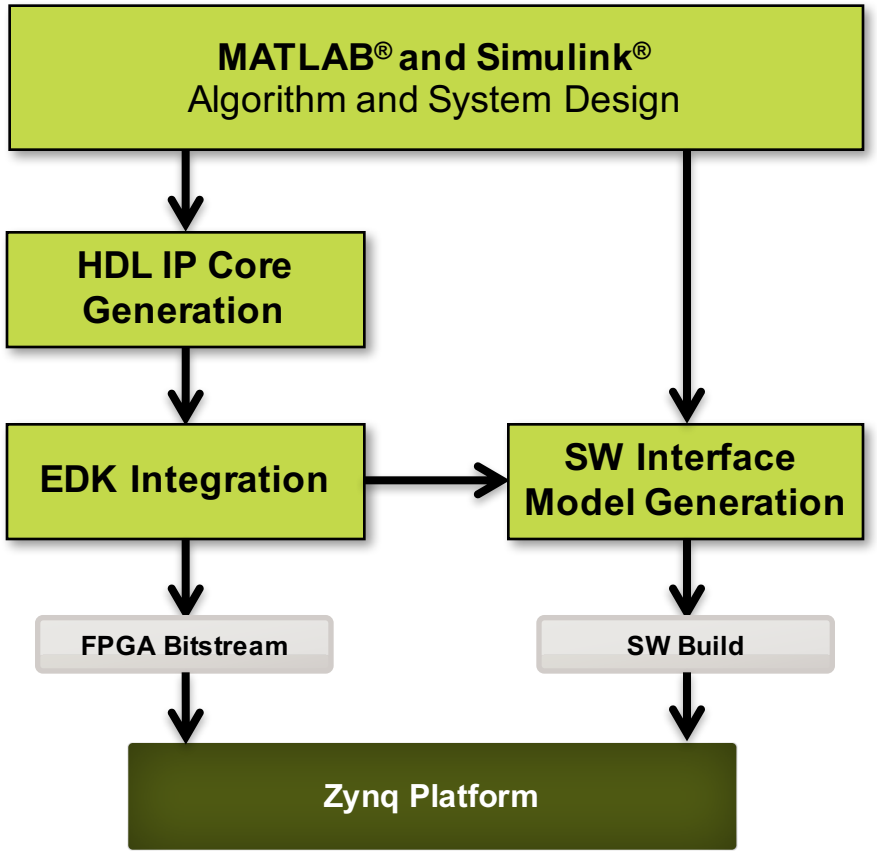
Zynq Model-Based Design Workflow

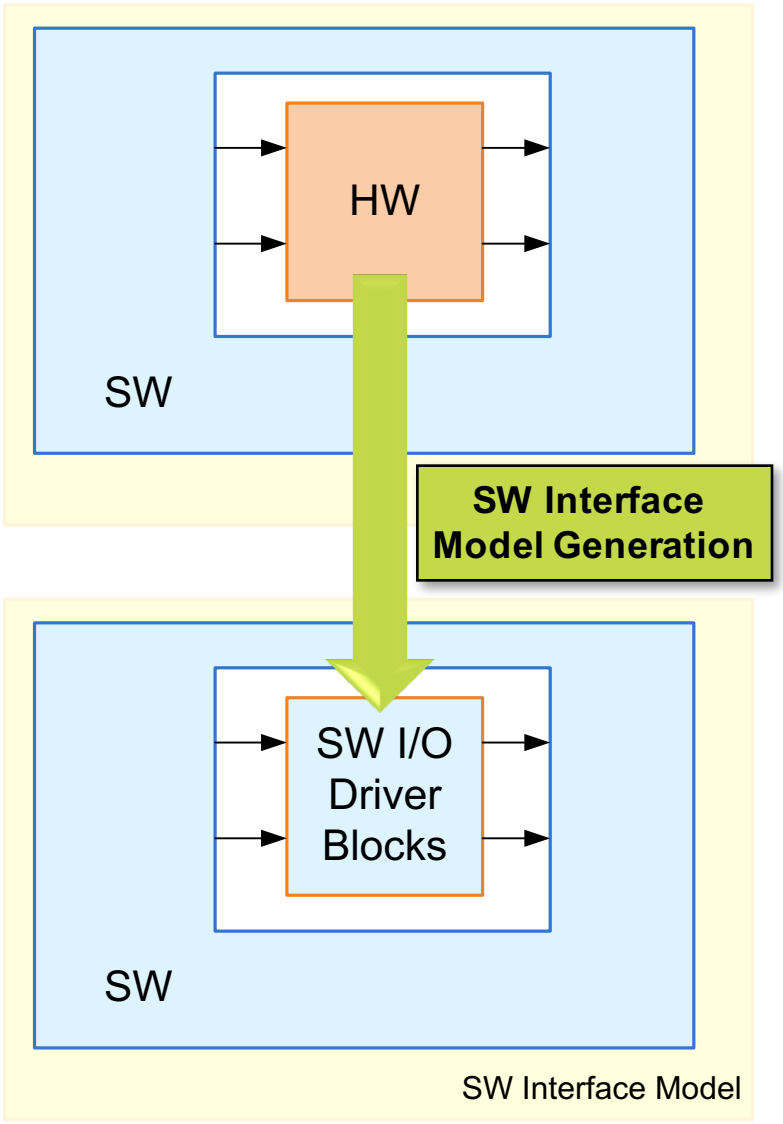


Zynq Model-Based Design Workflow

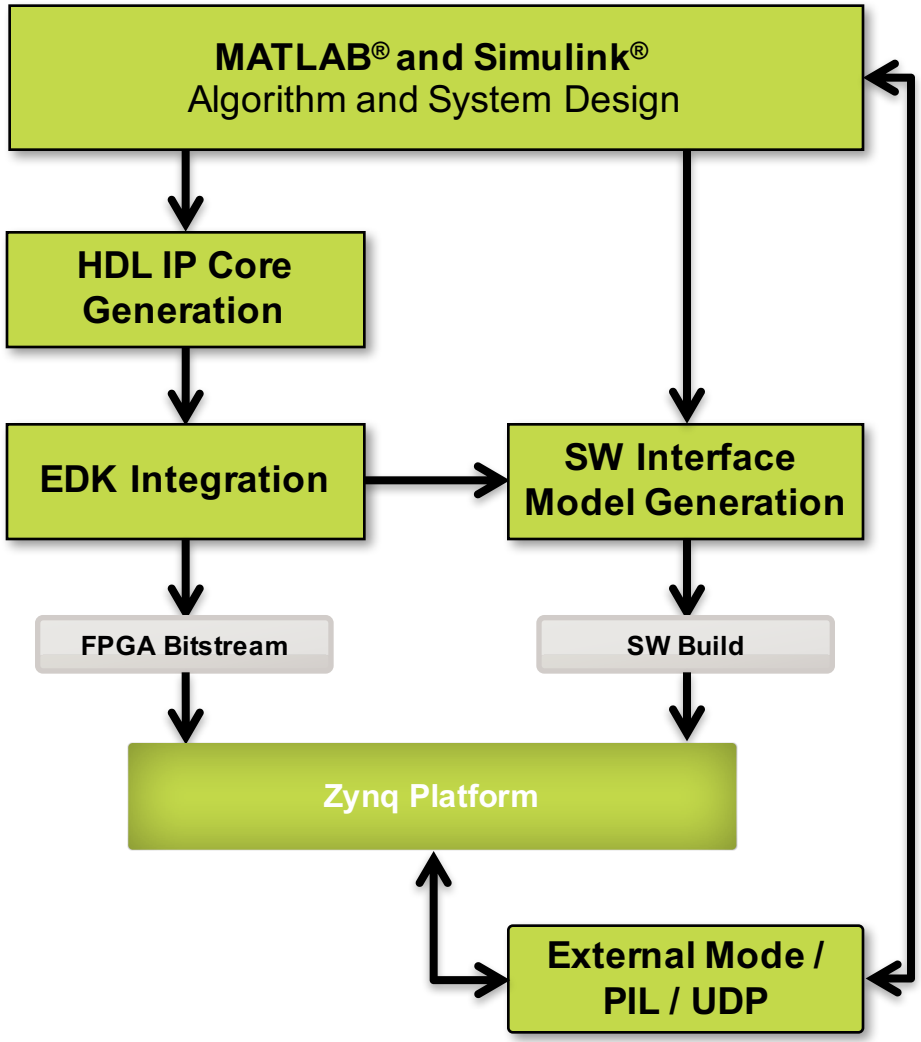


Zynq Model-Based Design Workflow



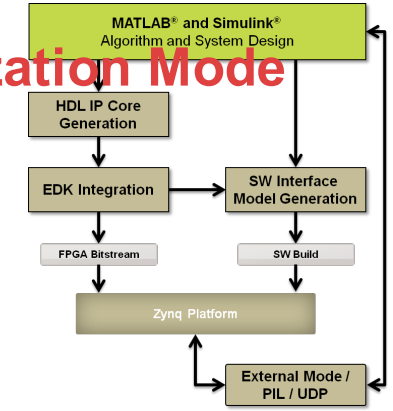


Zynq Model-Based Design Workflow

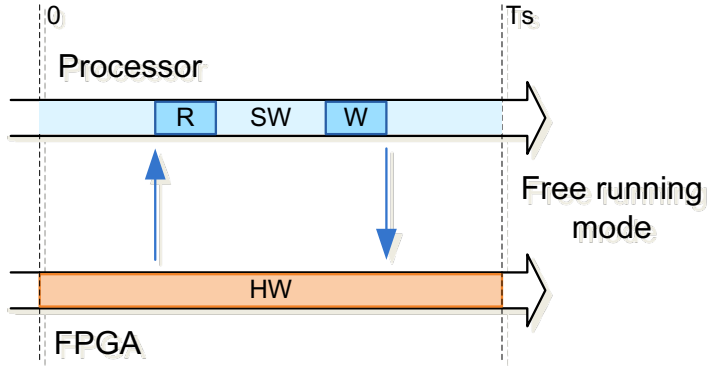


- Real-time Parameter Tuning and Verification
  - External Mode
  - Processor-in-the-loop
  - UDP
