Tool Flow for Heterogeneous Systems

2024.1

AMD
together we advance_

Development Platforms for ALL Developers

Software programmable platform

- Deploy applications from a standard framework
- Accelerate with the Versal platform

AI & Data Scientists

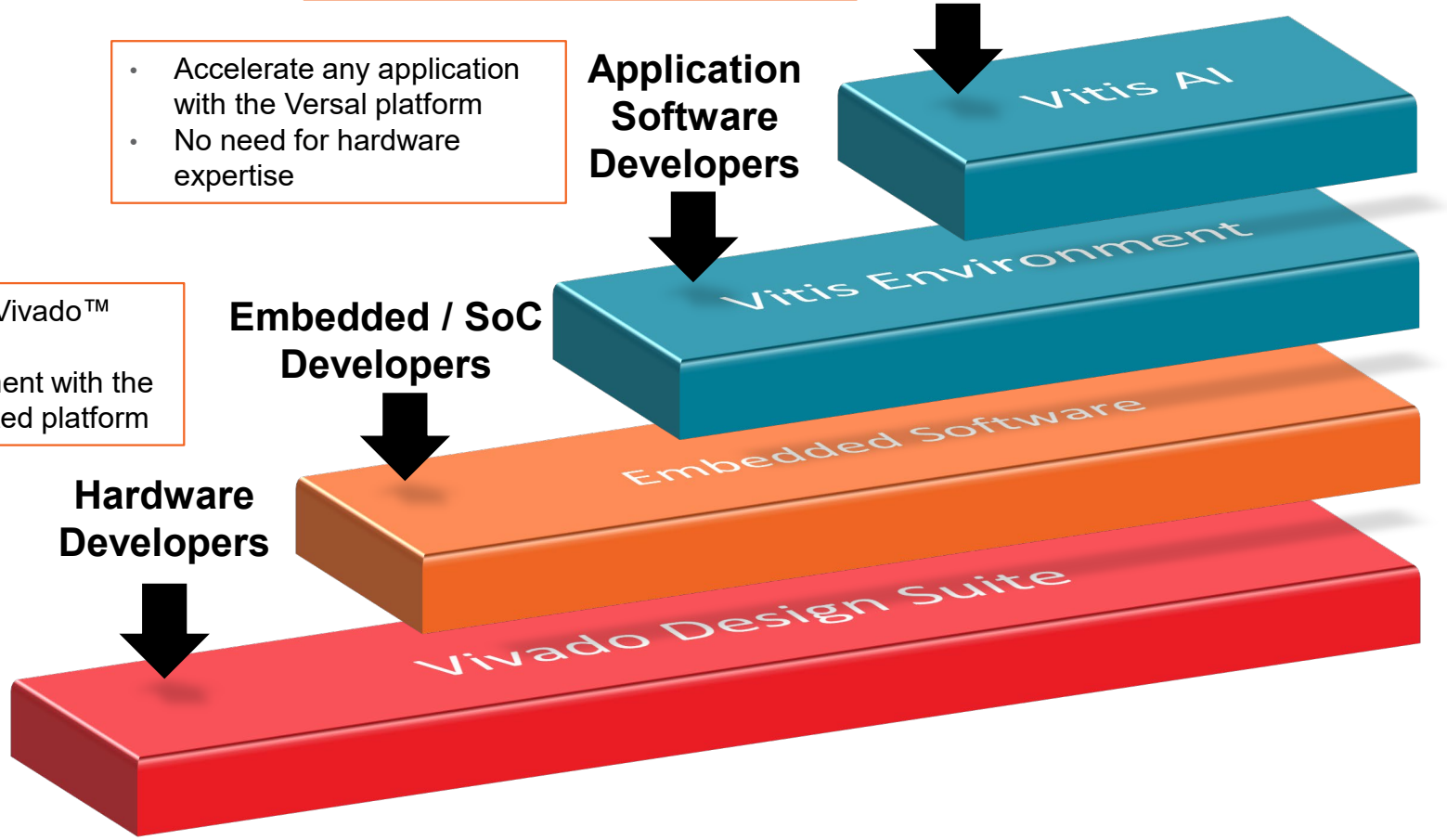
- Accelerate any application with the Versal platform
- No need for hardware expertise

Application Software Developers

- Design with the Vivado™ Design Suite
- Faster development with the Versal™ integrated platform

Embedded / SoC Developers

Hardware Developers



Vitis Software Platform Development Environment

For developing designs with:

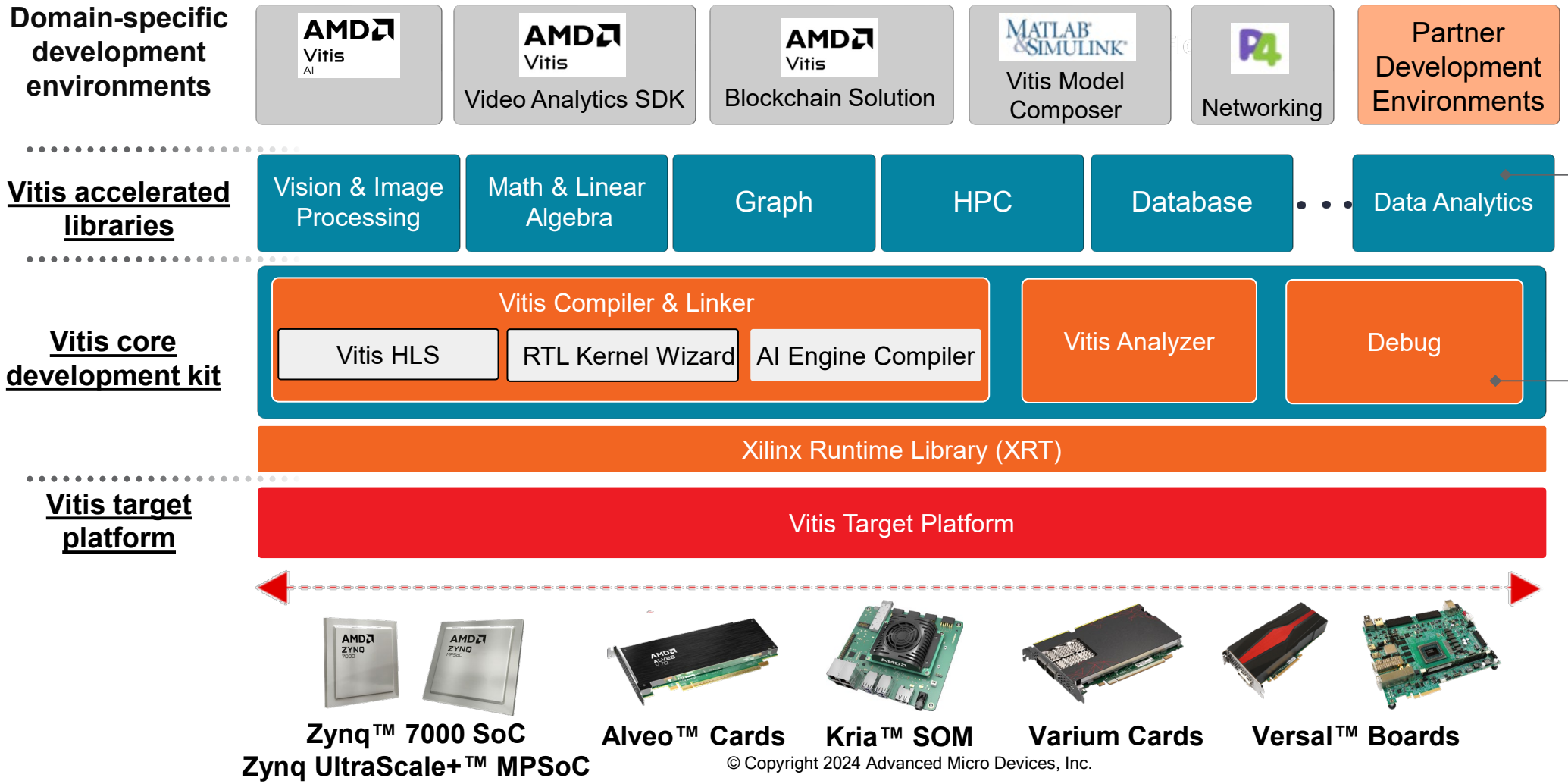
- FPGA fabric
- Arm® processor subsystems
- AI Engines

Includes the following tools:

- Vitis™ Embedded
- Compilers and simulators
  - Vitis AIE DSP design tools
- Vitis HLS
- Vitis Model Composer
- Vitis libraries

Vitis Unified Solution Stack
For Heterogenous Compute, Edge to Cloud

Vitis™ Unified IDE: Environment for development of applications for our embedded processors



- Optimized FPGA acceleration
- Libraries for math and domain-specific applications

- Build code
- Fix problems
- Analyze
- API and drivers

Vitis Unified IDE

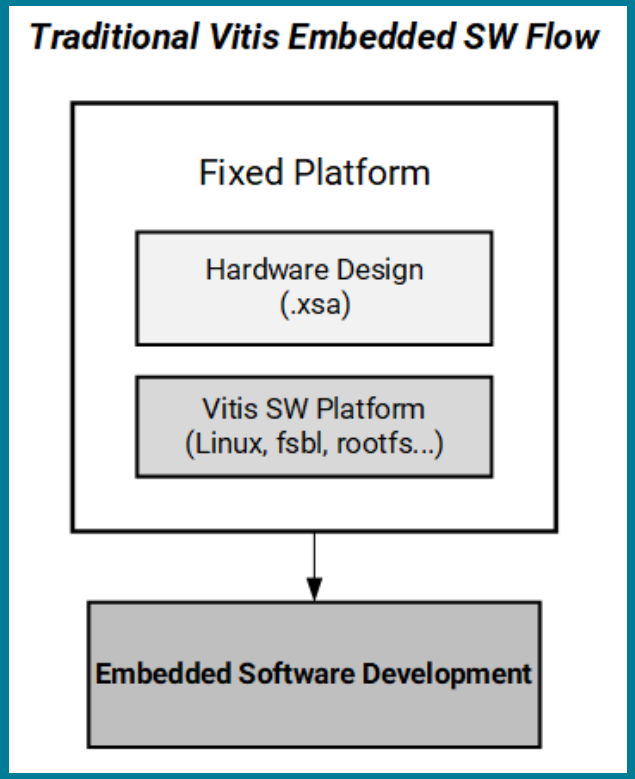
Unified software platform to enable system design for hardware and software developers

Compile | Run | Debug | Analyze

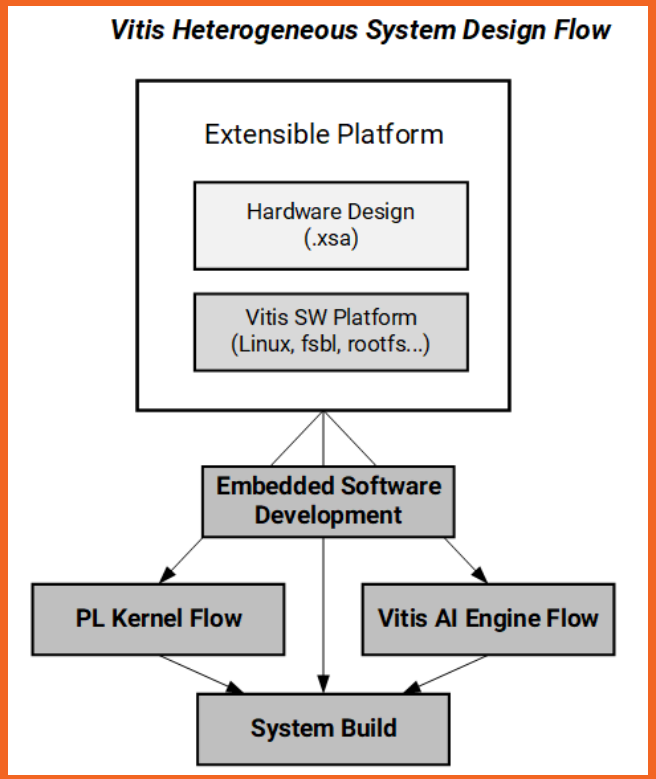
Application creation from example designs or templates	Creating platforms from Vivado™ Design Suite-generated hardware designs and generating BSPs for software development	Flash programming
Platform and application configuration and build		Component type-based project management in the IDE
Running, debugging, or profiling applications on hardware		Project management using user scripts
Multiple local or remote hardware connection management	Bootable image creation	Source code version control
Support for multiple devices and processors	Device configuration	Both GUI and command line interface support

Vitis Embedded System Design Flows

Embedded system elements: Vivado™ Design Suite-exported hardware designs, Vitis™ extensible platforms, Arm® processor applications, PL kernels, and AI Engine graph applications

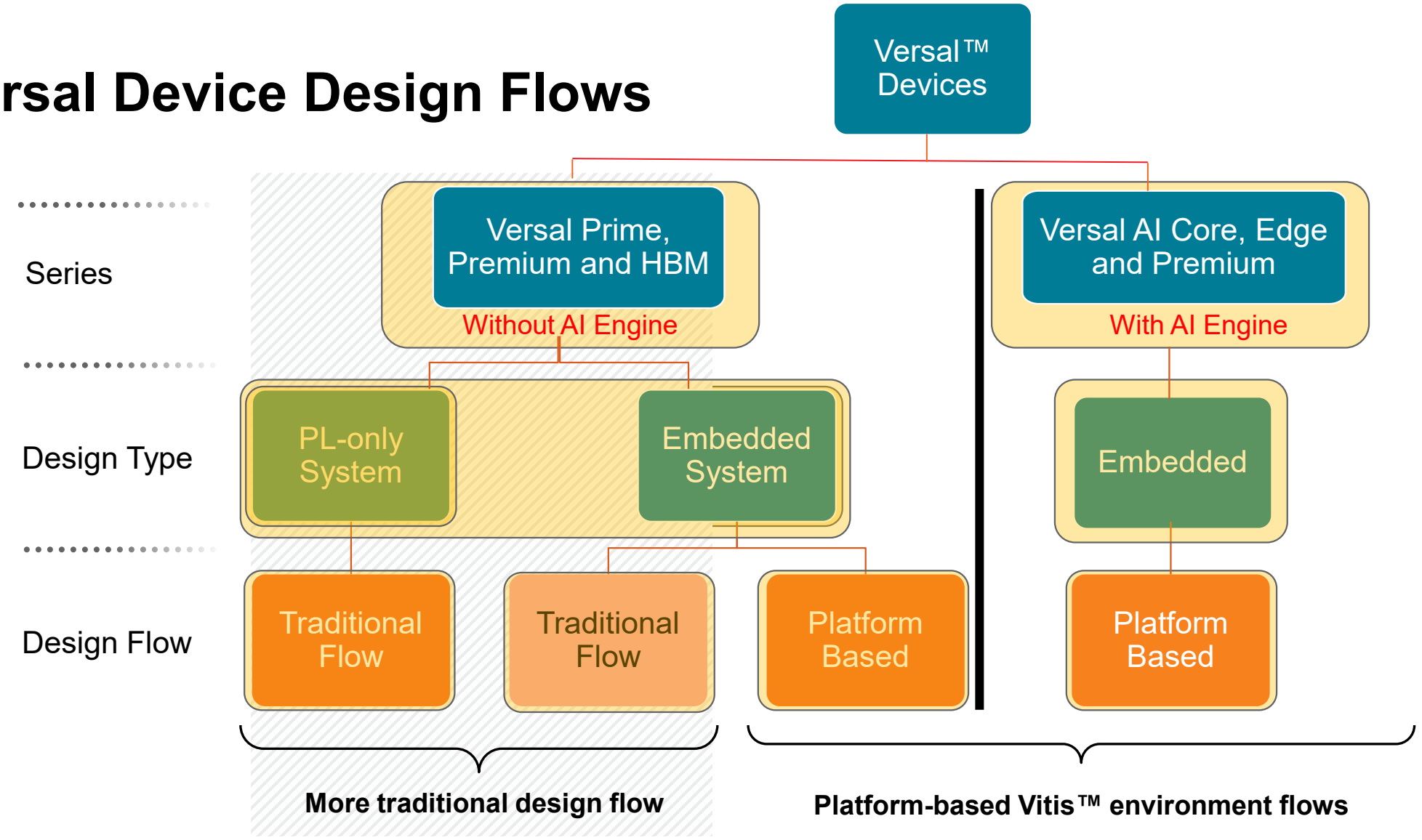


Relies on hardware design generated from Vivado IDE and software applications



Uses devices such as Versal™ adaptive SoCs, Kria™ SOMs, and Zynq™ UltraScale+™ MPSoCs

Versal Device Design Flows



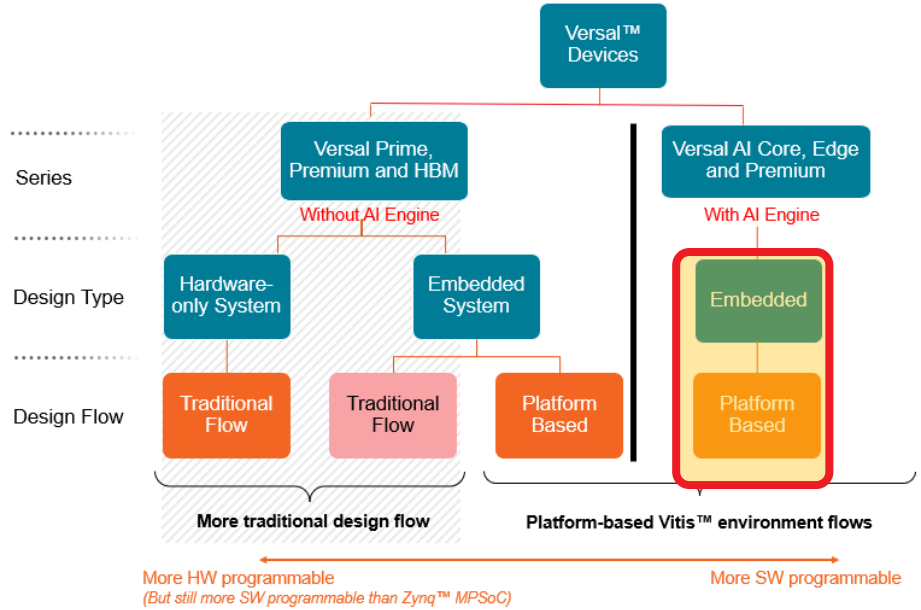
Embedded AI Engine System

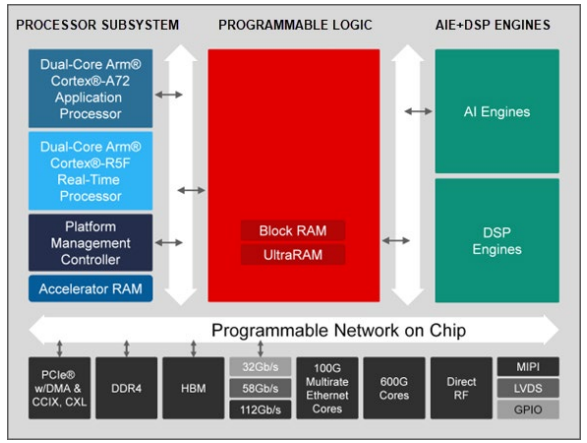
Comprises of:

- Embedded processor
- Acceleration logic:
  - Traditional PL
  - AI Engines

For the Versal™ device:

- Embedded processing system is running on the Arm® Cortex®-A72 and Cortex-R5F processors
- Hardware content in the PL and algorithmic content in the AI Engines
- Created using platform-based design flow





Using the Vivado Design Suite in the Design Flows

Key component in all Versal™ adaptive SoC design flows

Vivado™ tools can perform:

- Logic simulation
- NoC compilation
- Logic synthesis and implementation
- Design rule checks (DRC) and design methodology checks
- Power and thermal analysis
- Constraint definition and timing analysis
- I/O and clock planning
- Visualization of design logic
- Implementation results analysis
- Programming and debugging