- More probe and debug capability in the future

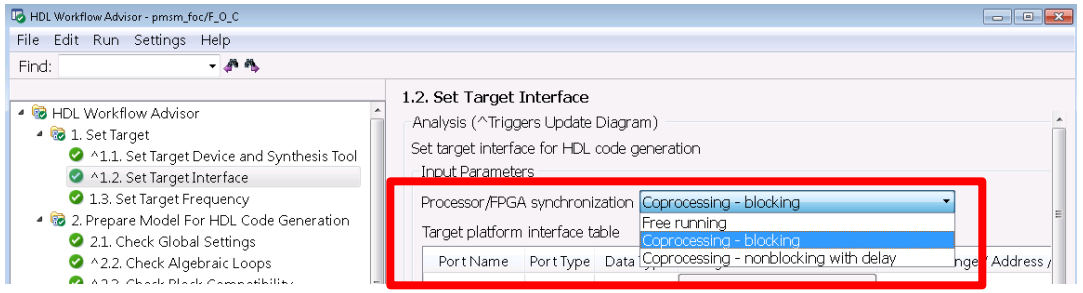
Communication between ARM & FPGA Synchronization Mode

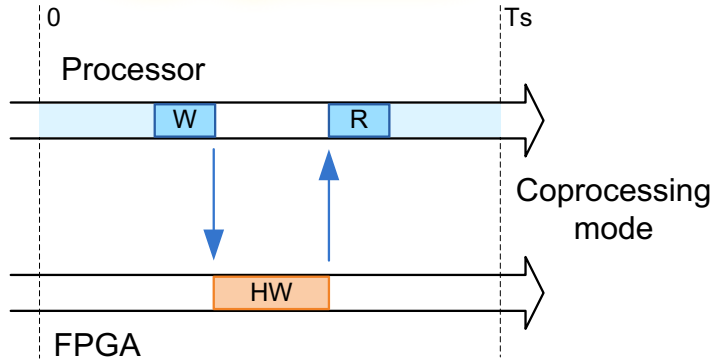


- Defines the synchronization behavior between ARM and FPGA
- Free running mode
  - No explicit synchronization between ARM and FPGA
- Co-processing mode
  - Explicit blocking synchronization between ARM and FPGA
- Choose in HDL Workflow Advisor
  - xPC/FPGA Turnkey Workflow
  - Zynq IP Core Generation



Processor/FPGA Synchronization





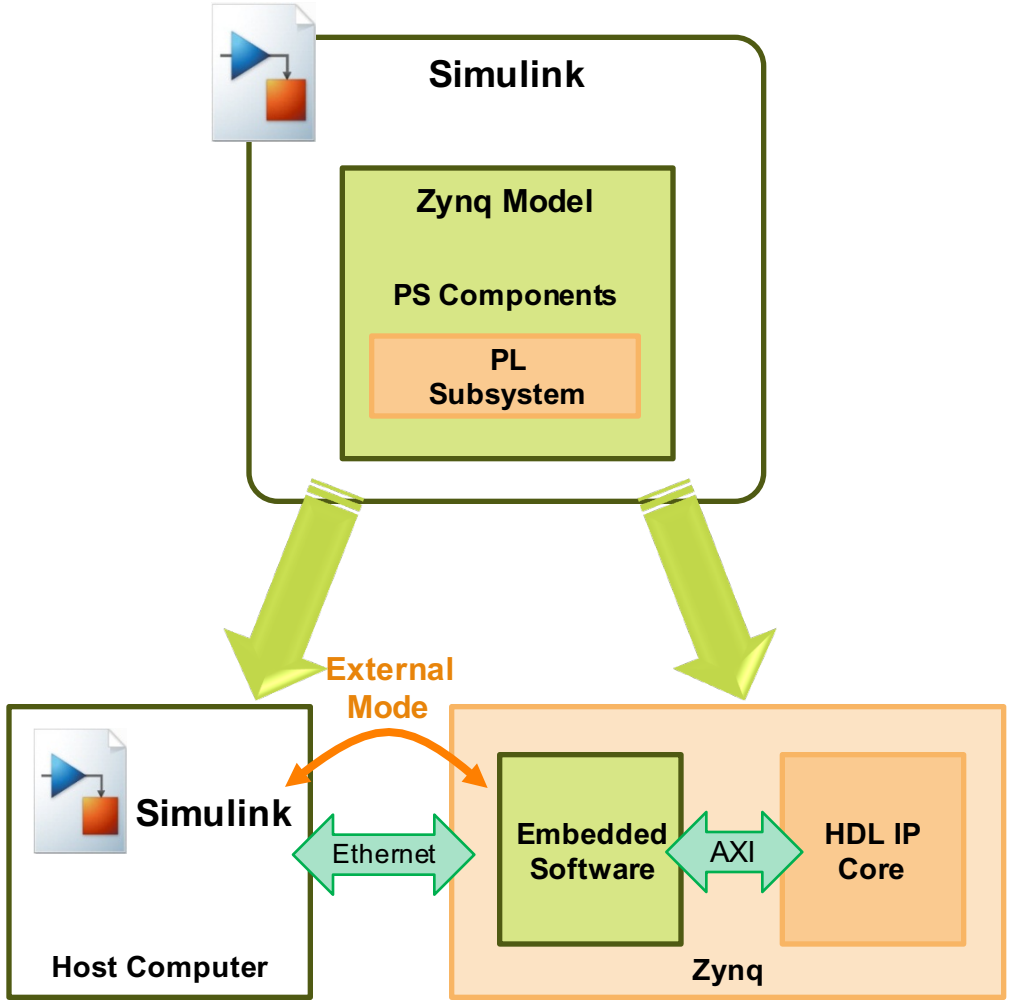
Agenda

- Introduction
  - What is Zynq?
  - Design Challenges
- ZYNQ Design Process
  - MBD on Programmable SoC
  - Code Generation
  - Workflow
- Verification & Partitioning
  - Parameter Tuning
  - UDP Interface
  - Processor In the Loop
- Additional Features
- Conclusions

Verification Challenges:

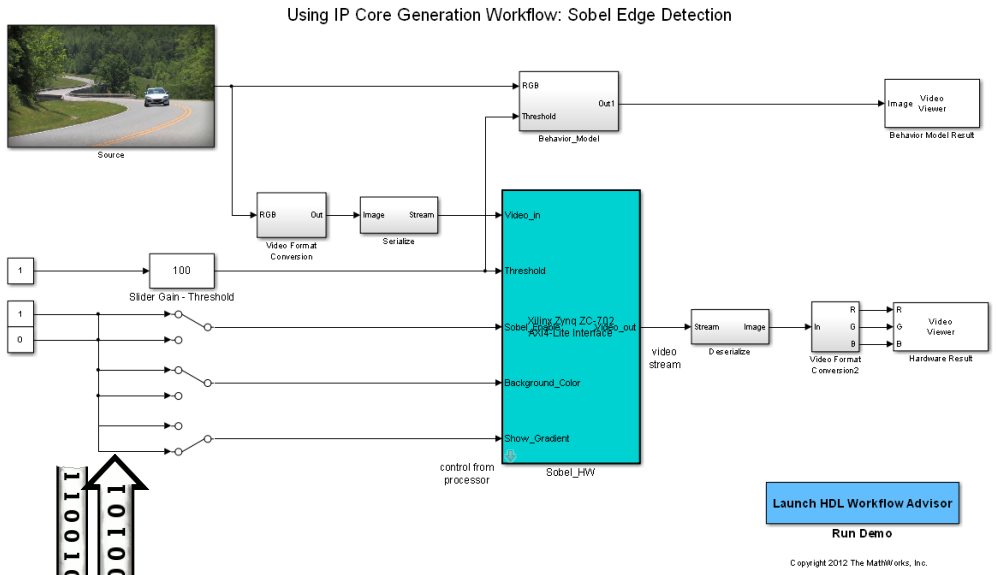
- Design the test bench twice
  - 10 – to – 1 ratio of Test bench LOC – to – Design LOC *(for HDL)*
- Many stimuli-files from MATLAB
- How to analyze results?