AMD Vivado

Primary Use Models – Vivado Tools in the Design Flows

Traditional Design Flows

Creating RTL and IP designs

- Use Vivado™ tools and Vivado IP integrator to automate design assembly

Platform-based Design Flows

Creating and packaging RTL kernels

- Use Vivado IP packager to package RTL kernels into XO file

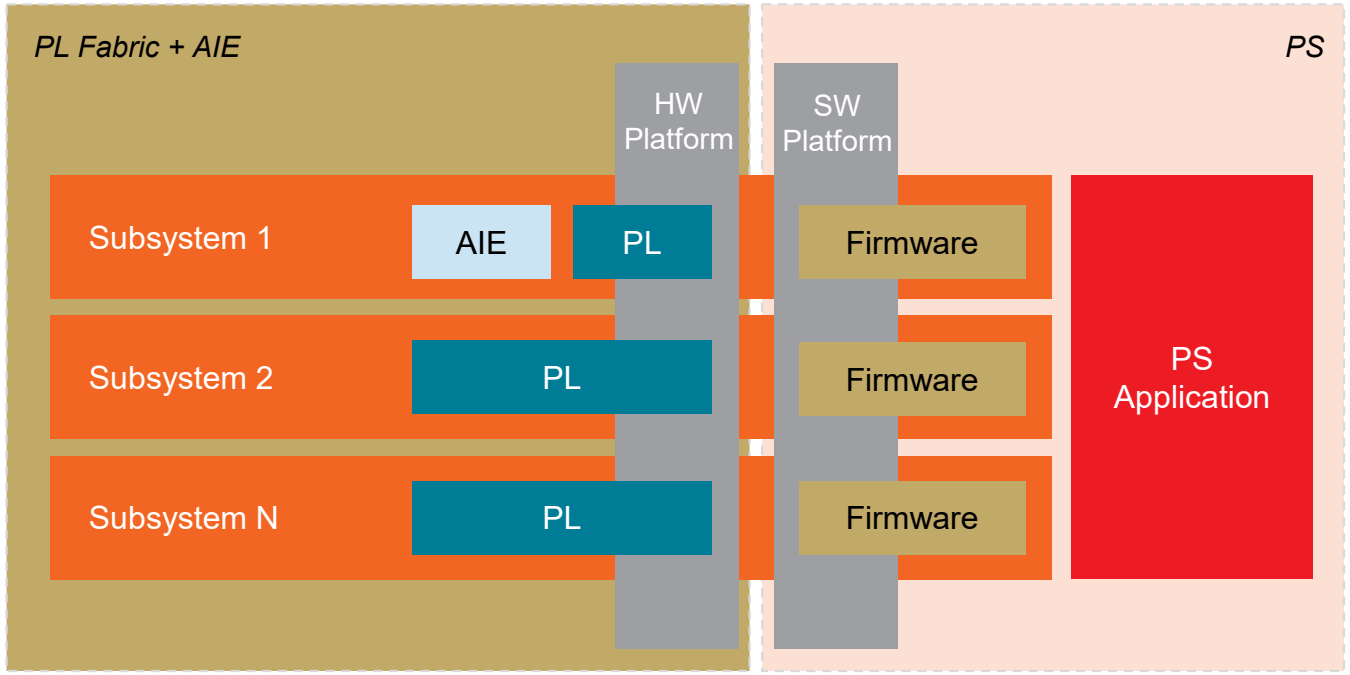
Creating and generating platforms

- Create extensible hardware platform that is then extended with processing system using Vitis™ tools
- Includes basic system-level resources shared by all accelerators

For platform-based design flow, AMD provides standard platforms as starting points, which can be customized and regenerated by the Vivado IP integrator

Subsystems and Platforms

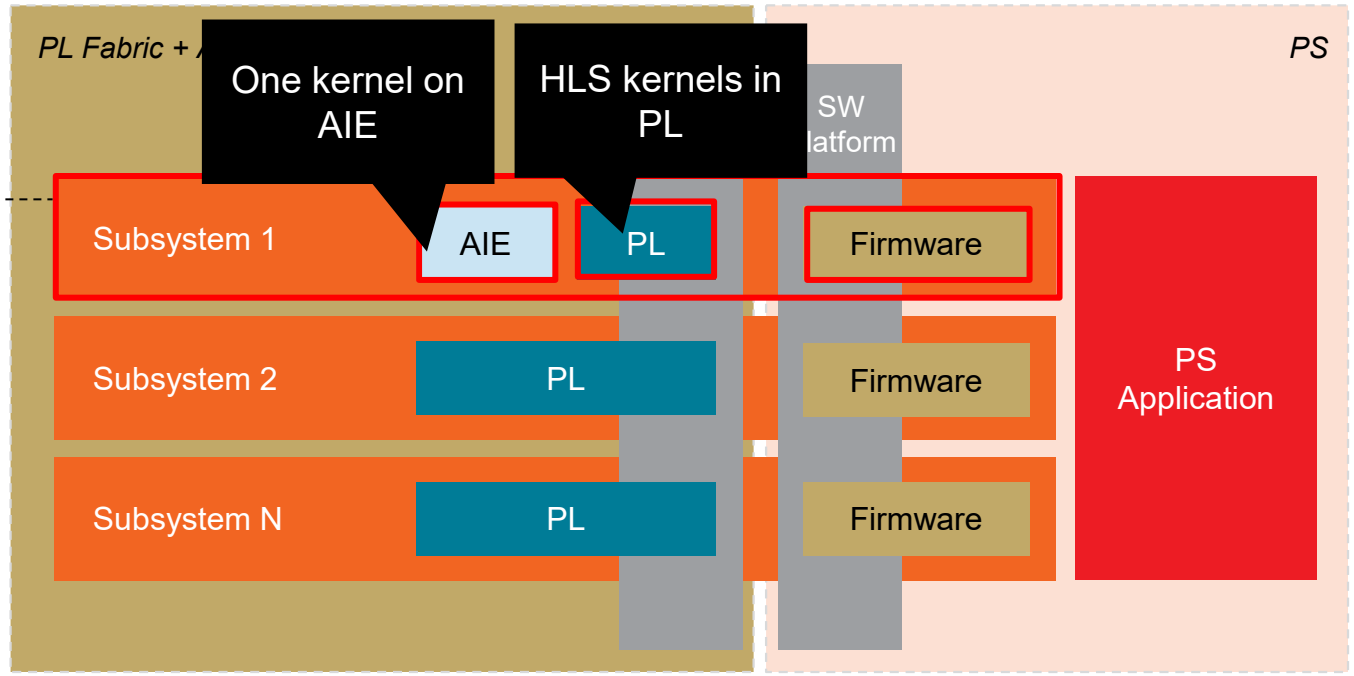
Approach for building a Versal™ system using the Vitis™ environment: Divide-and-conquer approach



Subsystems and Platforms

Subsystem

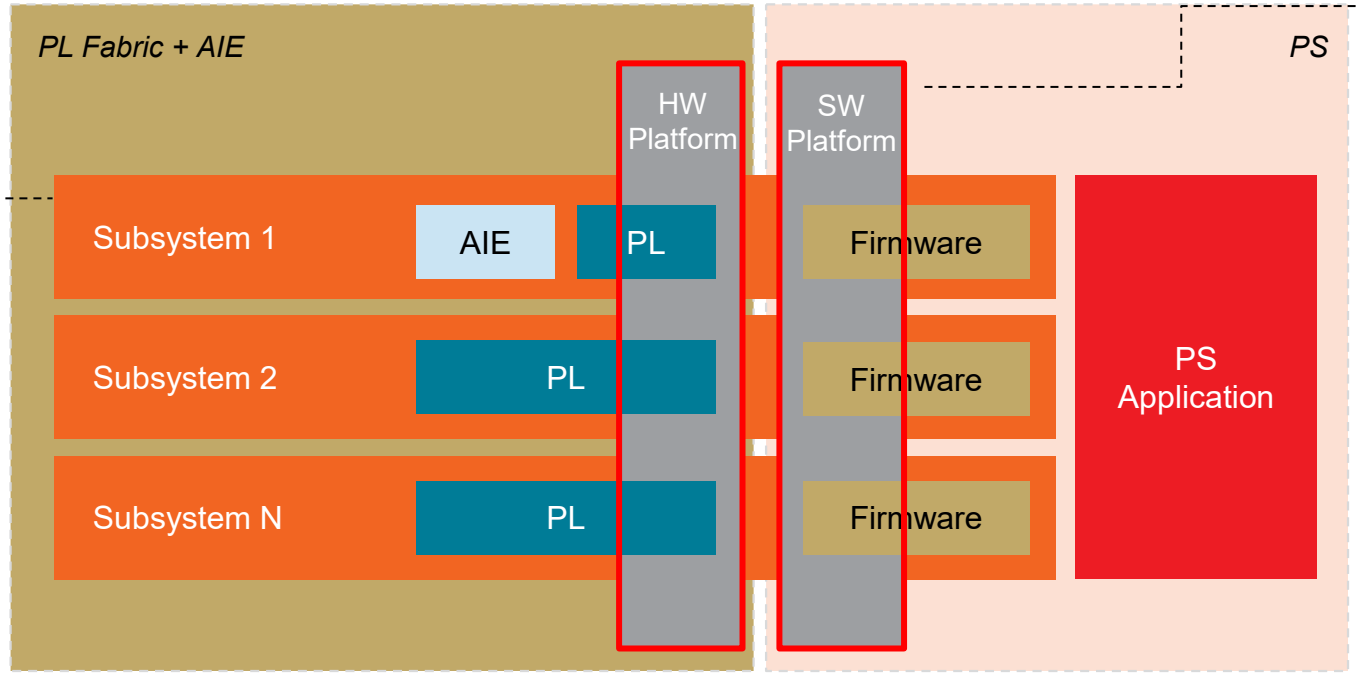
- Functions under PS supervision
- Developed and tested independently
- Interacts via shared memory
- v++ --link to add PL components
- v++ --package to add AIE and PS software



Subsystems and Platforms

Subsystem

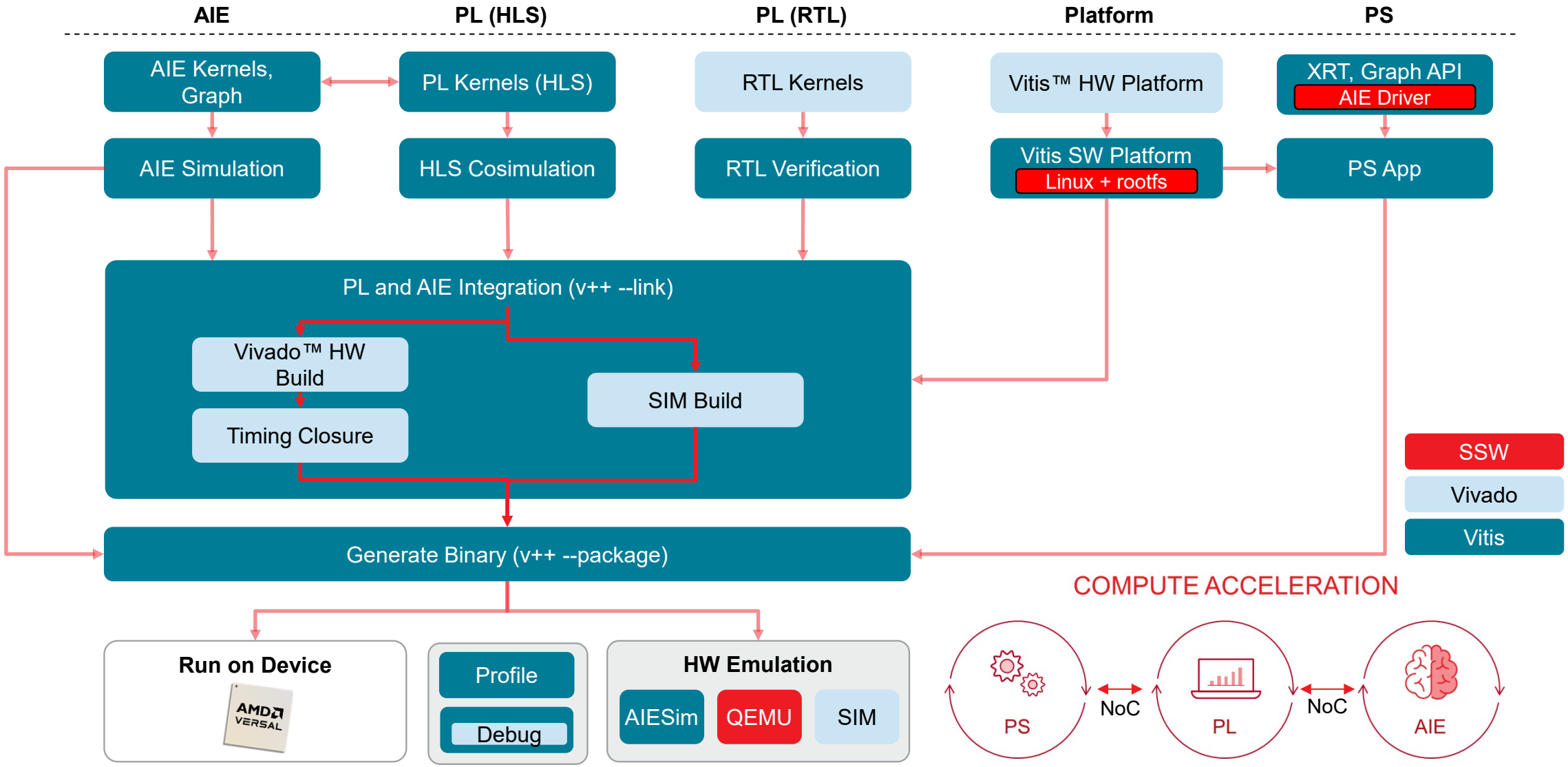
- Functions under PS supervision
- Developed and tested independently
- Interacts via shared memory
- v++ --link to add PL components
- v++ --package to add AIE and PS software



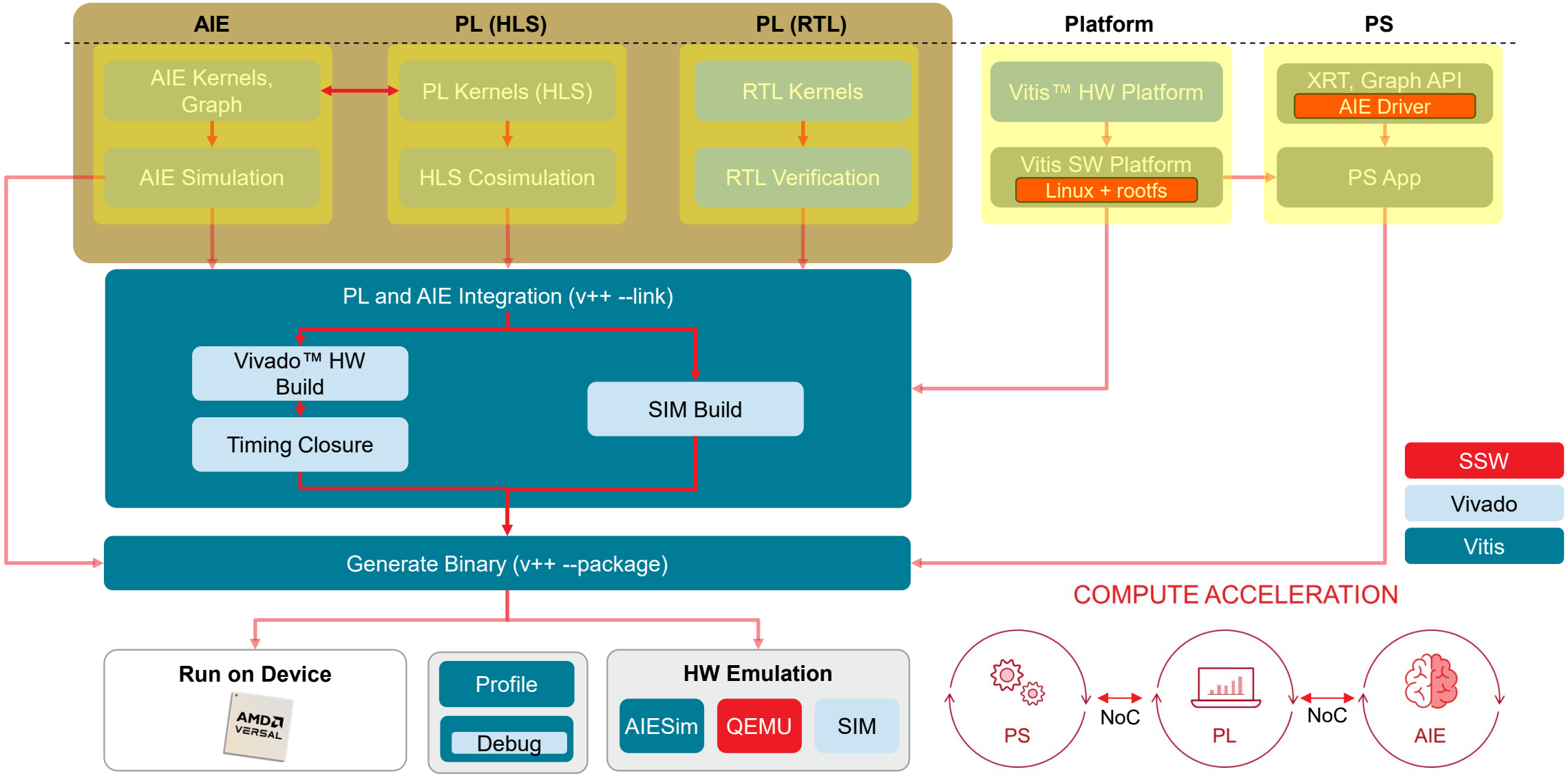
Platform

- Insulates developers from low-level details
- Enables focus on application development (SW, PL, or AIE)
- IPI block design containing essential IPs
- Generated using the PetaLinux tool suite