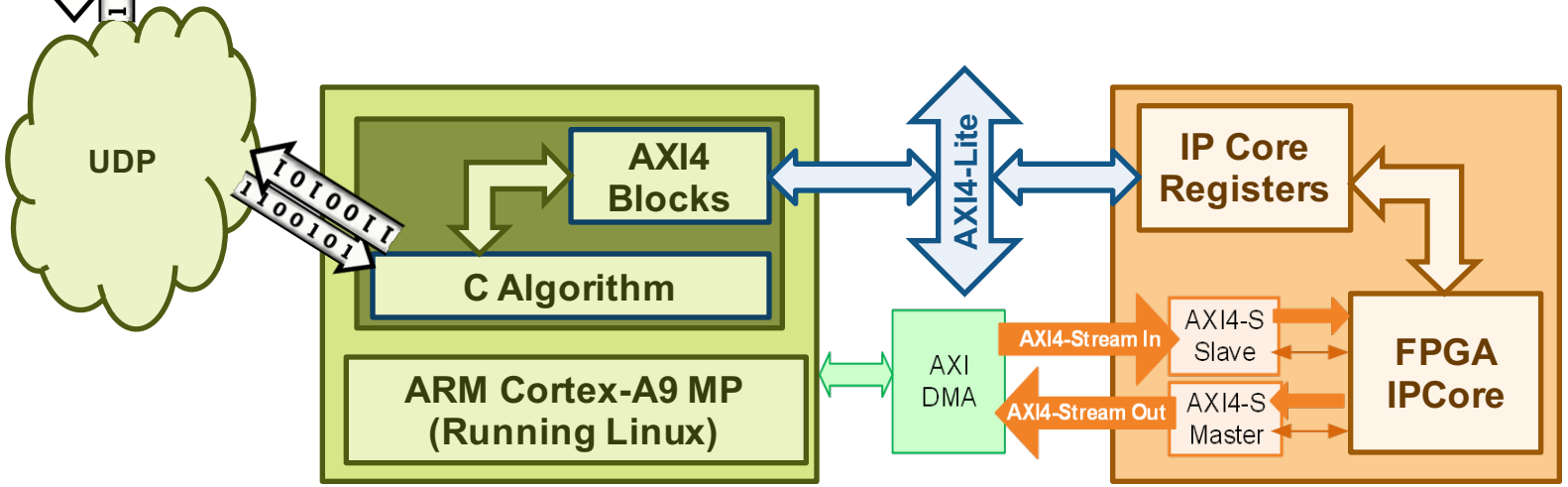
Parameter tuning through External Mode



- Tuning model parameters and evaluate the effects of different parameter values on model results in real time.
- Helping you find the optimal values to achieve desired performance.