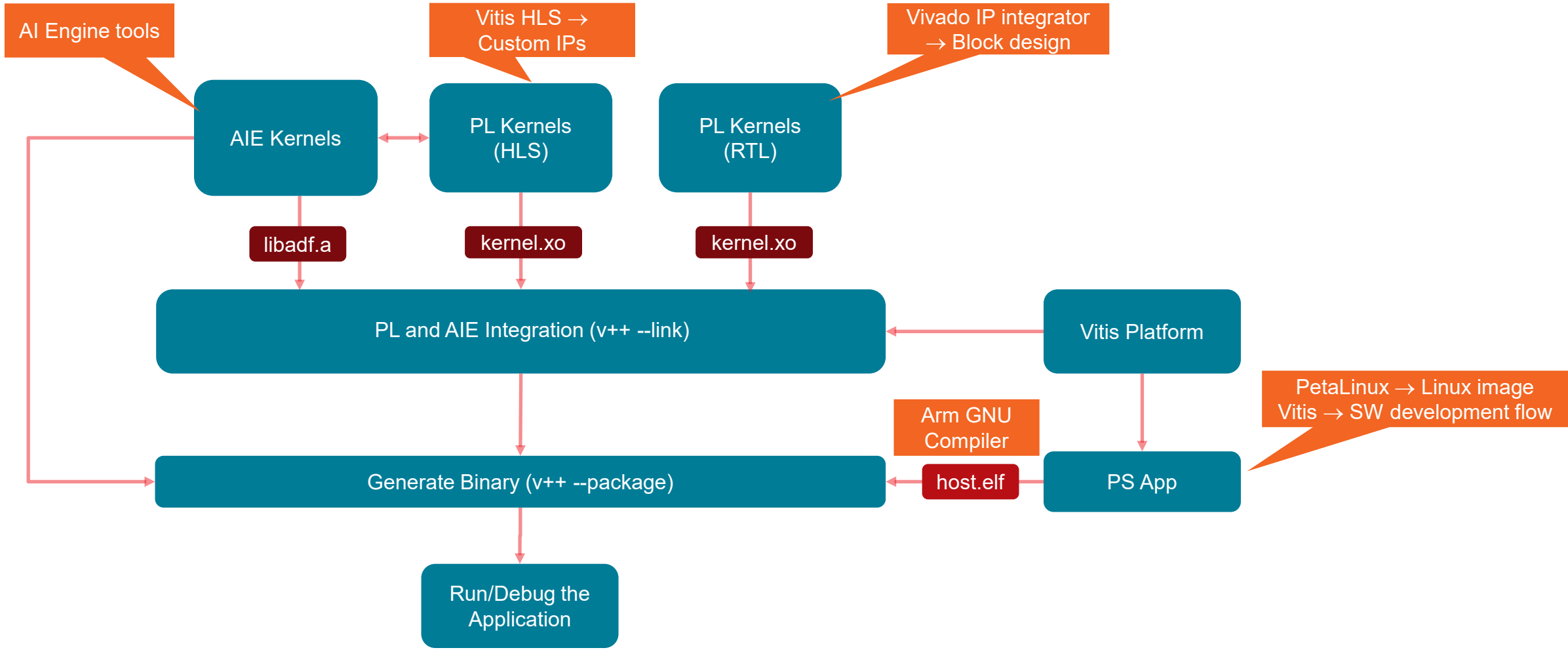
Vitis Tool Flow for Versal Devices



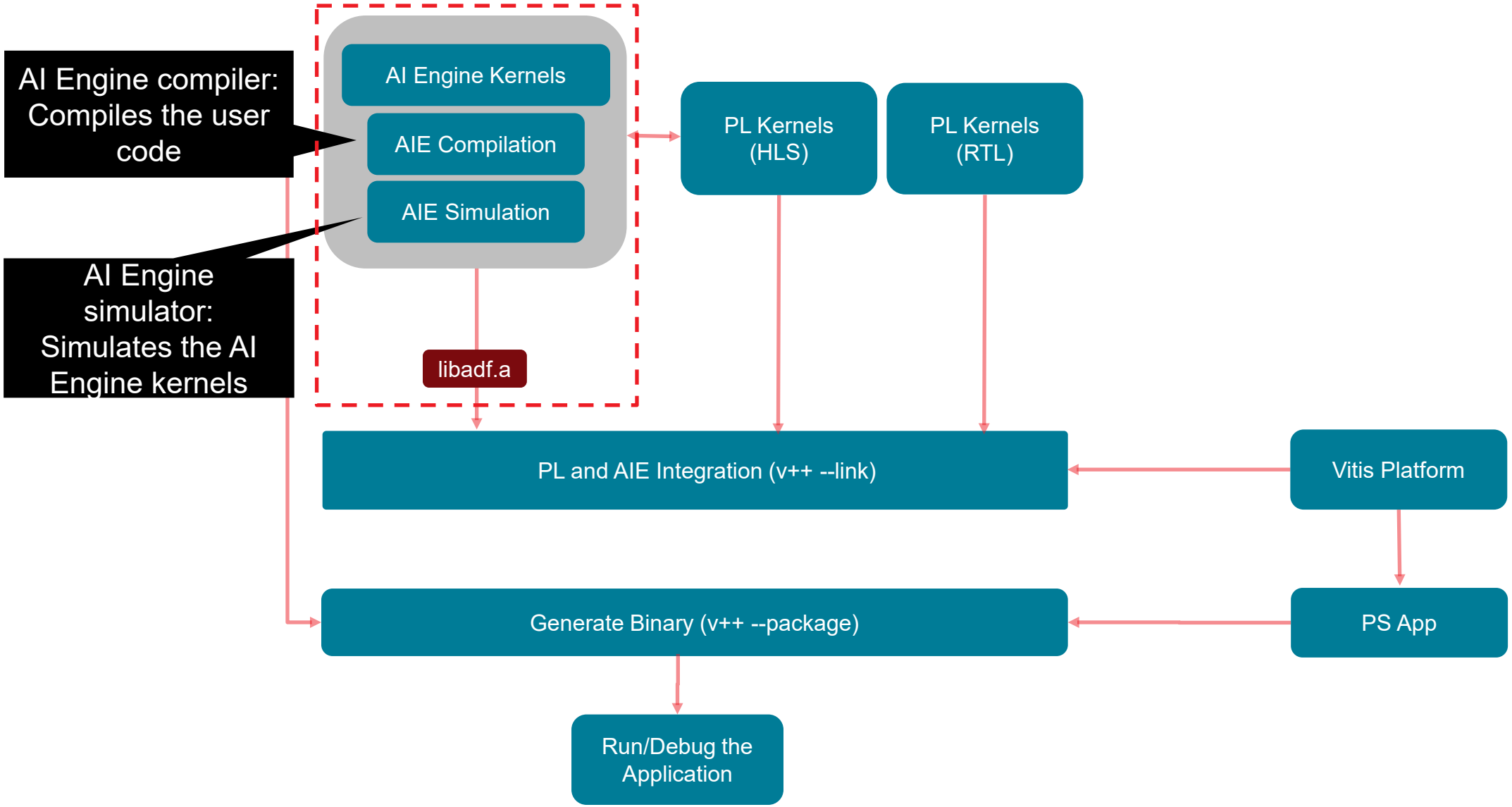
Vitis Tool Flow for Versal Devices



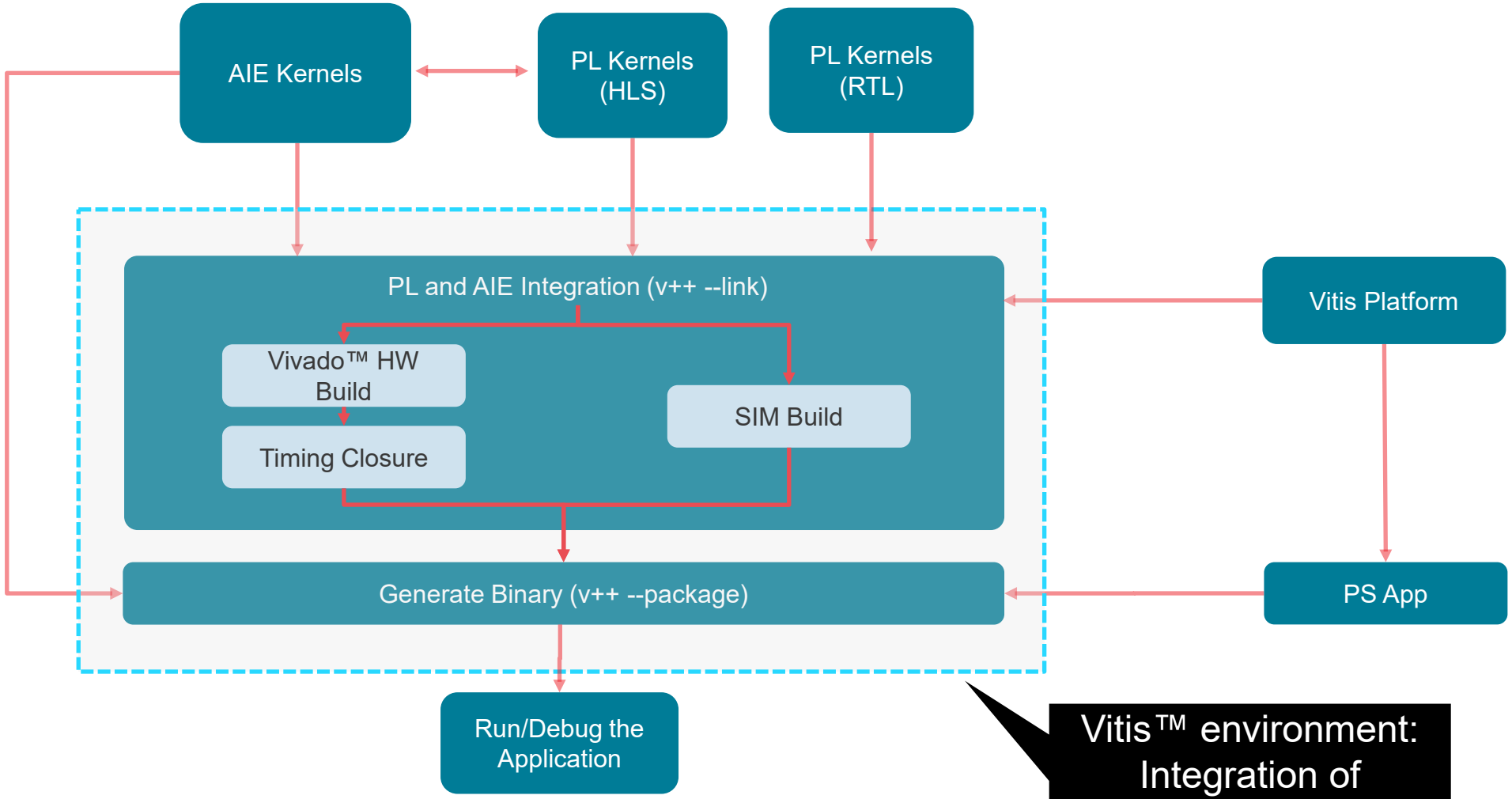
Vitis Tool Flow – Tools for Each Domain



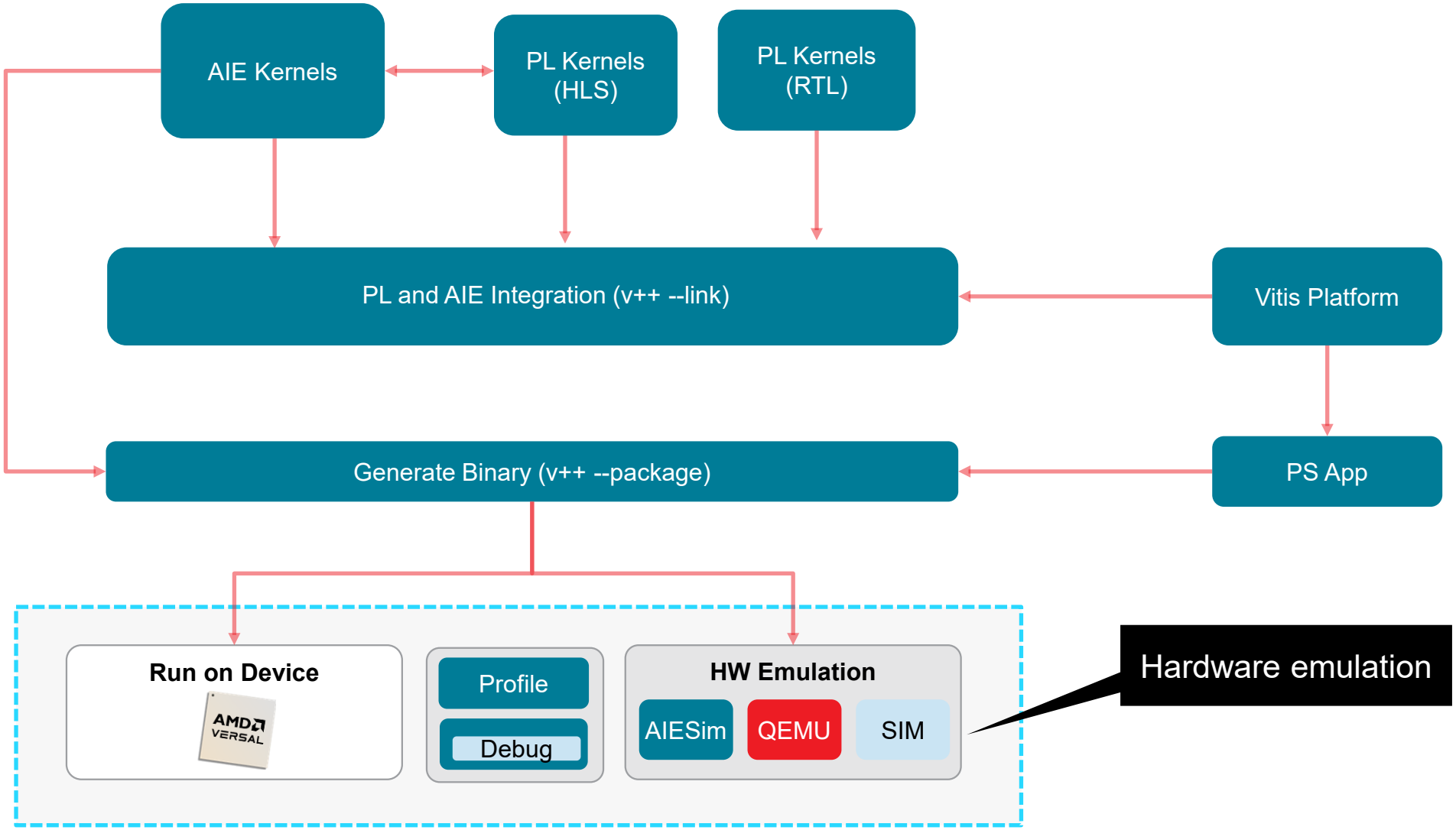
Vitis Tool Flow – AIE Compilation and Simulation



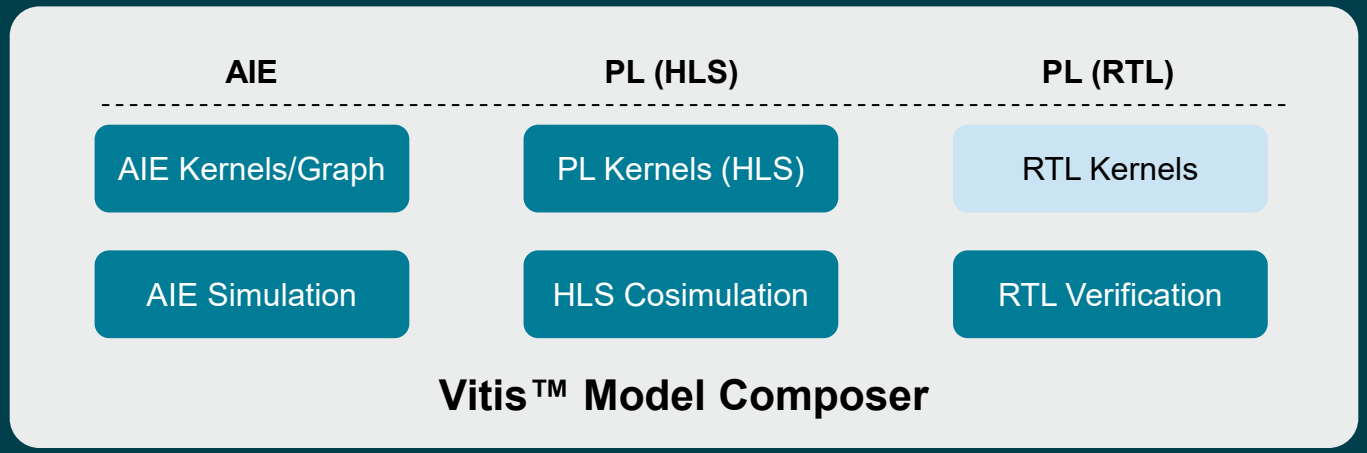
Vitis Tool Flow – Versal Integration



Vitis Tool Flow – Run/Debug

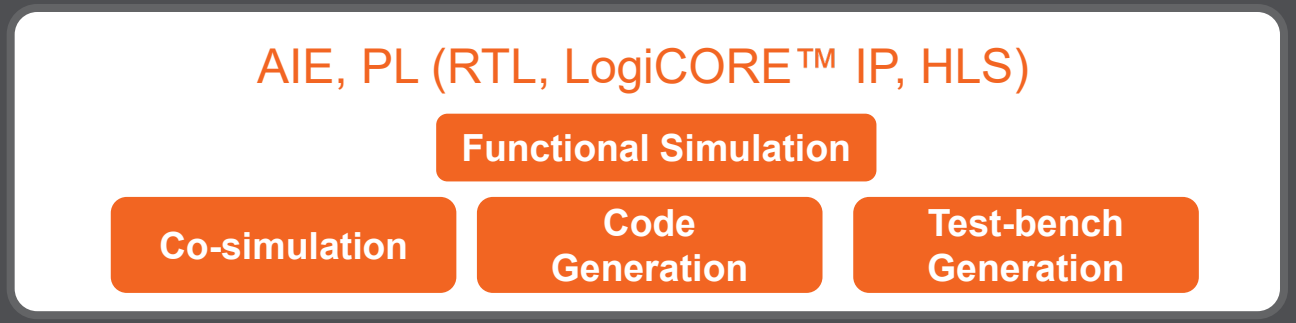


Vitis Model Composer



- Model-based design tool
- Automatic code generation
- Library of HDL, HLS, and AI Engine blocks
- Importing of custom HDL, HLS, and AI Engine code as blocks

Vitis Model Composer

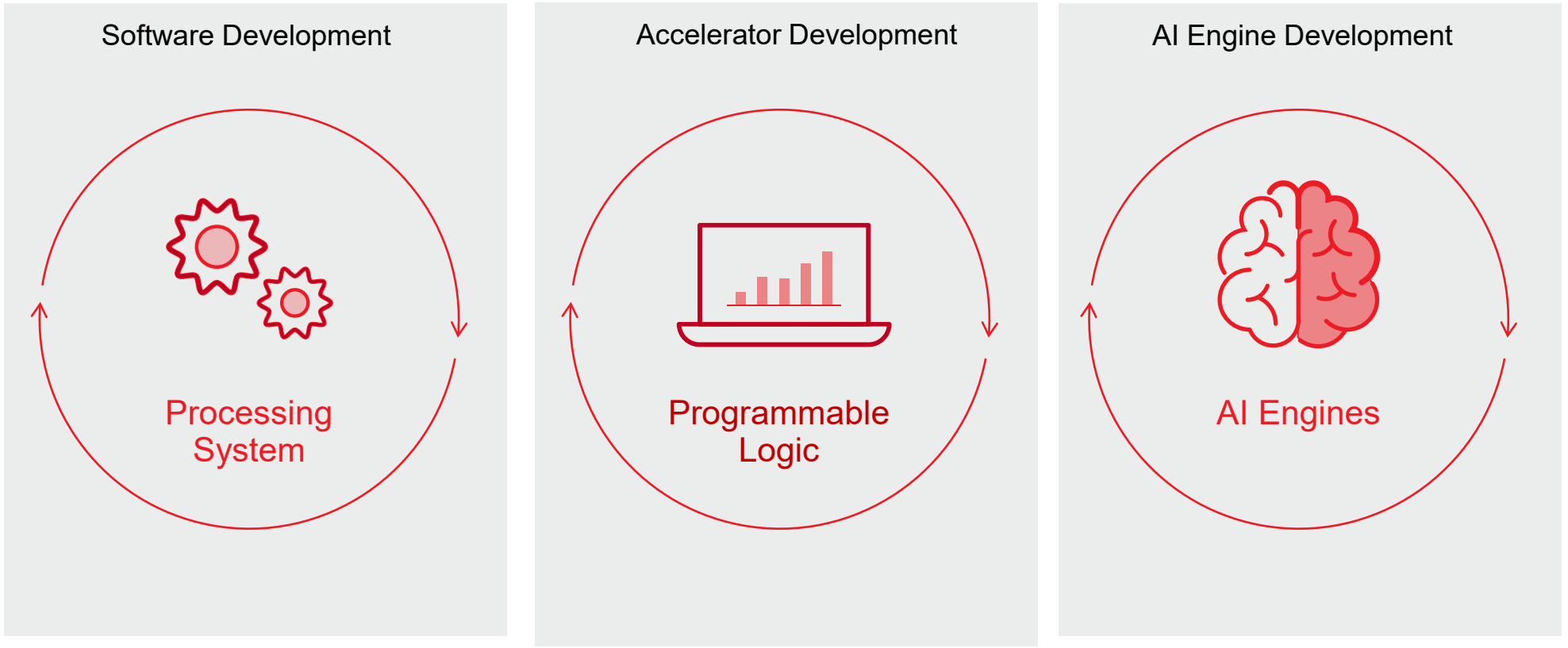


With Vitis™ Model Composer you can:

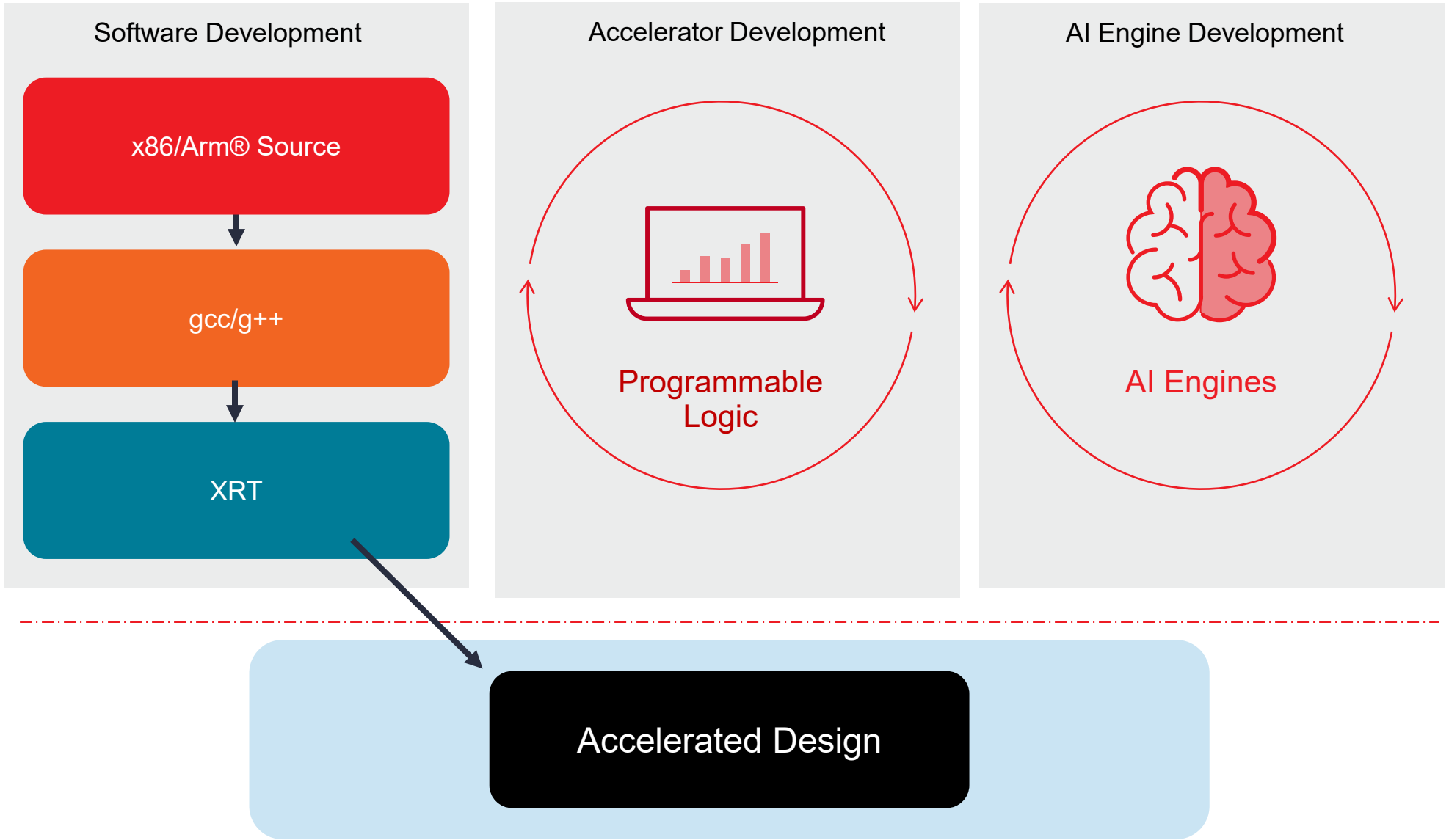
- Create a design using optimized blocks
- Visualize and analyze simulation results
- Seamlessly co-simulate AI Engine and programmable logic (HLS, HDL) blocks
- Automatically generate code (AI Engine data flow graphs, RTL, HLS C++) and test bench for a design
- Import custom HLS, AI Engine, and RTL code as blocks

Refer to *UG1483: Vitis Model Composer User Guide*

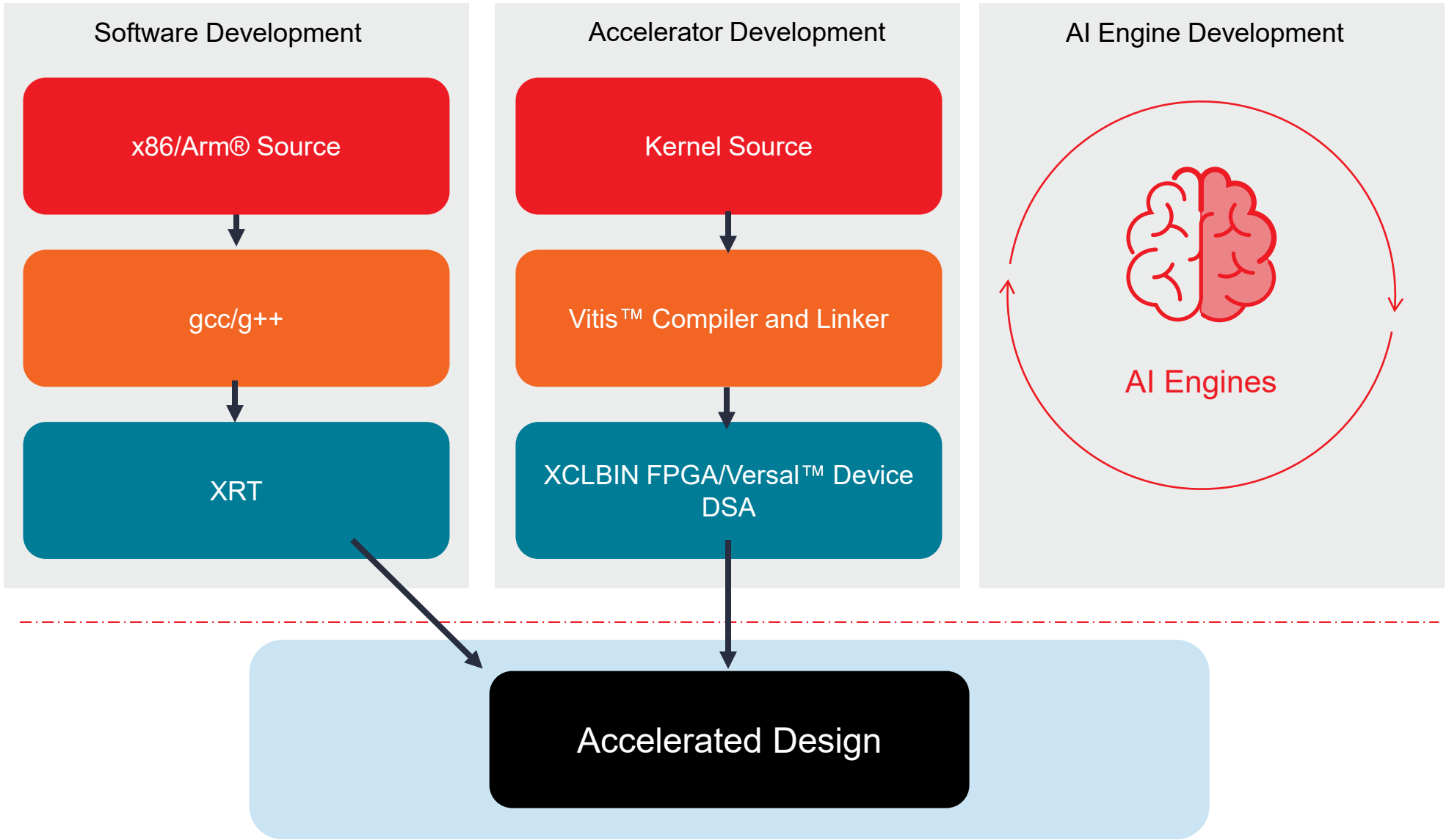
Vitis Environment: Full Application Acceleration Flow



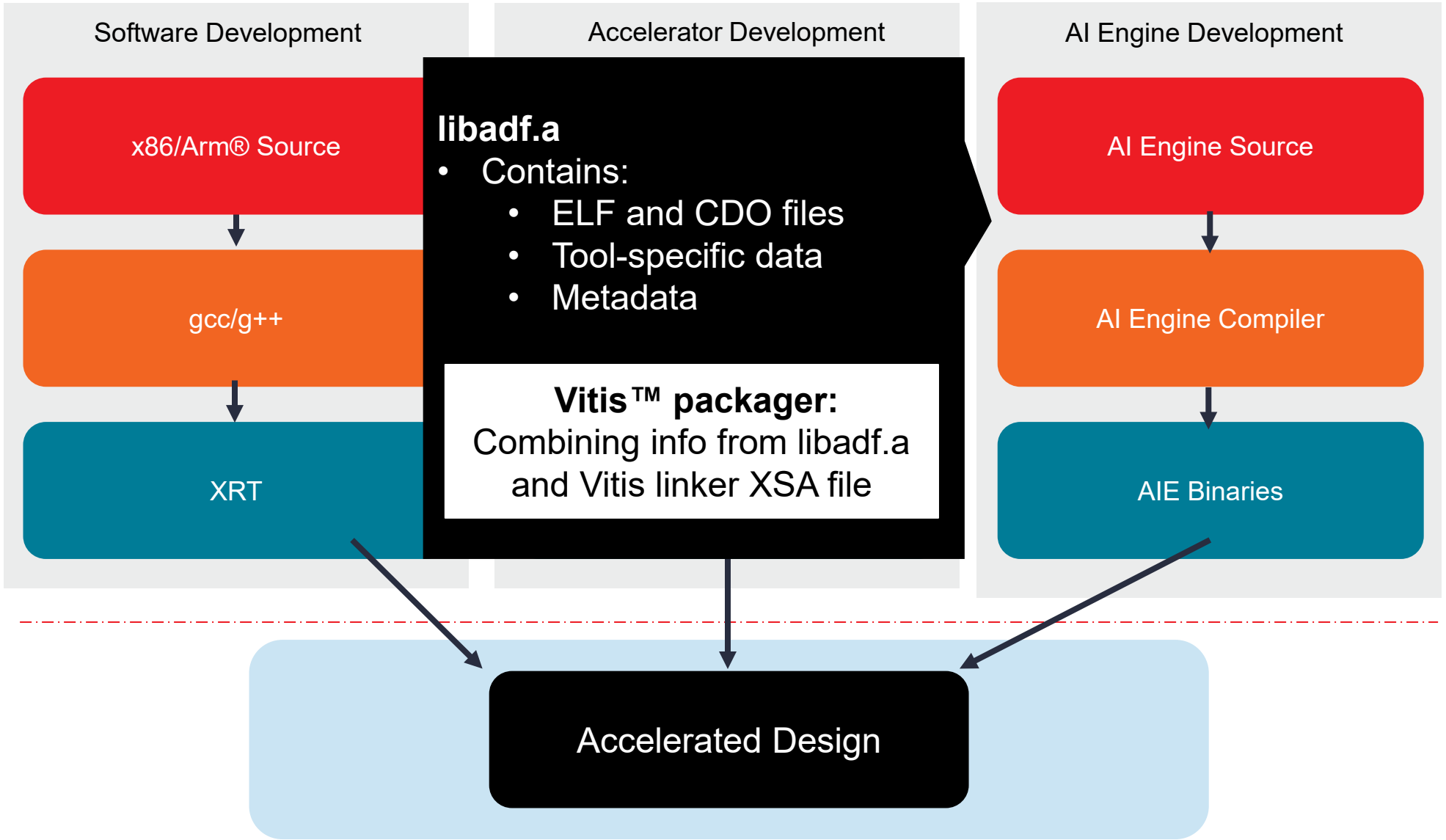
Vitis Environment: Full Application Acceleration Flow



Vitis Environment: Full Application Acceleration Flow



Vitis Environment: Full Application Acceleration Flow



Embedded Heterogeneous System Design Flow

2024.1

AMD

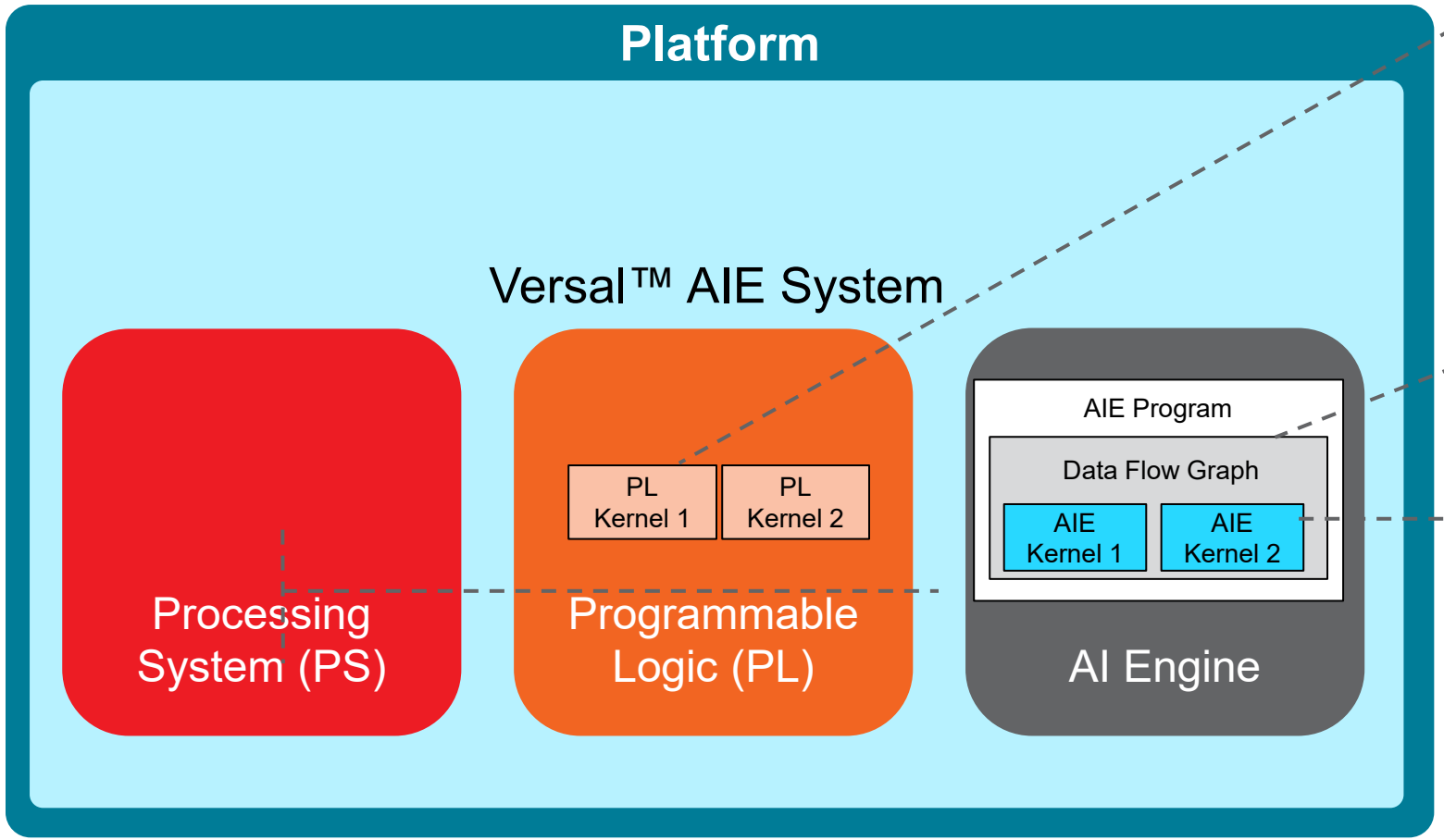
together we advance_

Terminologies

C/C++ {</>}

Kernel

C/C++ function that will be either compiled for an AI Engine or translated to RTL



C/C++ {</>}

PL Kernels

- Compilation units targeting PL kernels; referenced as nodes in the ADF graph or standalone IP
- Compiled into hardware using HLS

Data Flow Graph

- Written in C++
- Consists of nodes and edges

C/C++ {</>}

AIE Kernels

- C/C++ computation function
- Compiled for an AIE
- Compiled with aiecompiler and executed with aiesimulator

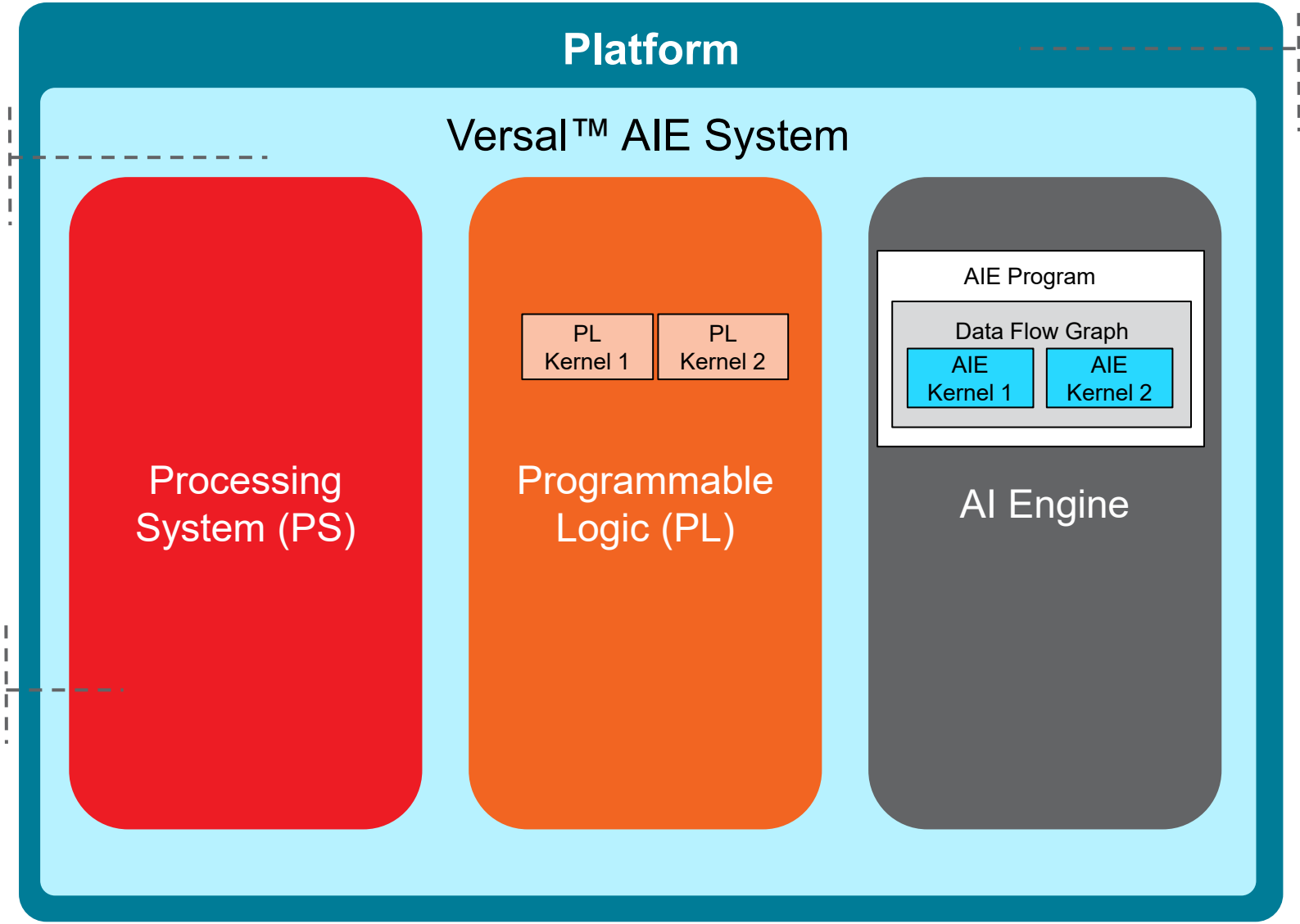
Terminologies

Test Bench

Top-level stimulus and response to the ADF graph representing the target system in simulation