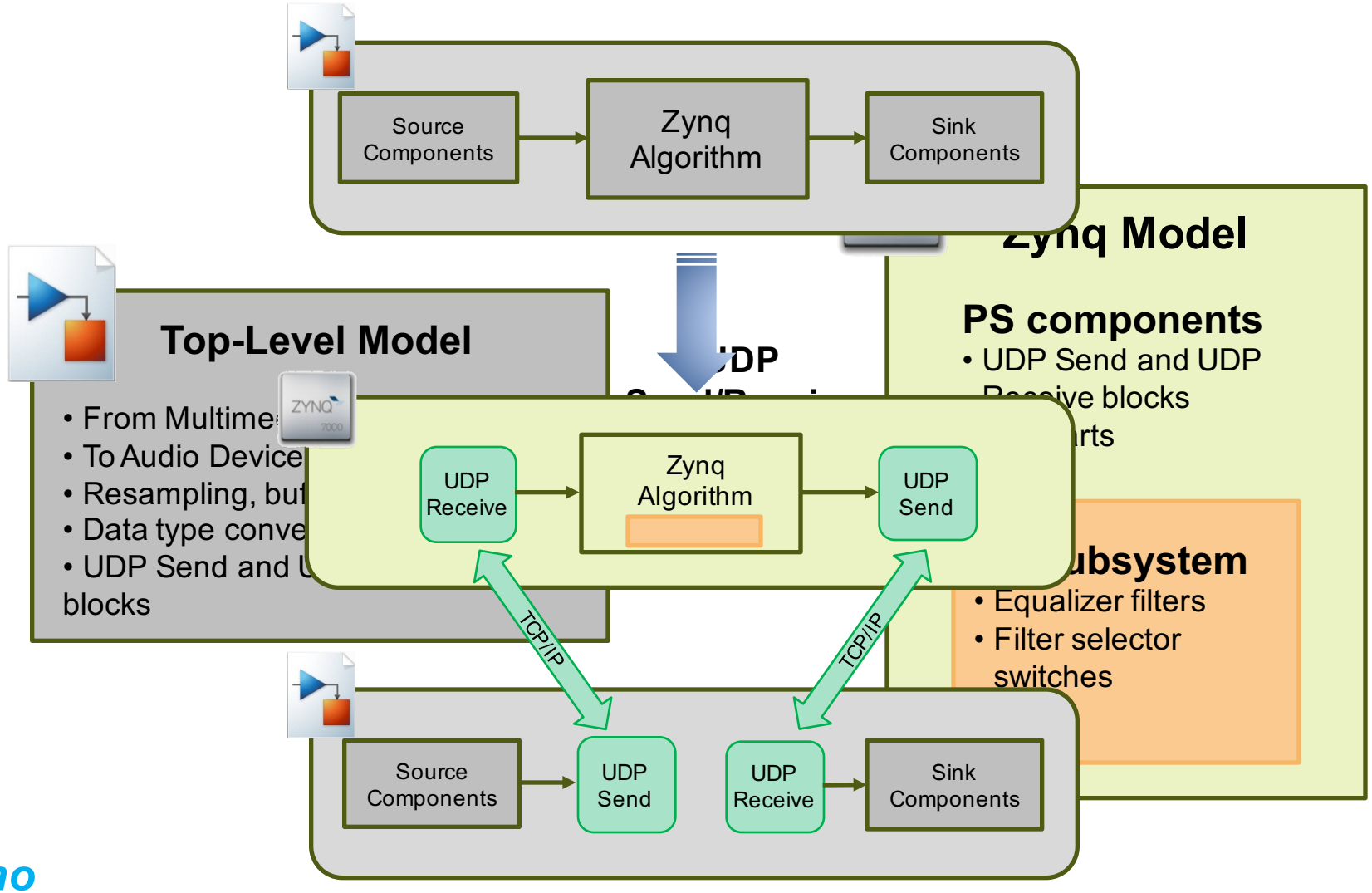
Verification through UDP Interface





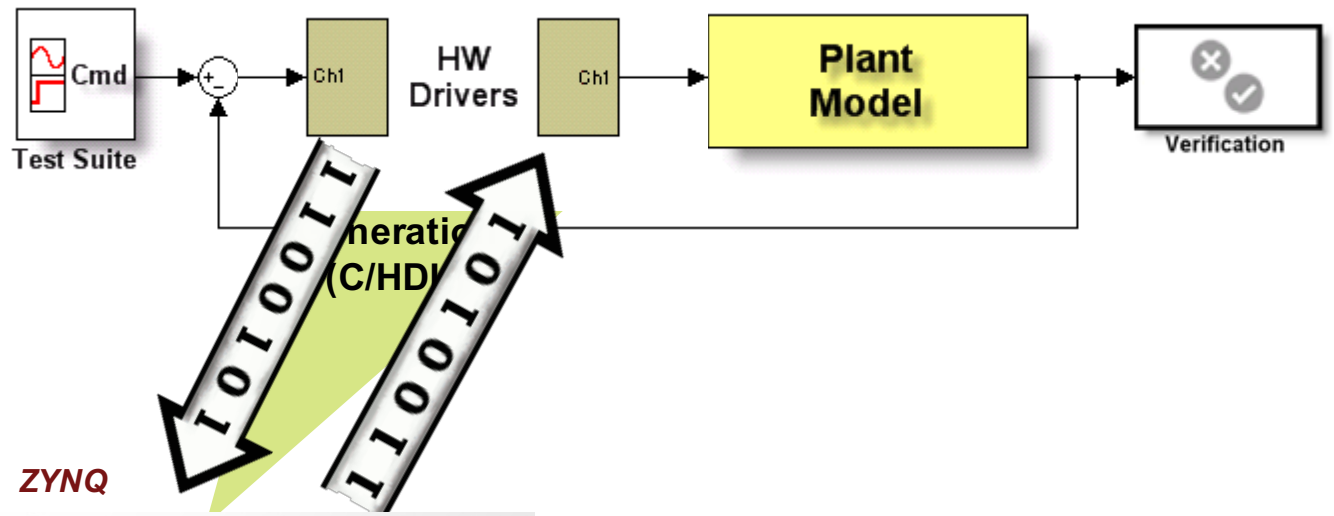
Fast prototyping, iteration, and live probing/tuning directly on ZYNQ hardware