Application code

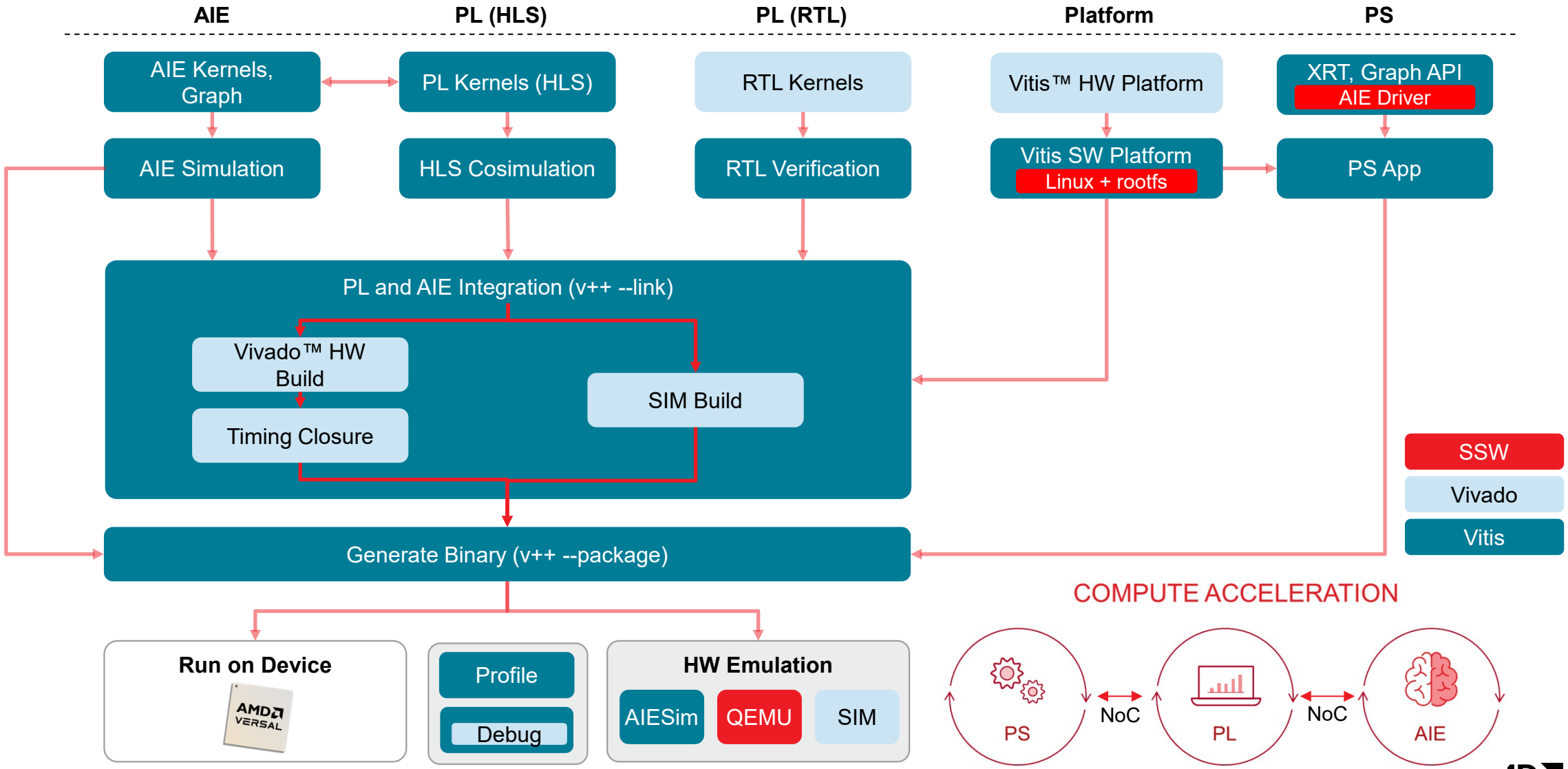
Application source targeting the APU/RPU or external CPU that controls the AIE subsystem



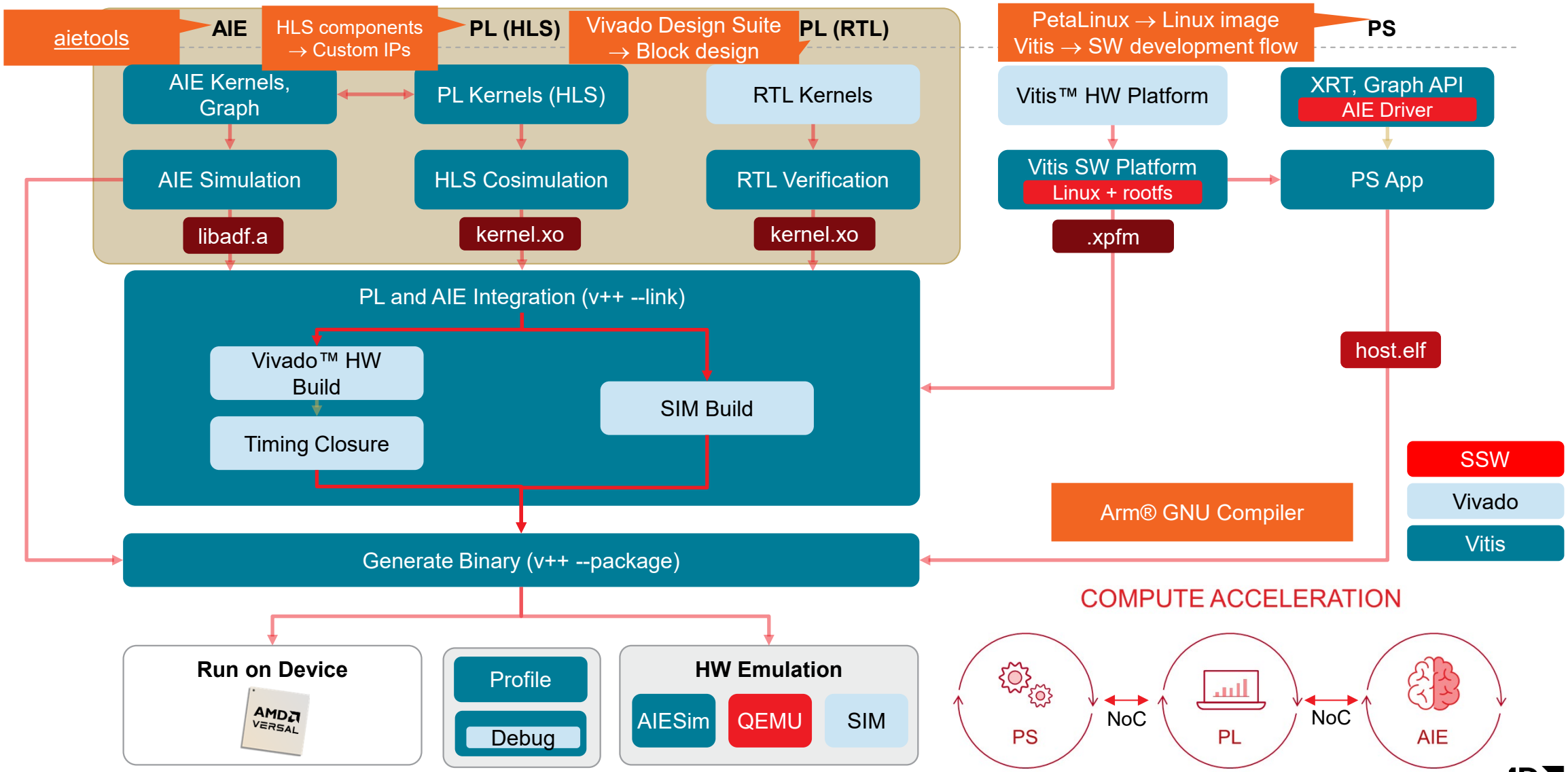
Platform description

Packaged hardware (.xsa) and software platform to support simulation and synthesis tool flows

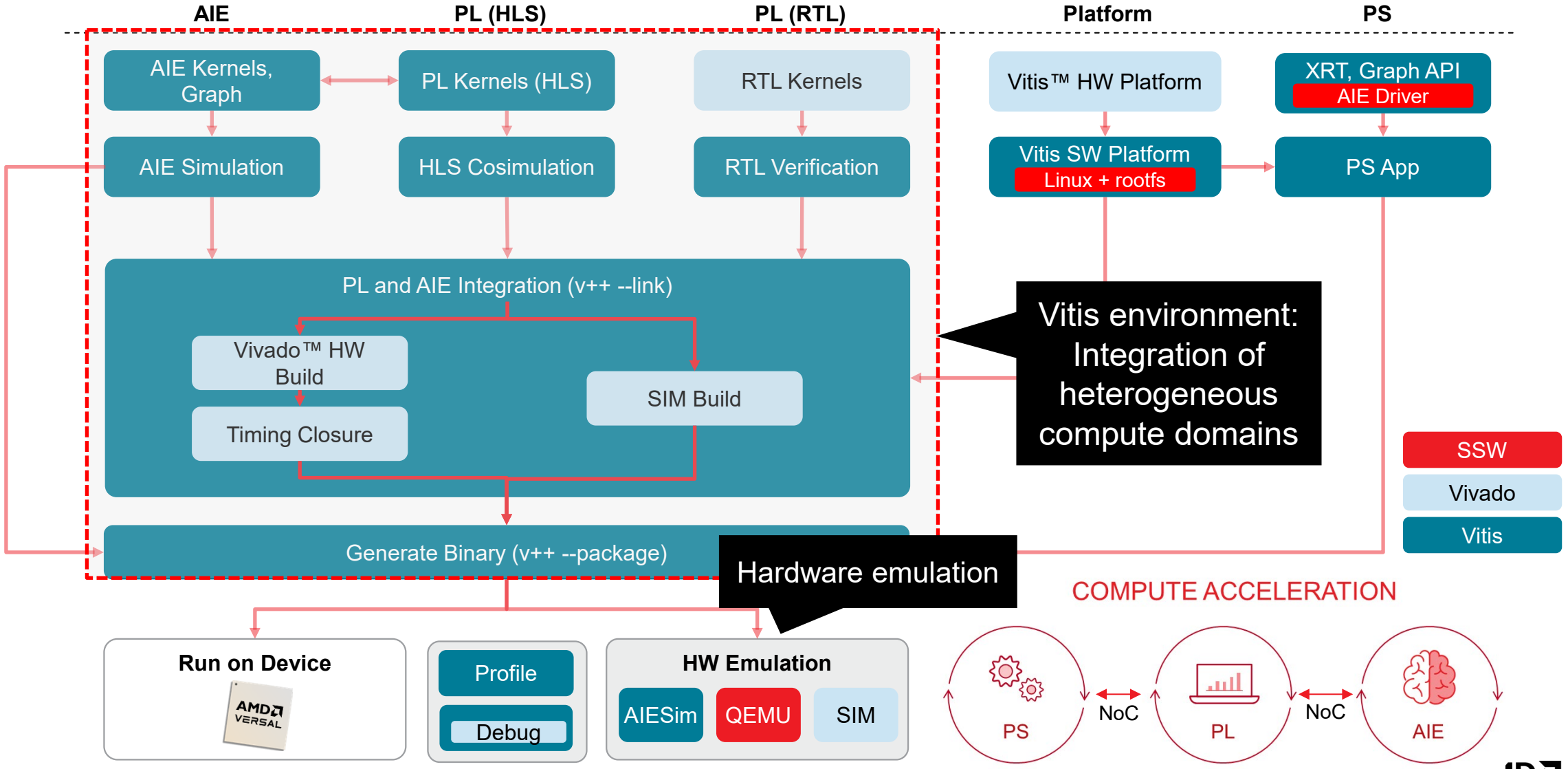
Vitis Tool Flow for Versal Devices



Vitis Tool Flow for Versal Devices



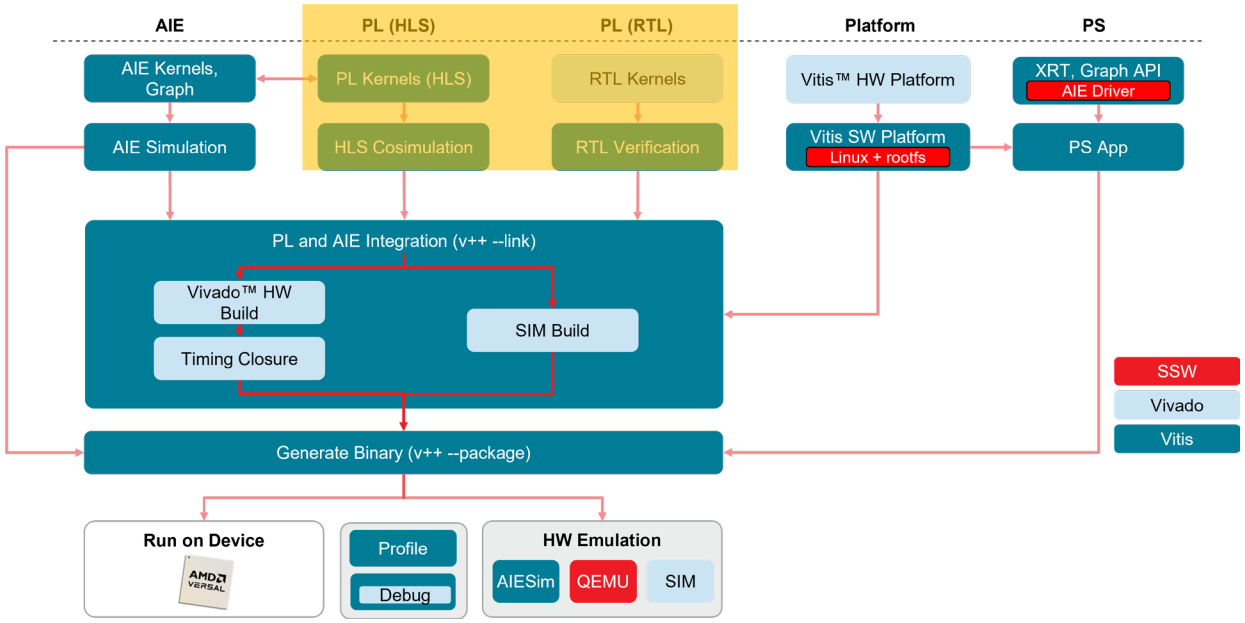
Vitis Tool Flow for Versal Devices

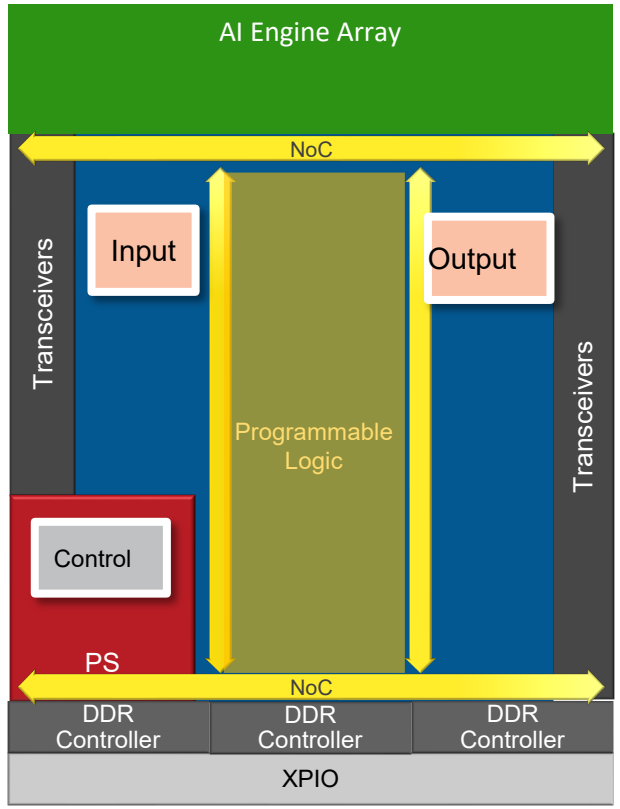


System Integration – AI Engine + PL Processing

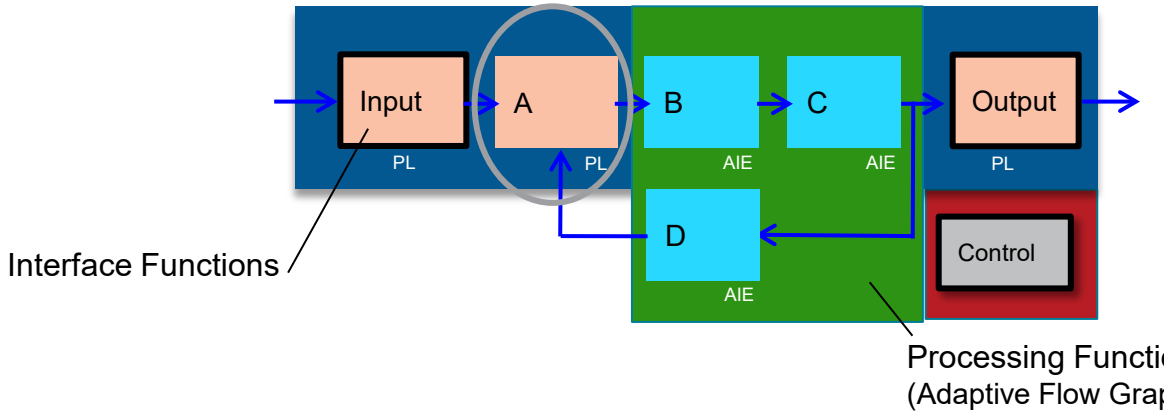
- Co-simulated using Vitis™ hardware emulation flow
- PL function: RTL IP or HLS C
- RTL function adapter
  - C/C++ function signature
  - Interface wrapper

Communication between the PL function and the ADF graph requires PLIO interfaces

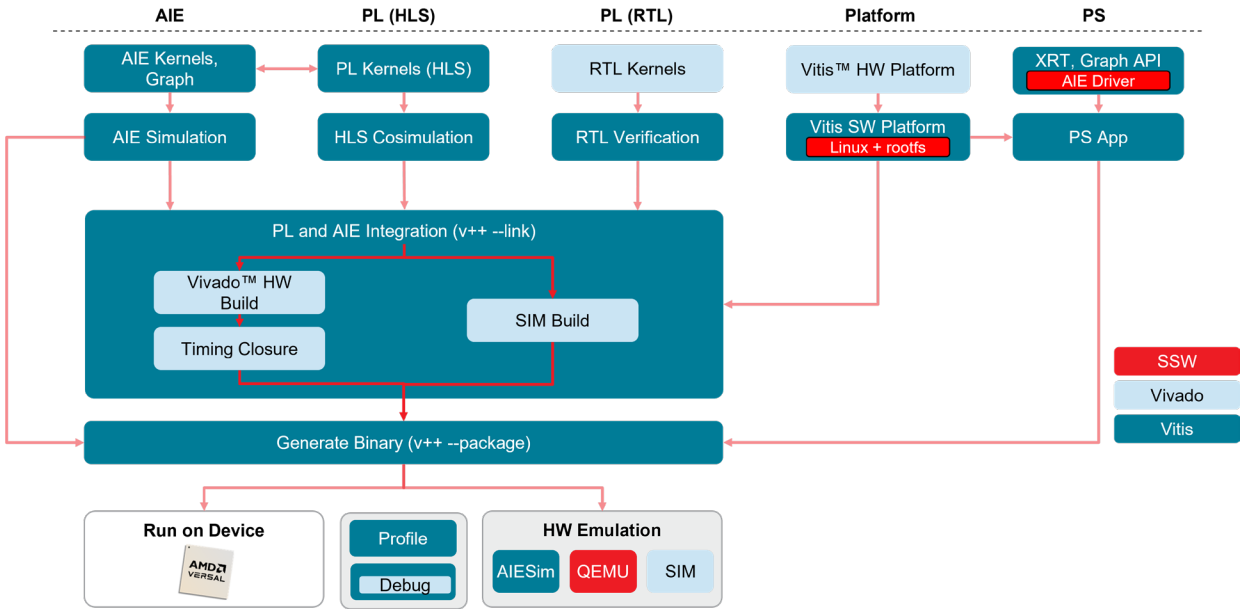


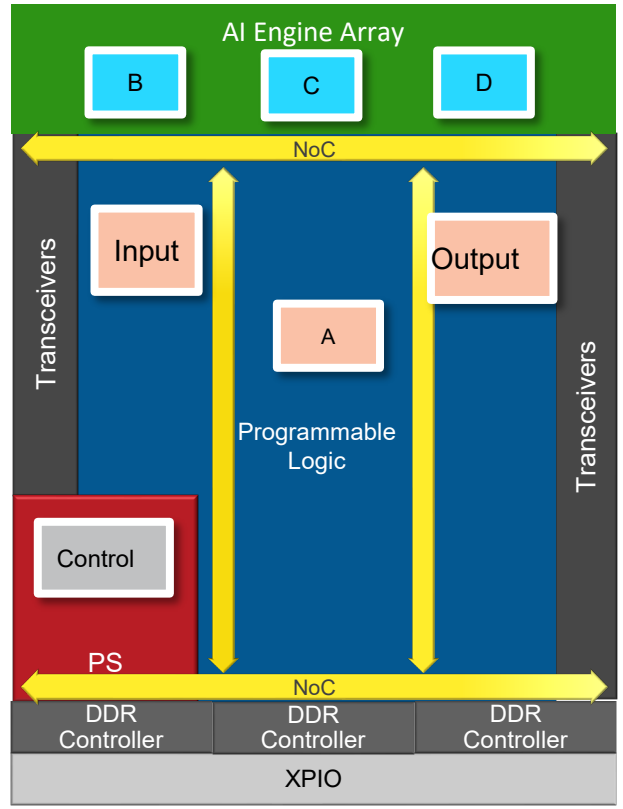


System Integration – AI Engine + PL Processing



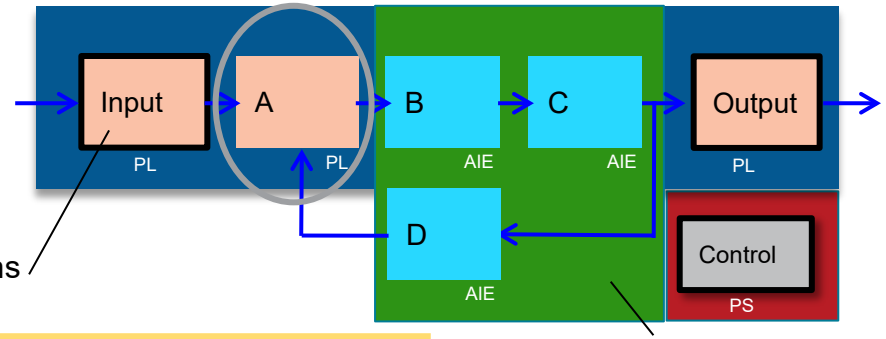
Automated DMA insertion and clock domain crossing