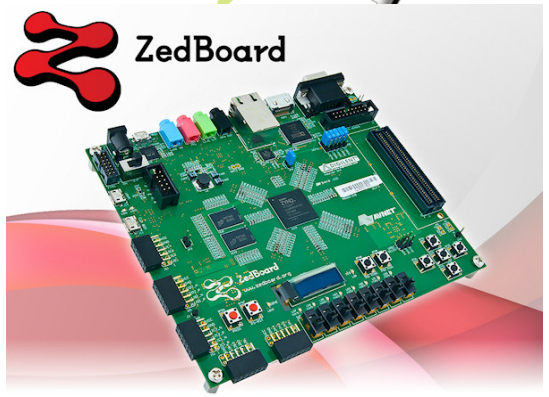
Design Partitioning for UDP



demo

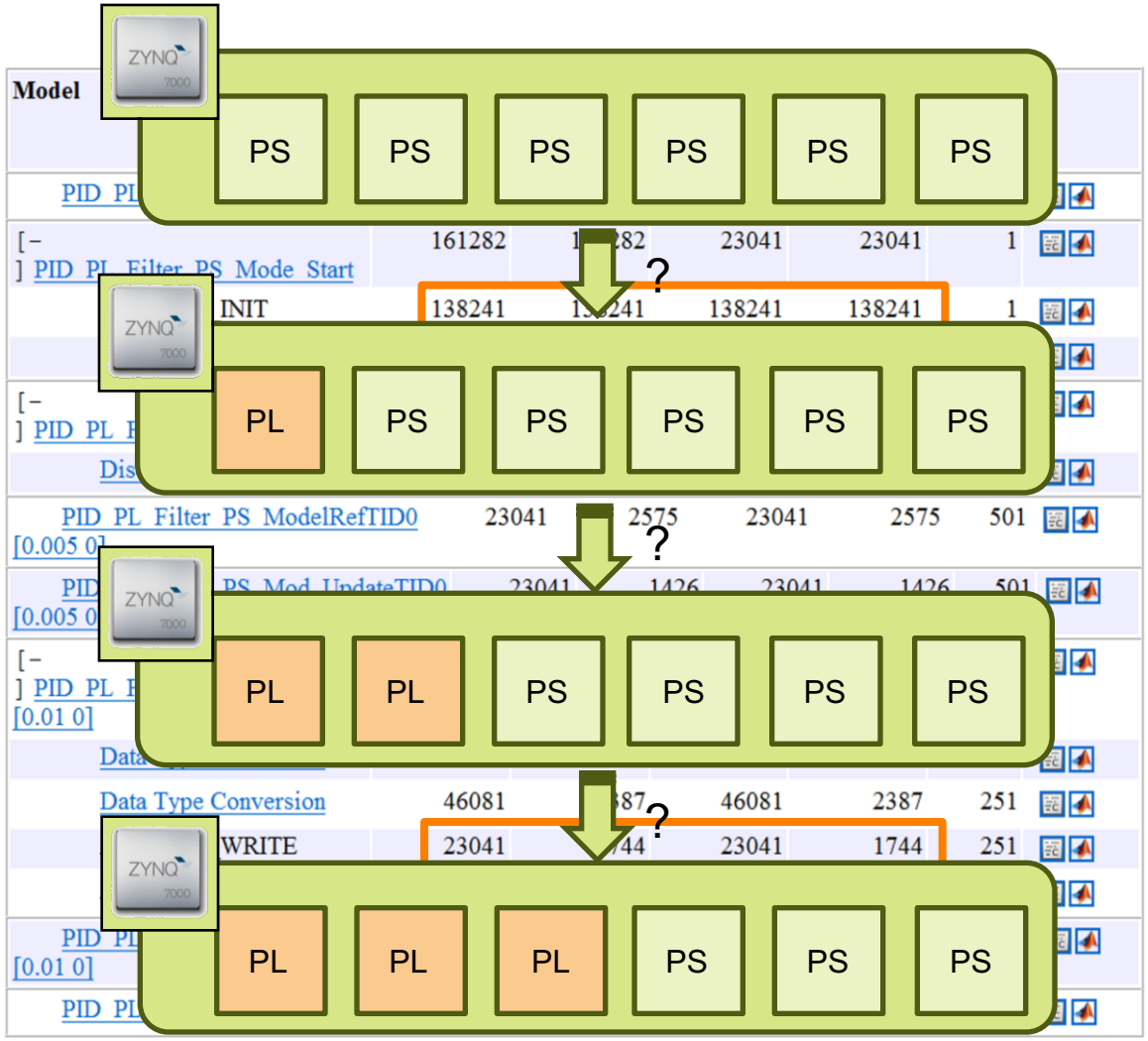
Processor-In-the Loop(PIL) Verification





Help you evaluate the behavior of a candidate algorithm on the target both PS/PL and profile the execution times

Partitioning Scheme via PIL



demo

Verification Tradeoffs

Feature	External Mode	PIL	UDP
Real-time execution	✓Yes	✗No	✓Yes
Parameter tuning	✓Yes	✗No	✗No
Test bench options	✗Limited	✓Unlimited	✓Unlimited
Code verification	✗No	✓Yes	✗No
Execution profiling	✗No	✓Yes	✗No
Single model	✓Yes	✓Yes/No	✗No
Data synchronization	✗Limited	✓Yes	✗Limited

Agenda

- Introduction
  - What is Zynq?
  - Design Challenges
- ZYNQ Design Process
  - MBD on Programmable SoC
  - Code Generation
  - Workflow
- Verification & Partitioning
  - Parameter Tuning
  - UDP Interface
  - Processor In the Loop
- Advanced Features
- Conclusions

Recently updated features in R2015a/R2015b

- R2015a
  - Internal-interface API in custom reference design
  - Zynq AXI Stream support
  - Support Front Digital IO of Speedgoat IO331 board
  - Save target setting as model properties

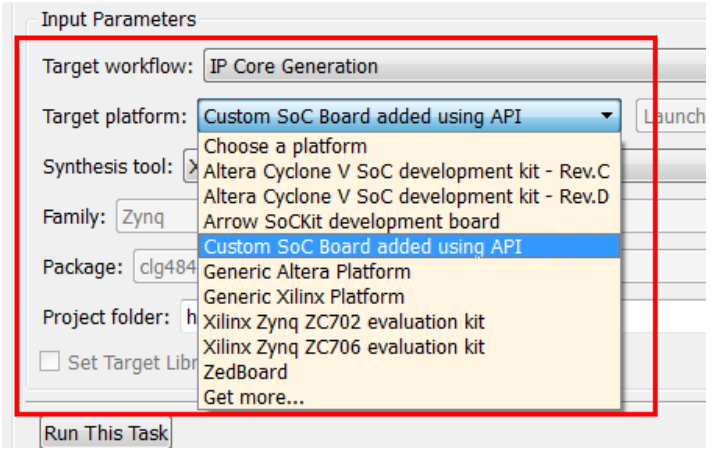
- R2015b
  - Command-line API for WFA
  - Export/Import between API and WFA
  - Map tunable parameters to AXI interfaces
  - AXI stream vector mode
  - Internal-interface API in custom reference design

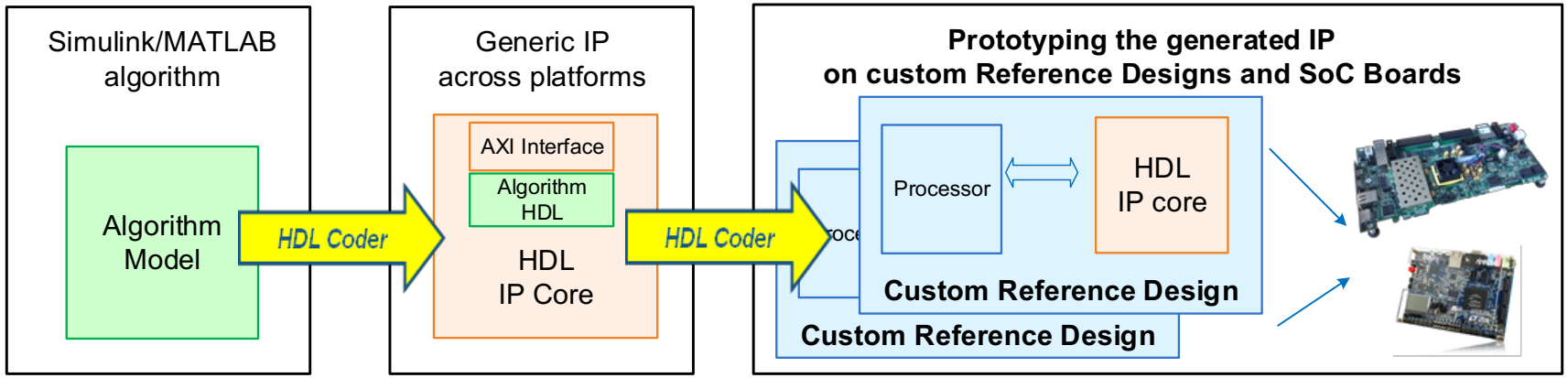
Custom Board and Reference Design API for IP Core Generation Workflow

R2015a

Define your own Zynq or Altera SoC Board and Reference Design

- Enable fast on-board prototyping and iteration
- Access to SW interface model generation, AXI driver, External Mode