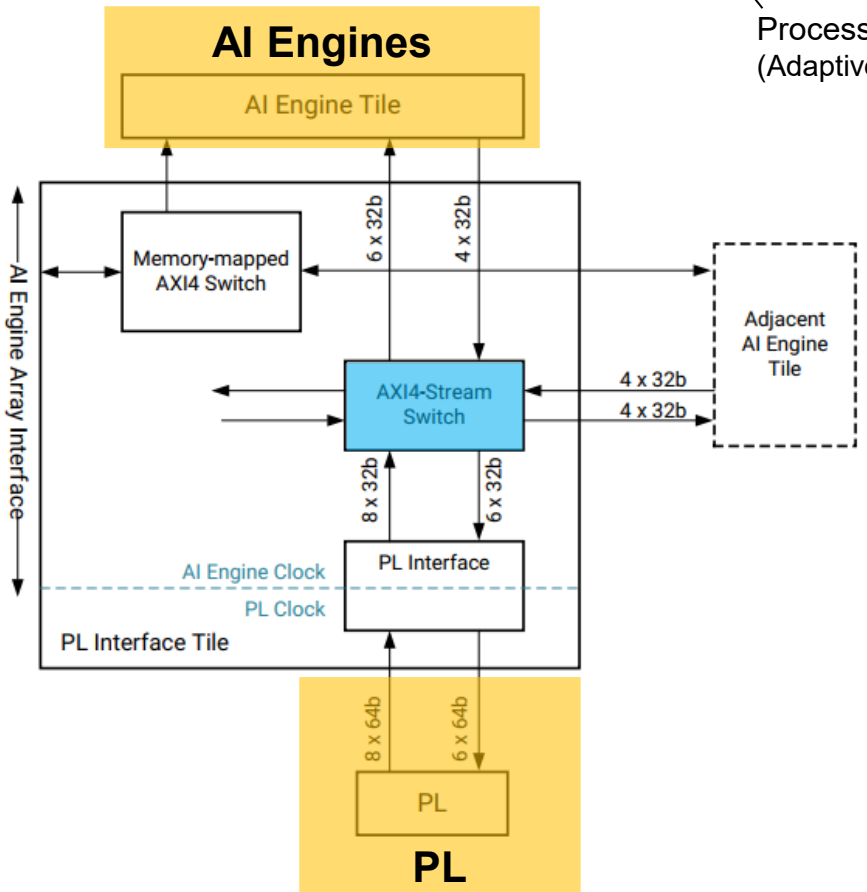
AMD
together we advance_

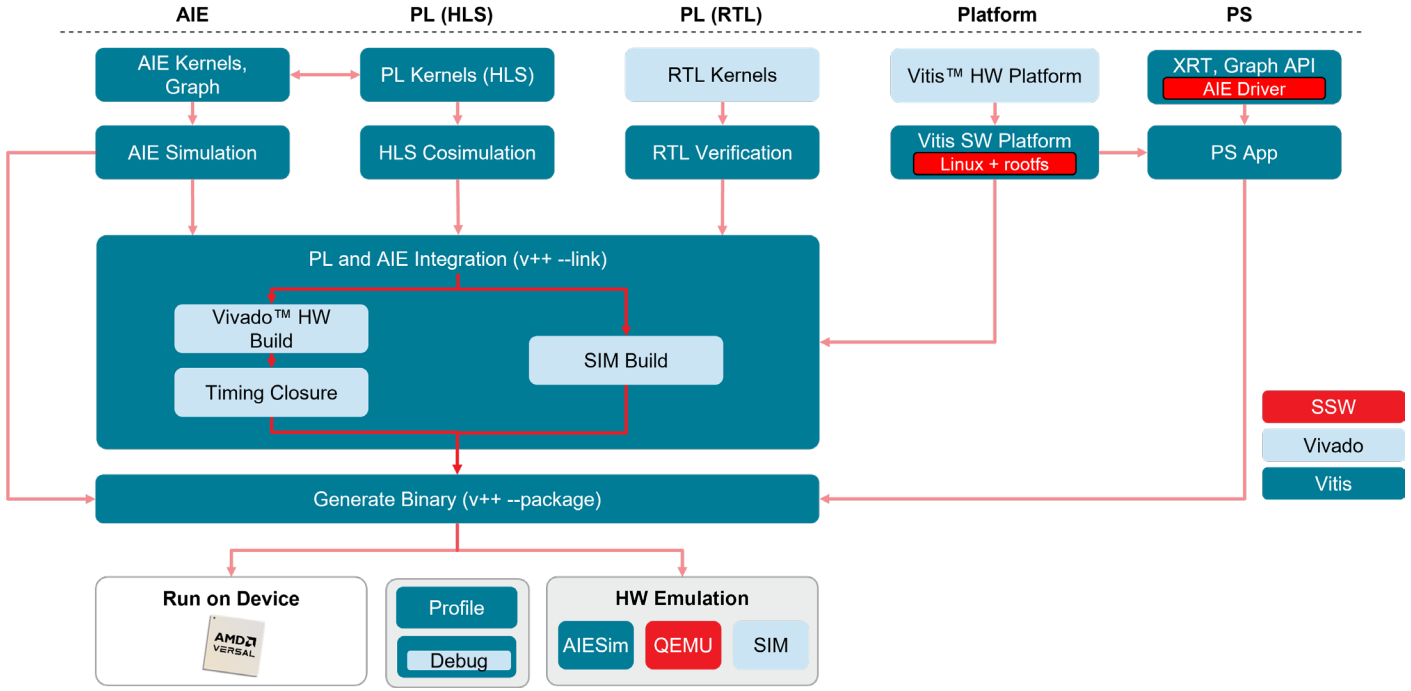
System Integration – AI Engine + PL Processing



Interface Functions

Processing Functions
(Adaptive Flow Graph)





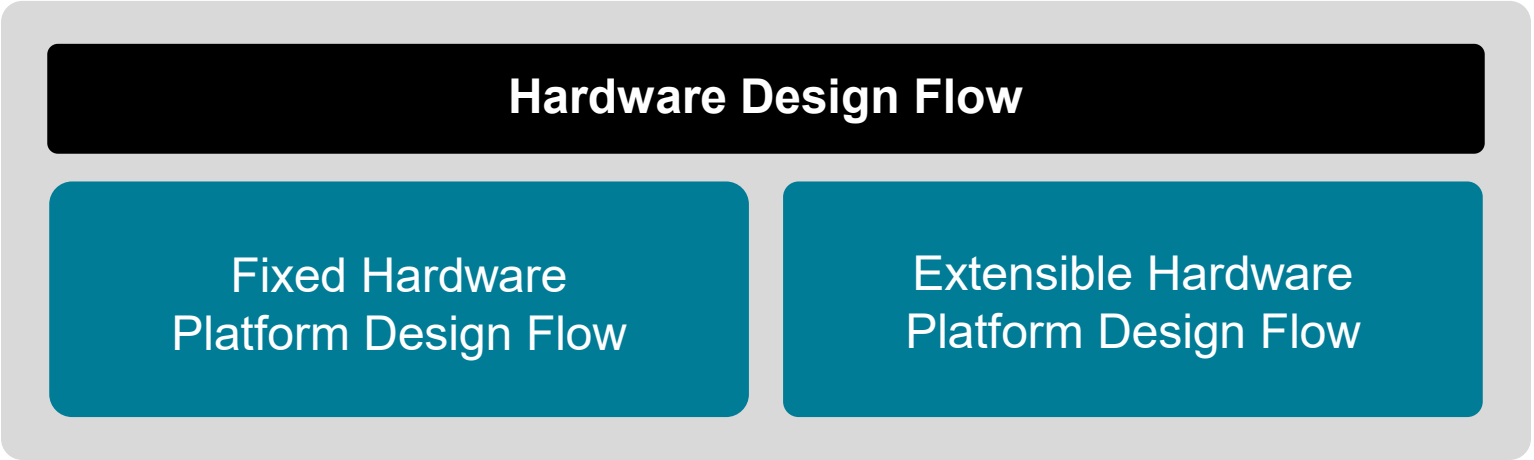
Platform Types

For RTL and IP only:

- Vivado™ Design Suite can be used to generate a PDI
- Design sources are added and compiled through Vivado implementation flow

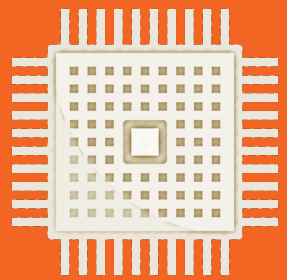
Archived Vivado Design Suite projects can be redesigned and packaged for reuse:

- Fixed firmware device
- Base hardware for additional hardware accelerators



Fixed Hardware Platform Design Flow

Embedded Software Design Flow



Hardware Creation

Hardware Team: Creates the hardware platform

Software Team: Develops the embedded software application

Application Development

Hardware design – Vivado™ Design Suite – IP block design/RTL

Hardware C++ function – Vitis™ HLS

.xpfm – Used to create embedded applications

Extensible Hardware Platform Design Flow

Extensible Platform

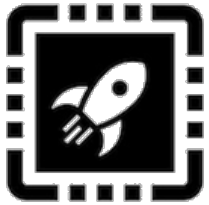
- Defines an extensible hardware design (.xsa)
  - Allows the addition of programmable components
- Versal™ adaptive SoC IP blocks
  - CIPS, NoC, and AI Engine
- Board interface IP blocks
  - High-speed I/Os and memory controllers

Programmable Components

Composed of both:

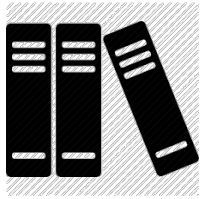
- PL kernels
- AI Engine blocks

Supports application acceleration



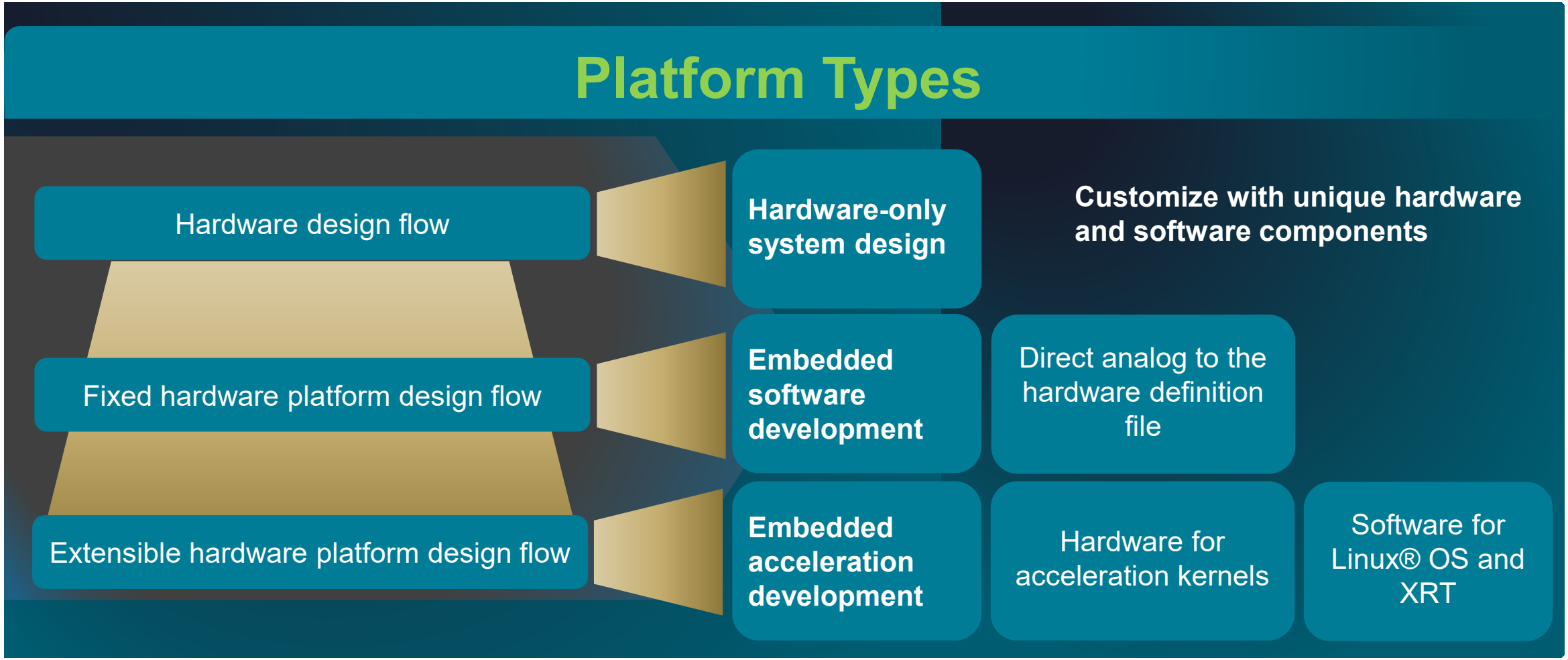
Hardware for accelerating kernels

Software targeting Linux® OS



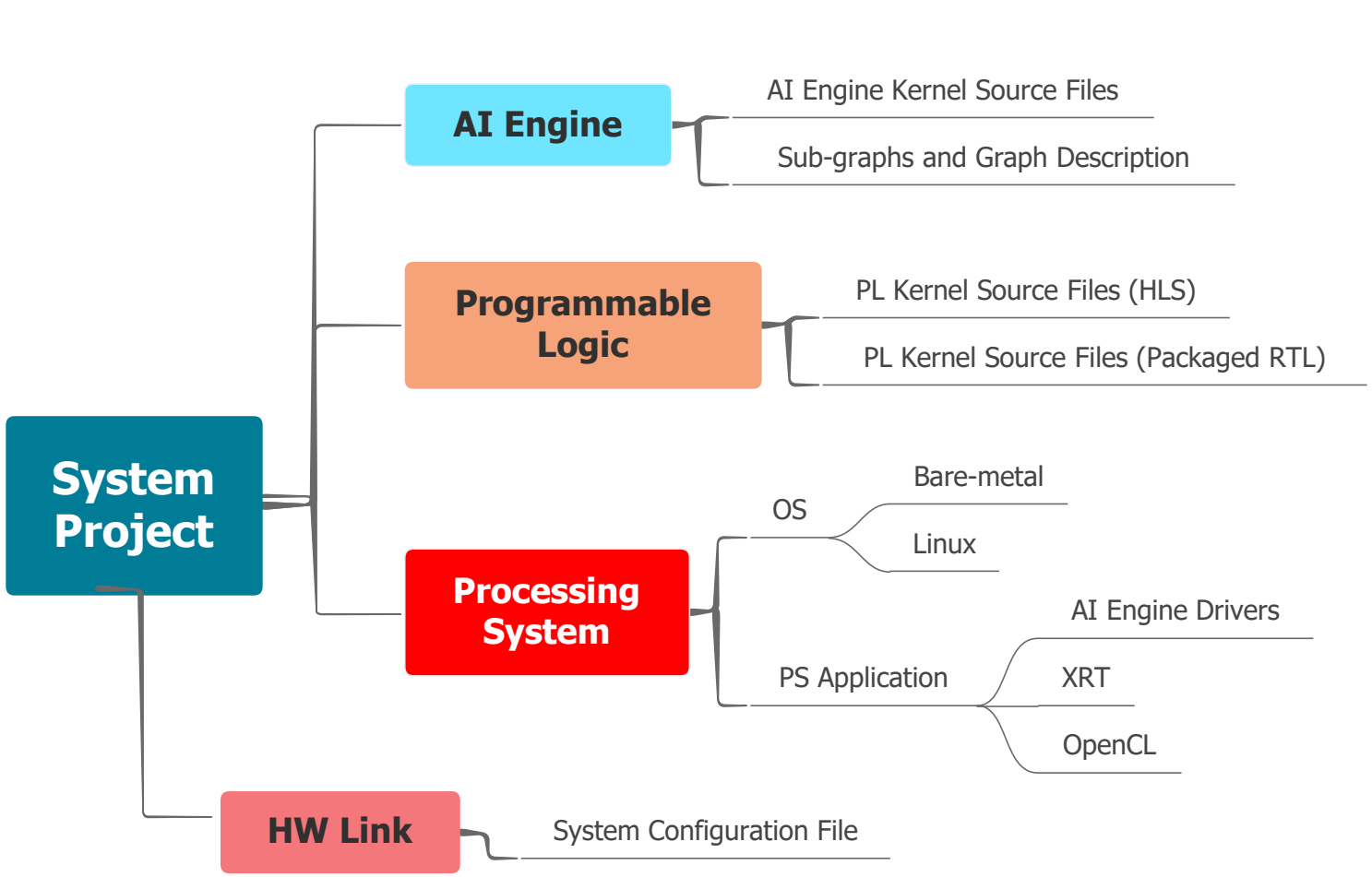
Xilinx Runtime Library (XRT)

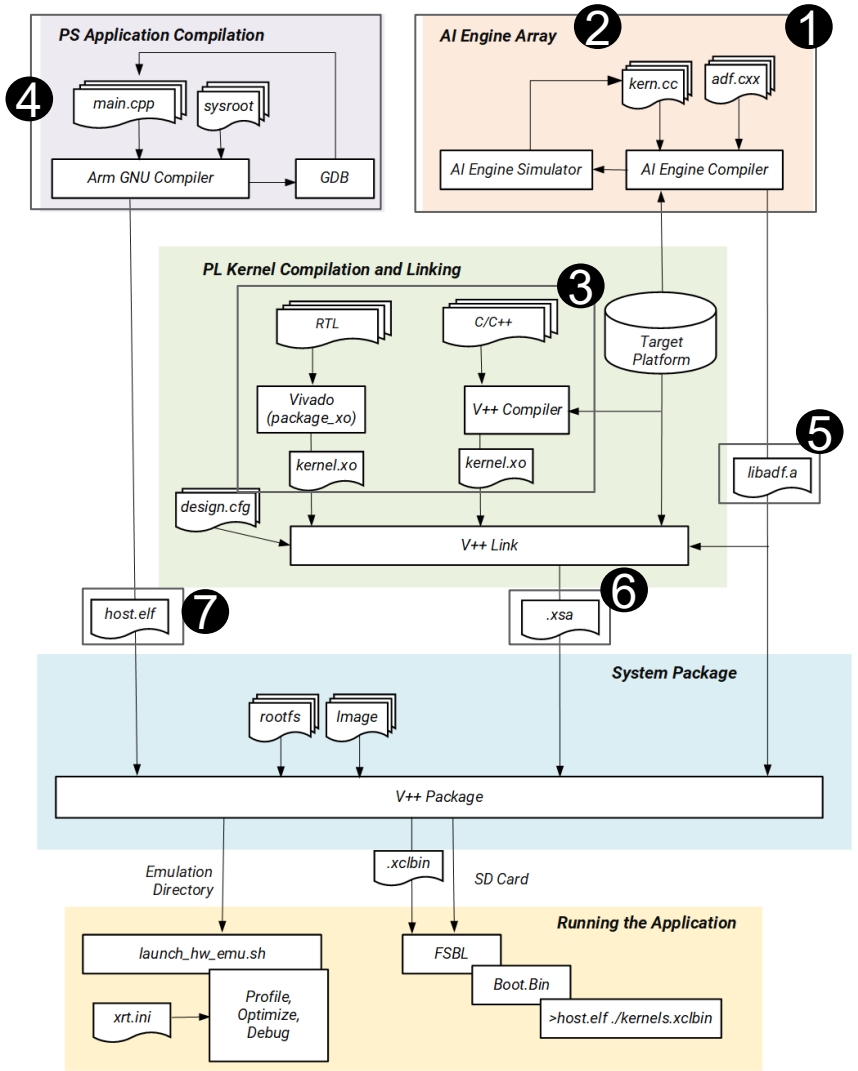
Platform Types – Summary



For more information on the XRT library, go to github.com/Xilinx/XRT

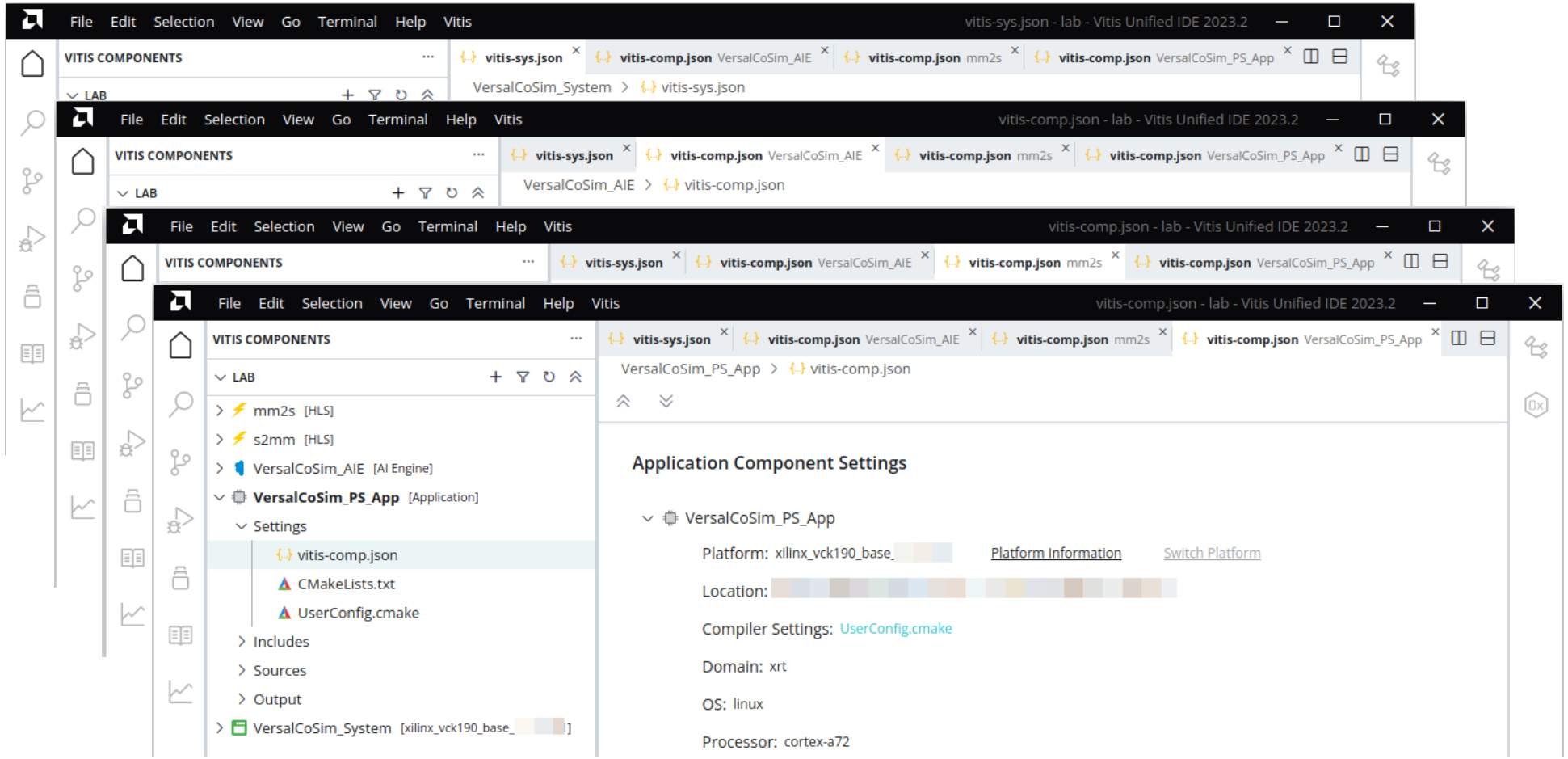
Project Structure in the Vitis Environment





Components in the Vitis Environment

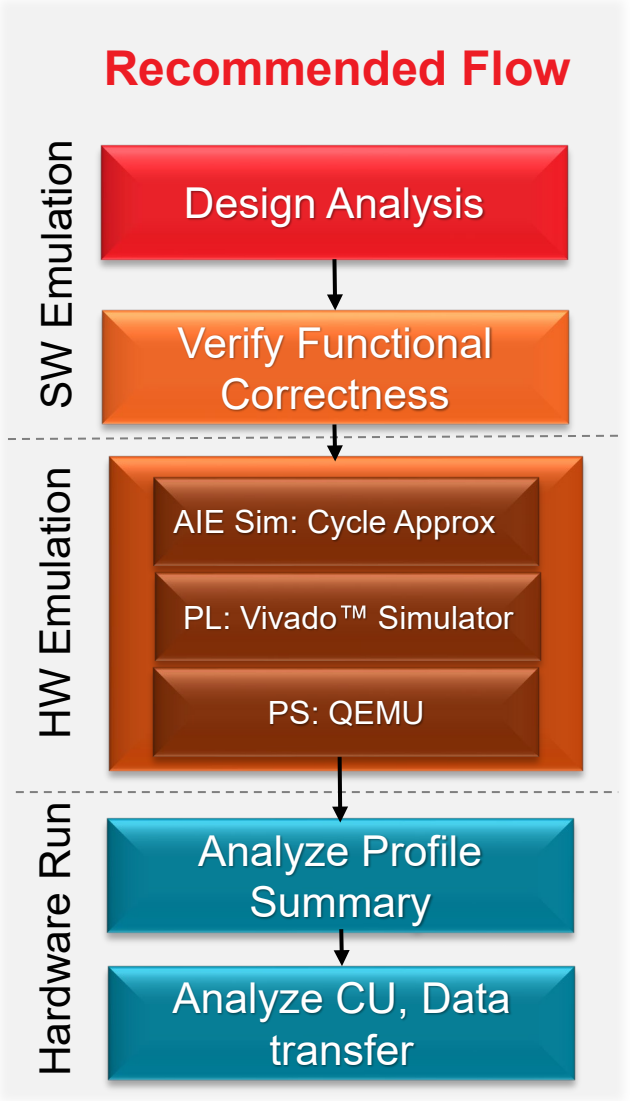
System Component | AI Engine Component | HLS Component | Application Component



Vitis Tool Emulation and Implementation Flows – Versal Device

Executing the application with the actual FPGA and AI Engine ELF is the end goal

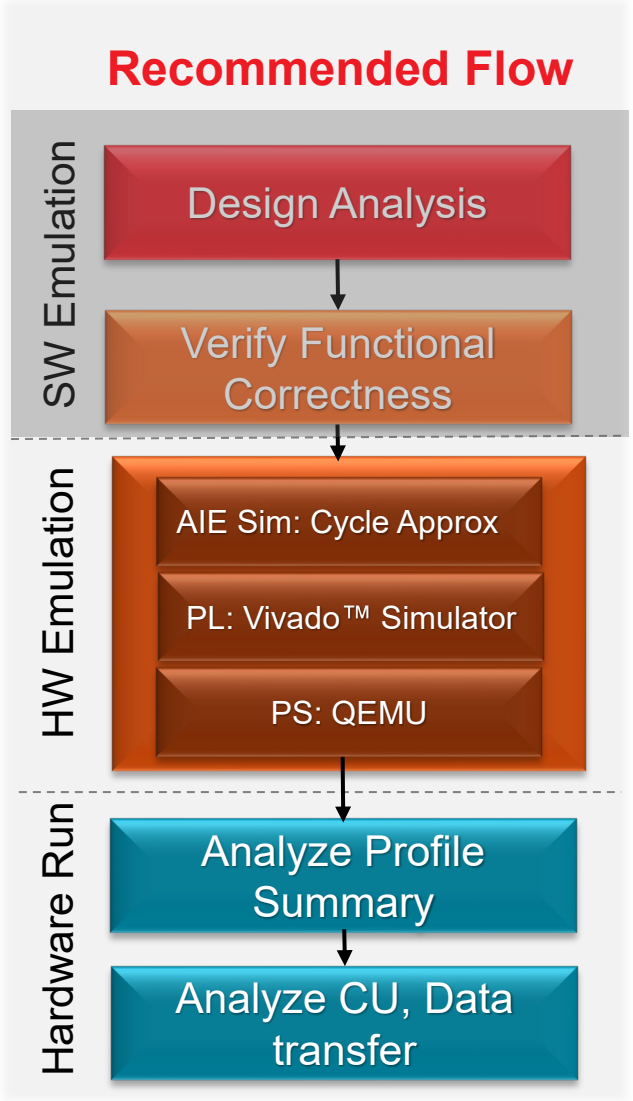
- SW Emulation
- HW Emulation
- Hardware Run



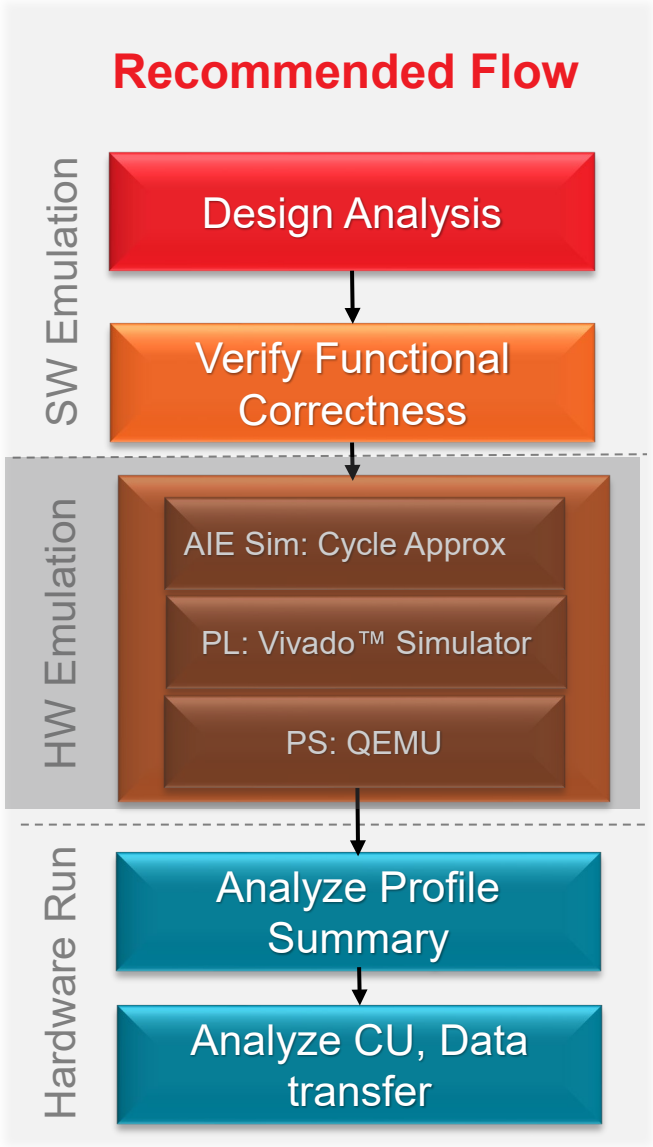
Vitis Tool Emulation and Implementation Flows – Versal Device

SW Emulation

- AI Engine instruction set SystemC simulation: bit exact
- HLS PL kernels are simulated using the C code
- PS code runs on the development platform

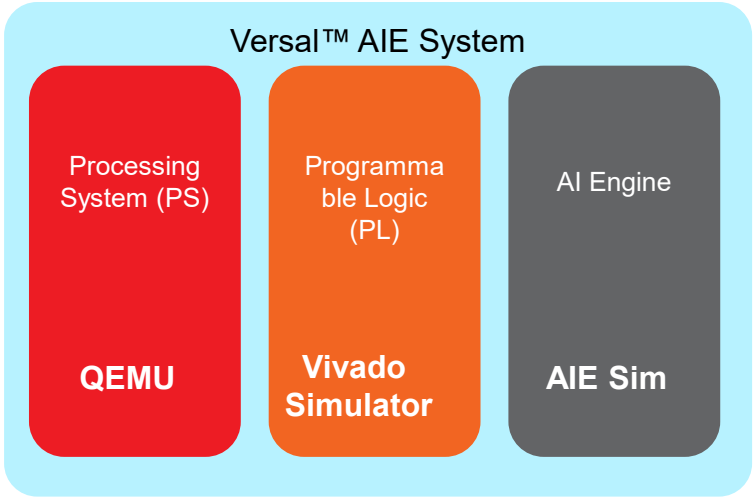


Vitis Tool Emulation and Implementation Flows – Versal Device



- AI Engine instruction set SystemC simulation: Bit exact and cycle approximate
- Vivado simulator used for the PL partition (compiled HLS kernels and RTL kernels)

Simulates

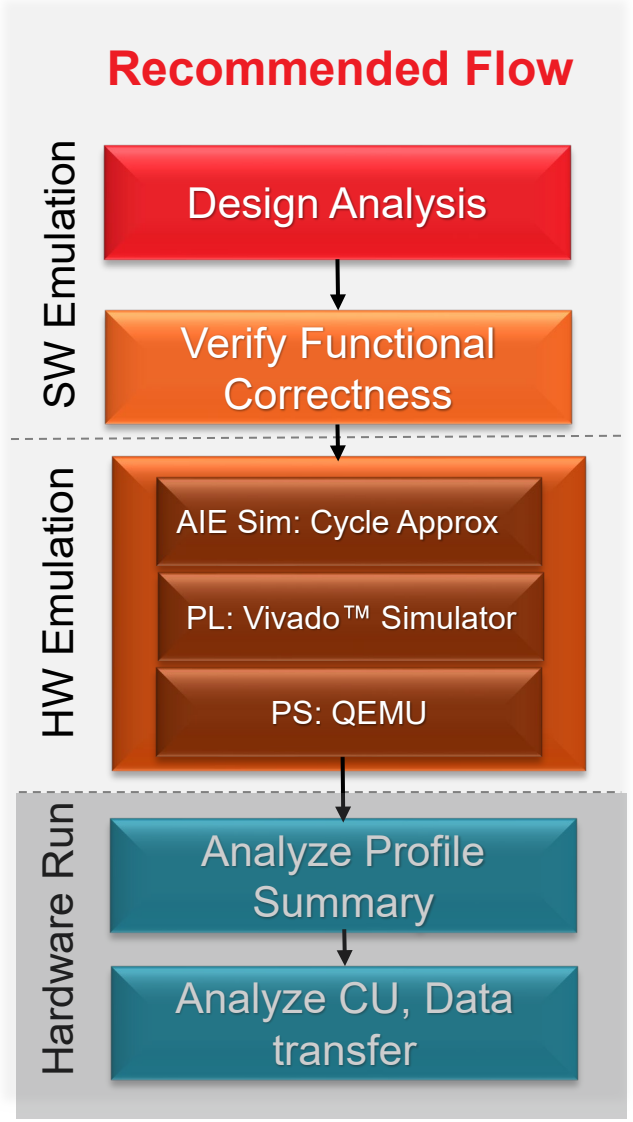


- QEMU used to simulate the processing system
- Full debug visibility into all aspects of the application
- Performance bottlenecks

Vitis Tool Emulation and Implementation Flows – Versal Device

Hardware Run

- Generation of the complete system (AI Engine, bitstream, PS application)
  - Programmable device image (PDI) to program and configure the Versal™ device
- PDI consists of headers, the platform loader and manager image, and design data image partitions
- PDI also contains configuration data, ELF files, and NoC register settings
- PDI image is programmed through the PMC block by the bootROM and PLM



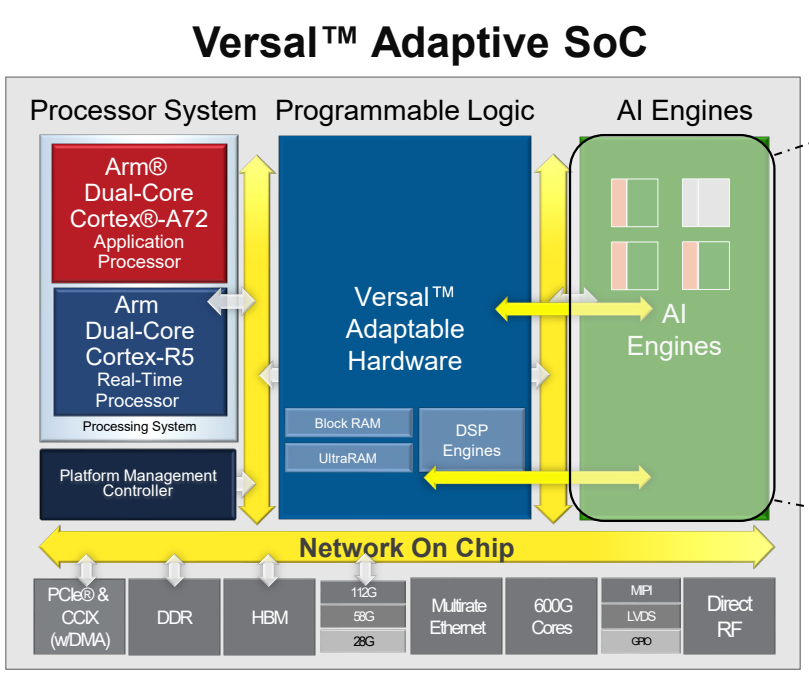
Vitis Tool Simulation and Implementation Flows – AI Engine

x86 Simulation

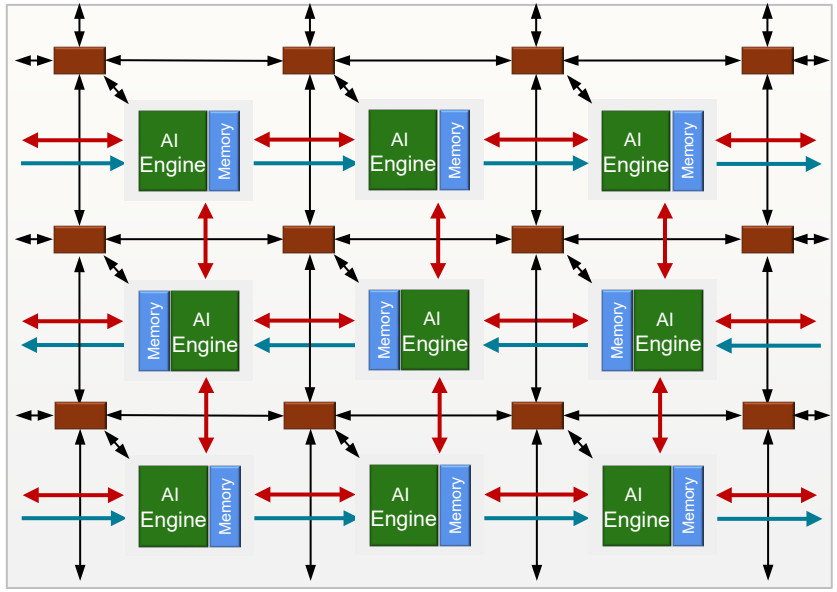
- Pure functional simulation
- AI Engine instructions
  - Simulated on the host
  - Compiled with gcc

AIE Simulation / Hardware

- AI Engine simulation: bit exact and cycle true
- AI Engine compiler
  - Generates the AI Engine code
- Compiled by the AI Engine compiler for use in the actual device