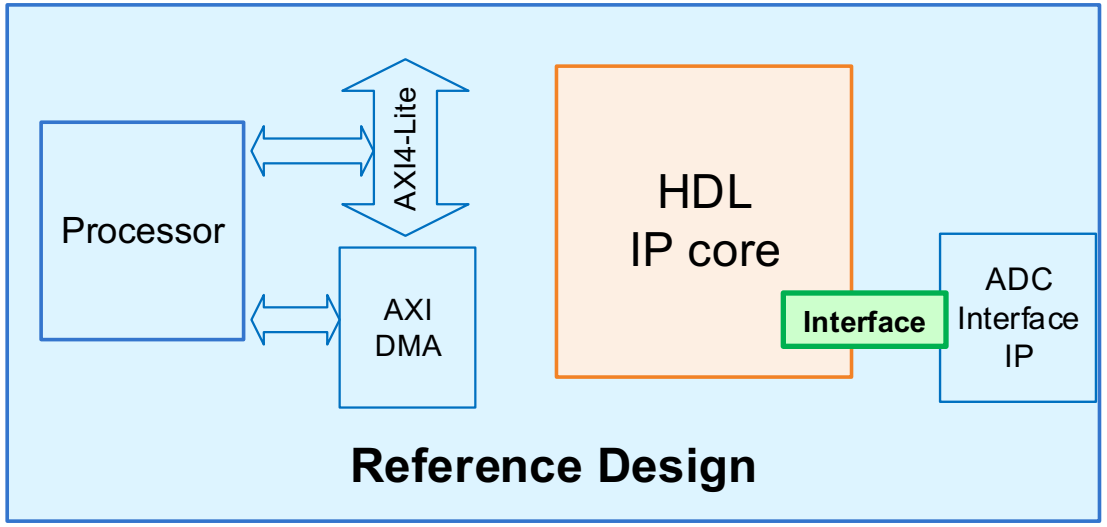




Custom Reference design API extension for Internal Interface

R2015b

- Define an interface that connects to another IP in the reference design

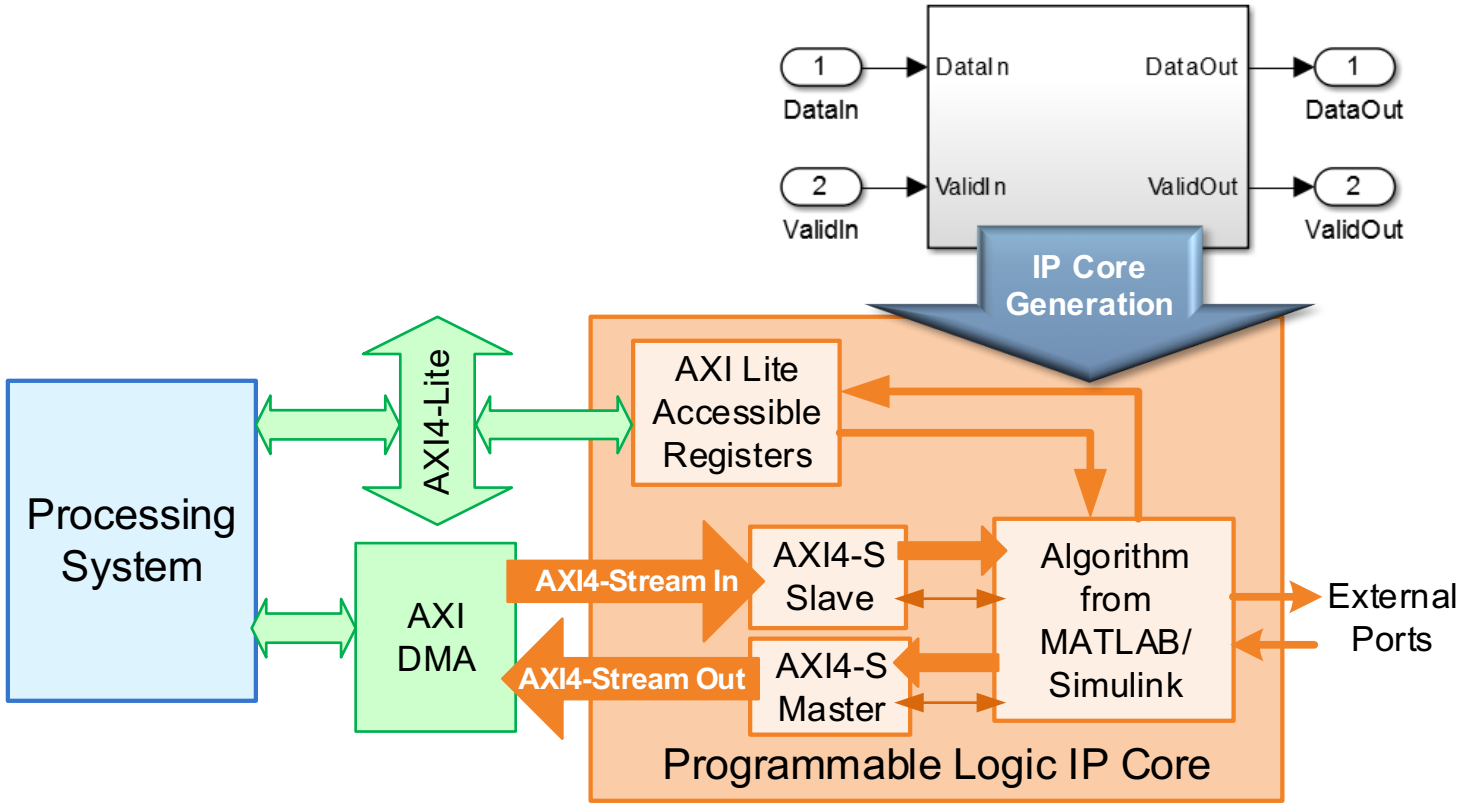



```
% add reference design internal interface
hRD.addInternalIOInterface( ...
    'InterfaceID',    'ADC Data In', ...
    'InterfaceType',  'IN', ...
    'PortName',       'ADCDataIn', ...
    'PortWidth',       16, ...
    'InterfaceConnection', 'hdlcoder_adc_interface_ipcore_0/DataOut');
```

Zynq Streaming Interface Support

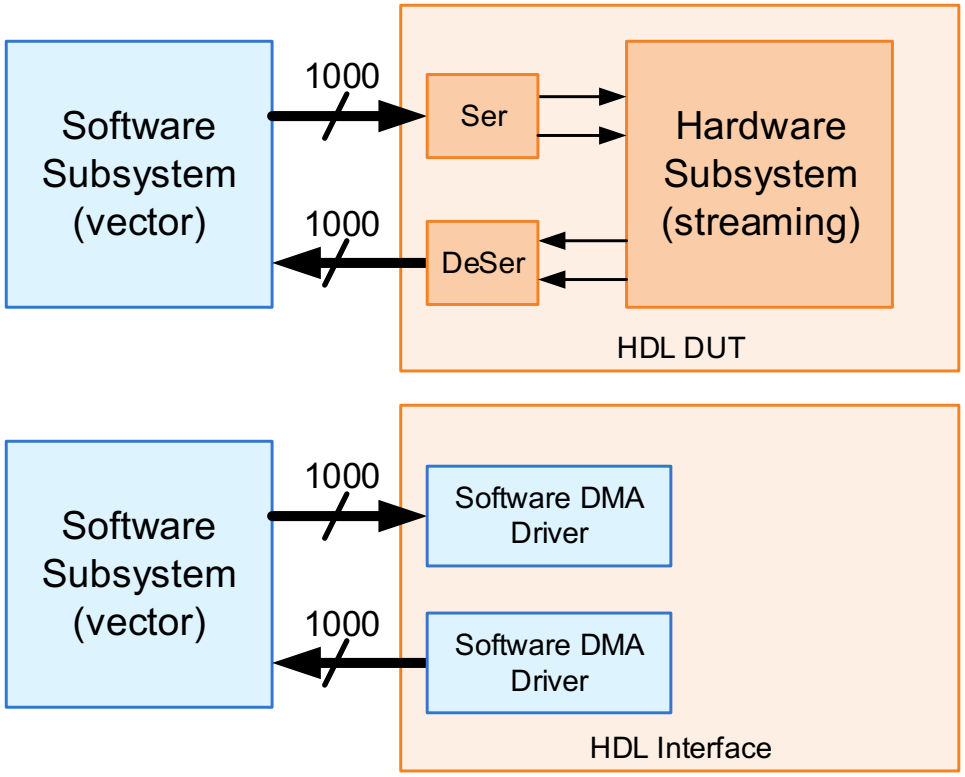
R2015a

- Generate HDL IP core with AXI4-Stream interface
- Enable high speed data transfer
- Simplify streaming protocol



AXI4-Stream Vector Mode

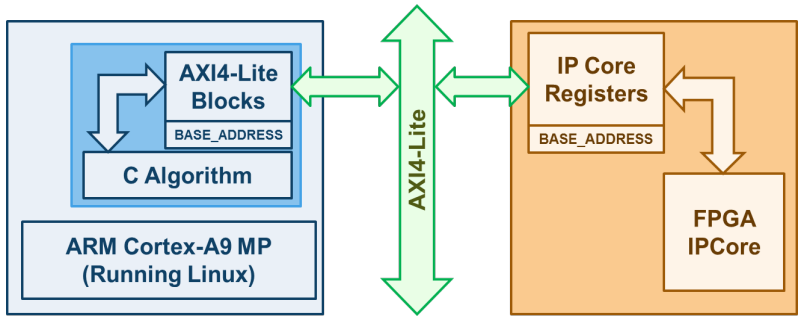
- Modeling HW and SW together
- Automatic generation of SW DMA driver
- Focus on HW/SW Rapid Prototyping



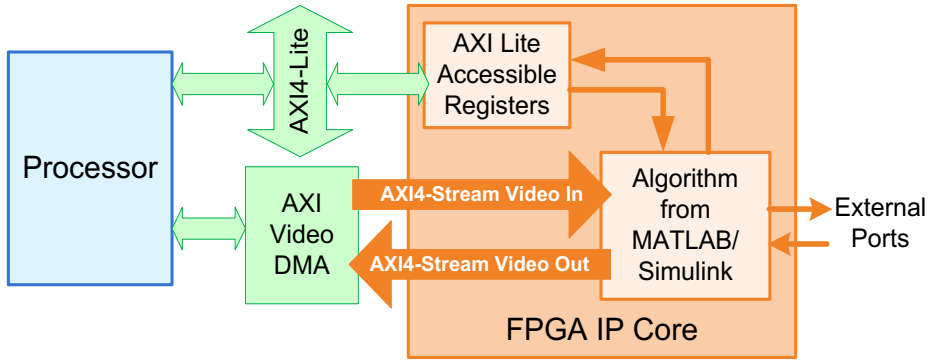
Agenda

- Introduction
  - What is Zynq?
  - Design Challenges
- ZYNQ Design Process
  - MBD on Programmable SoC
  - Code Generation
  - Workflow
- Verification & Partitioning
  - Parameter Tuning
  - UDP Interface
  - Processor In the Loop
- Advanced Features
- Conclusions

Conclusions: Design Concept Summary



AXI4-Lite Interface



AXI4-Stream Interface

- Focus on algorithm and system design
- Stay on higher level of abstraction
- Automatic code generation and HW/SW integration
- Fast Prototyping and Easy integration with IDE
- Partitioning through Profiling

Conclusions: Related Training Summary

Training Title	Details
Signal Processing with MATLAB	Signal Analysis and Algorithm Design (2 days)
Image Processing with MATLAB	Image Analysis and Processing (2 days)
Signal Processing with Simulink	Signal Processing Based Modeling & Dynamic Simulation (3 days)
Communication System Modeling with Simulink	Communication System Modeling & Simulation (1 day)
Stateflow for Logic-Driven System Modeling	Flow Chart and FSM Modeling (2 days)
Interfacing MATLAB with C	Interfacing MATLAB and C Code via APIs (1 days)
MATLAB to C with MATLAB	using MATLAB Coder (2 days)
Embedded Coder for Produc Generation	stem from Simulink Model (3 days)
Generating HDL Code from Simulink	**HDL Code Generation & Verification (2 days)**
Programming Zynq with MATLAB and Simulink	**HW/SW Co-design on Zynq (include ZedBoard) (2 days)**

Questions?

Thank You

© 2016 The MathWorks, Inc.