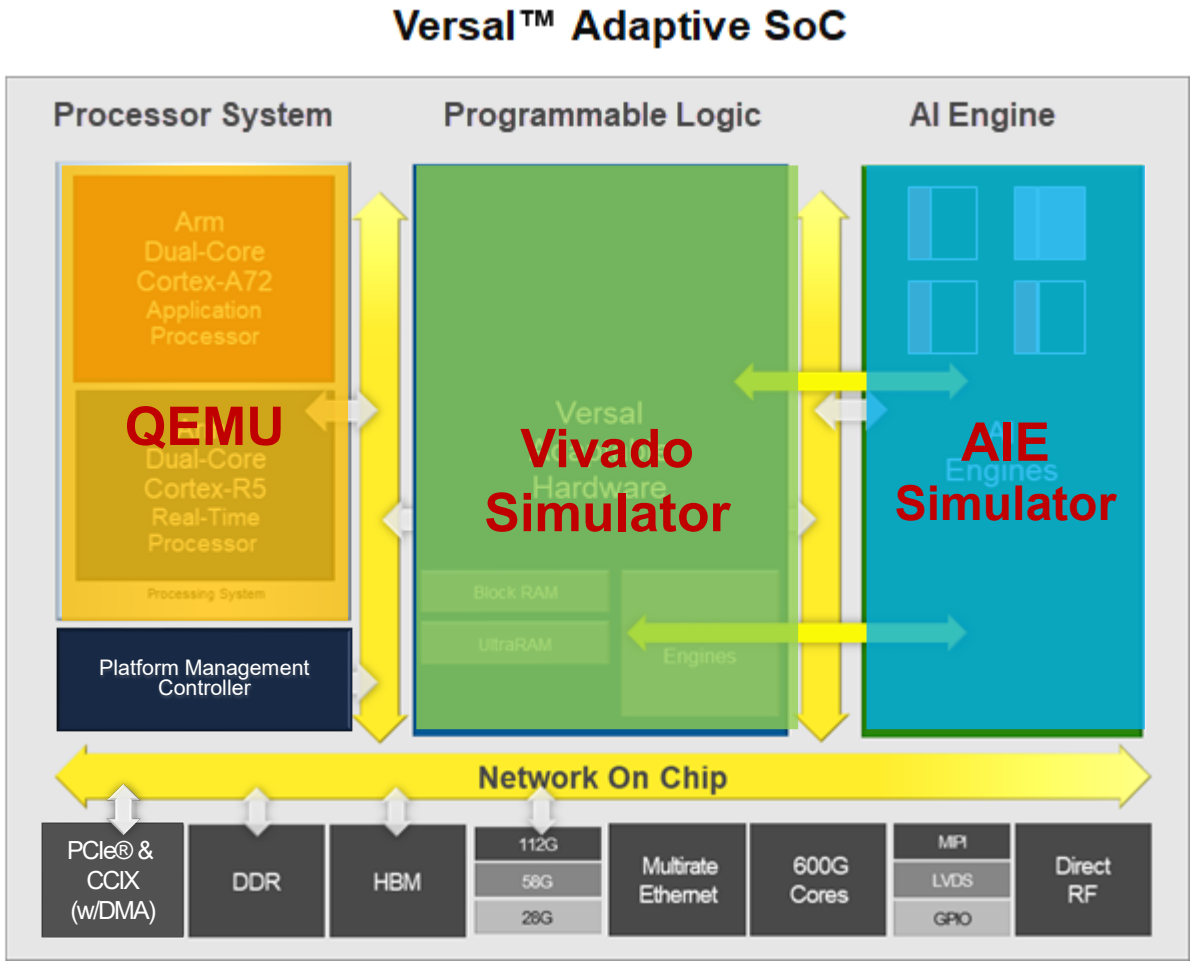


AI Engine Array



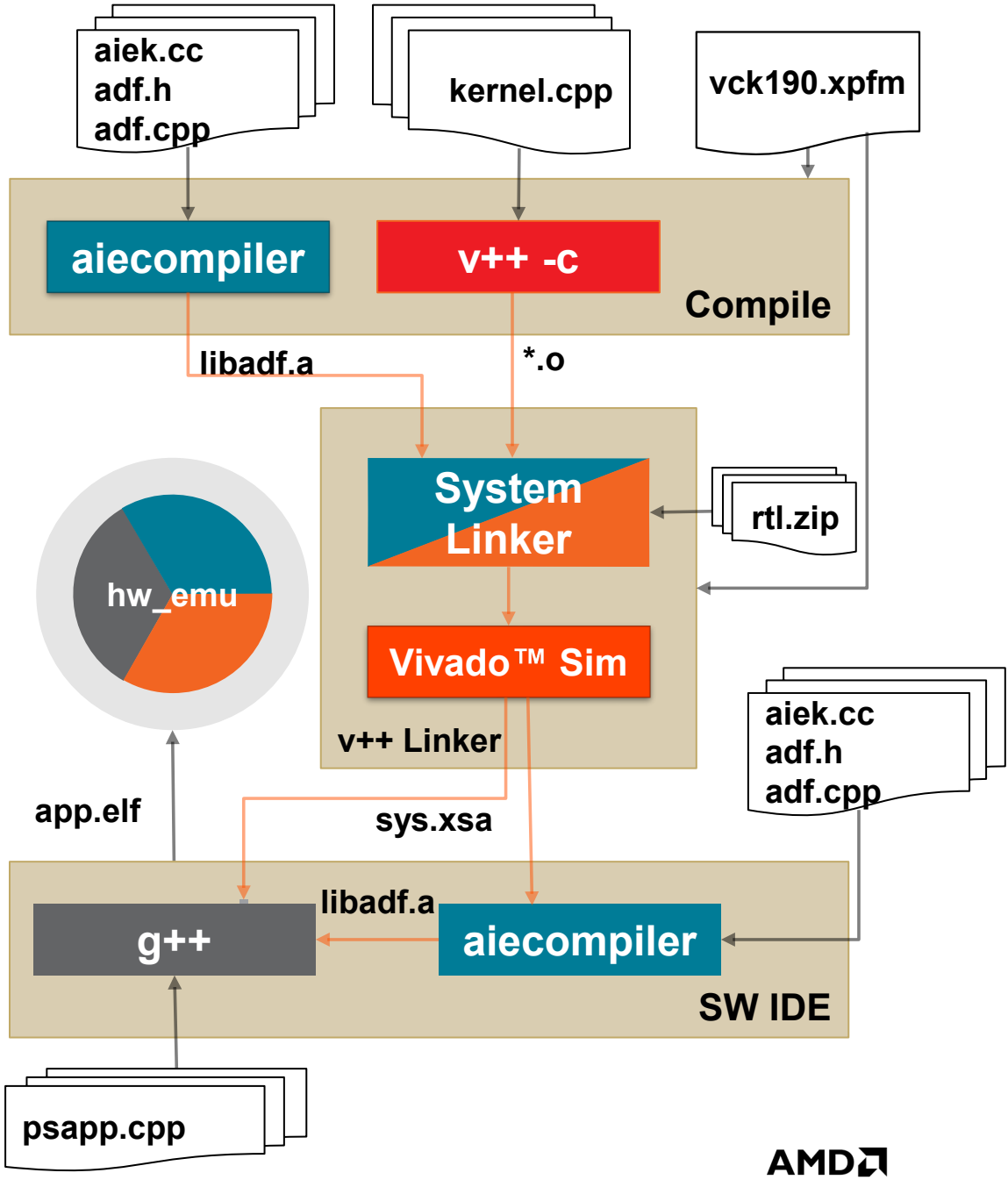
Vitis Tool Emulation-HW – Complete System

- Complete Vitis™ tool hardware emulation for heterogeneous simulation
- QEMU models Arm® Cortex®-A72 and Cortex-R5 PMC cores to run the same cross-compiled binaries as actual hardware
- AIE modeled through a SystemC simulator that runs the same cross-compiled AIE kernels
- PL is modeled through RTL simulation (Vivado™ simulator)



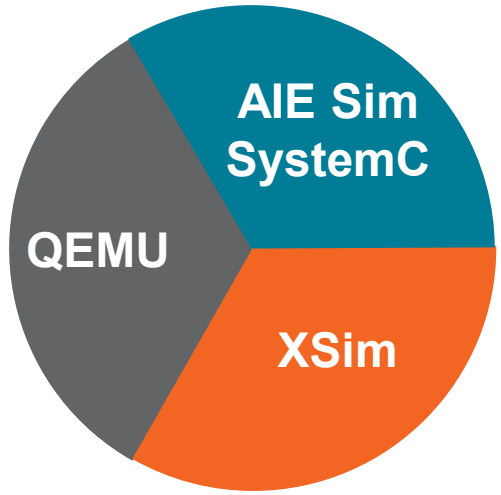
Vitis Tool Hardware Emulation (1)

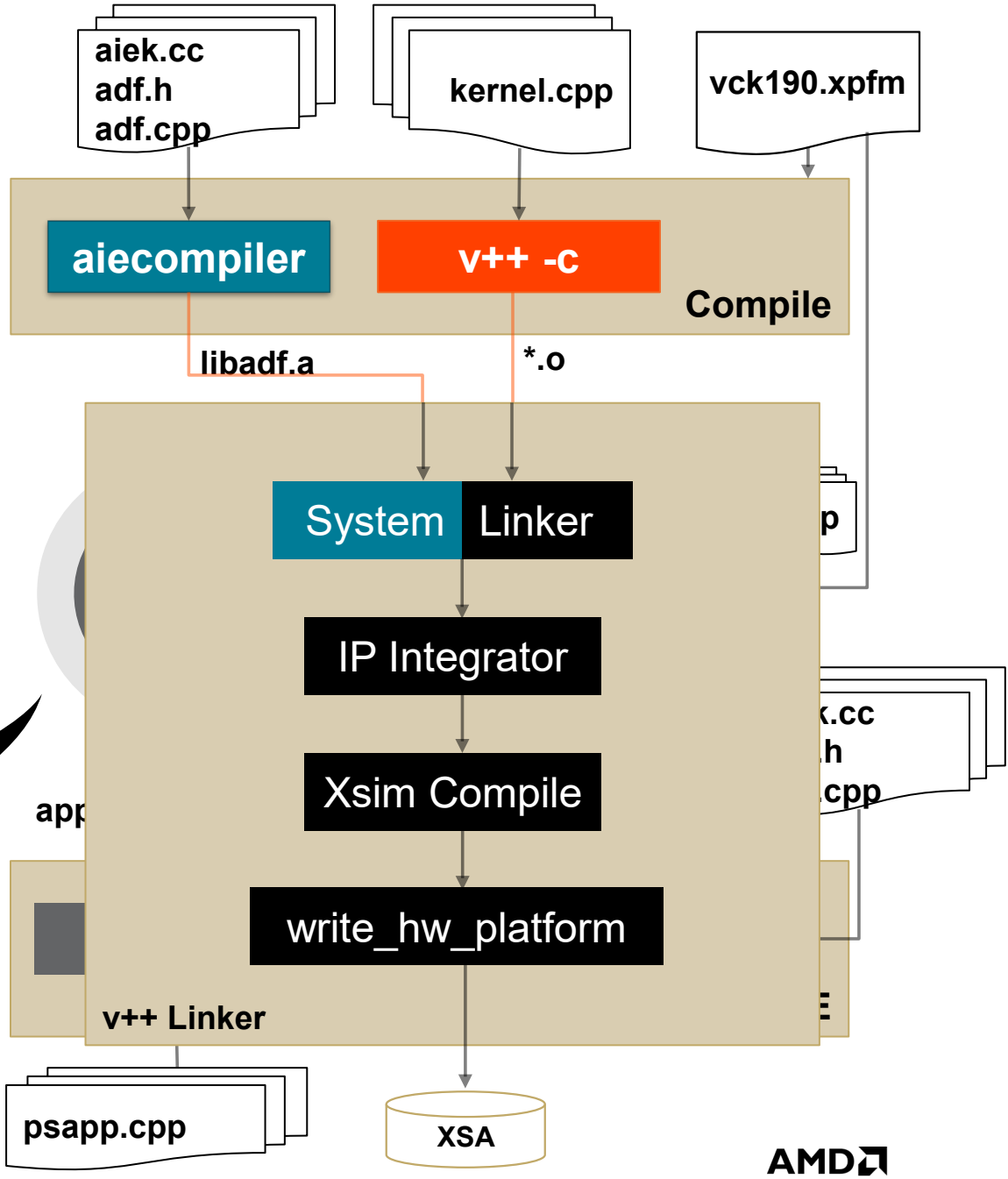
- Hardware emulation simulates a complete Versal™ adaptive SoC system composed of the AI Engine, PS, and PL



Vitis Tool Hardware Emulation (2)

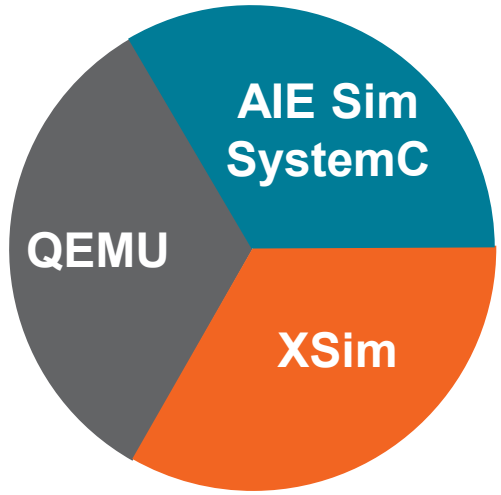
- QEMU/RTL co-simulation
  - SystemC models for packaged IPs
- AIE kernels and graph compiled through the Vitis™ tool
  - Bottom-up and top-down compilation
- Hardware/software emulation model of PS, AIE, and accelerator networks generated by Vitis compiler

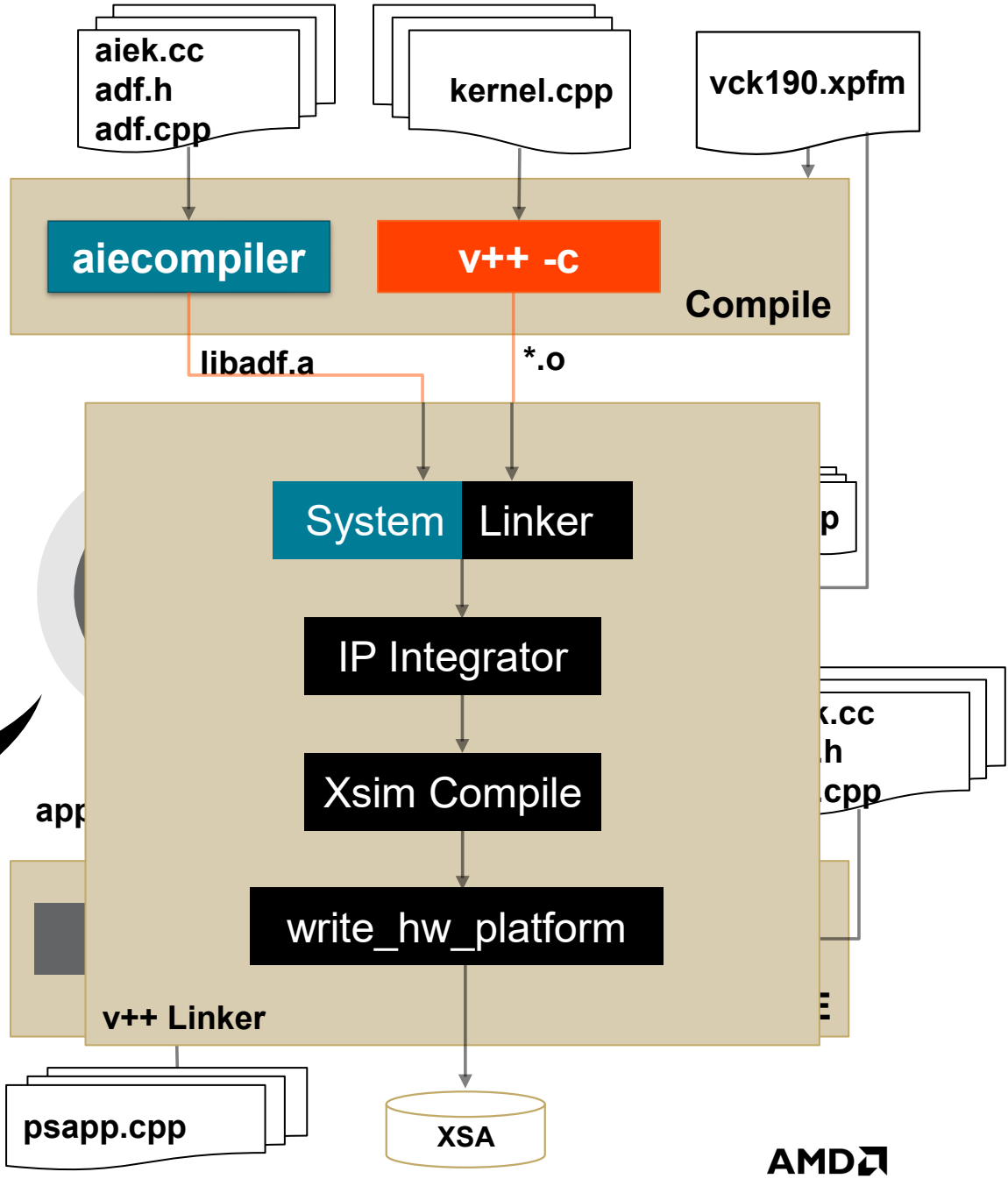




Vitis Tool Hardware Emulation (3)

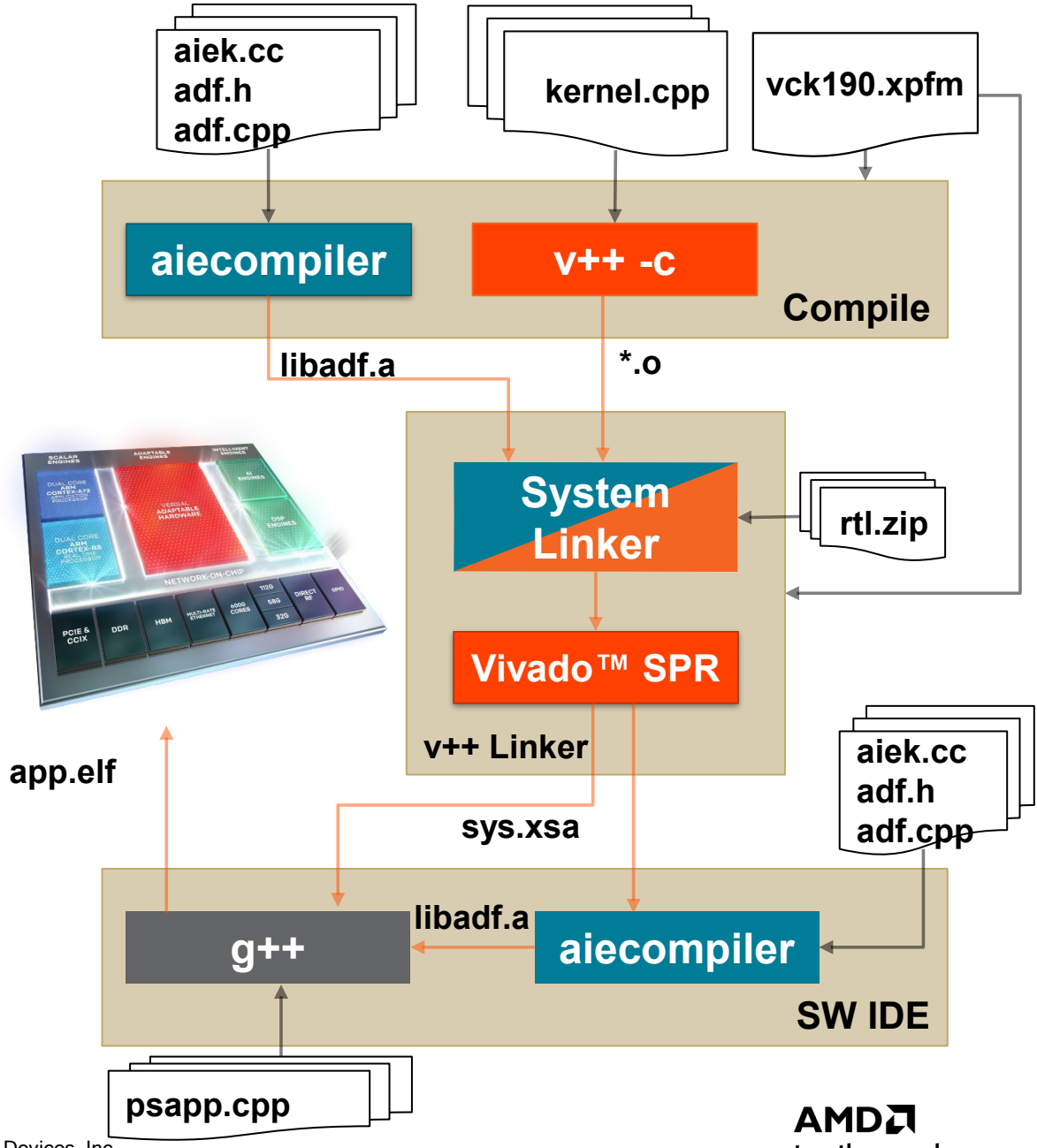
- Abstraction is very close to but not fully cycle accurate
- QEMU:
  - Generic and open-source machine emulator
  - Provides the ability to execute CPU instructions at almost real time without the need for real hardware





Vitis Tool Hardware Build (1)

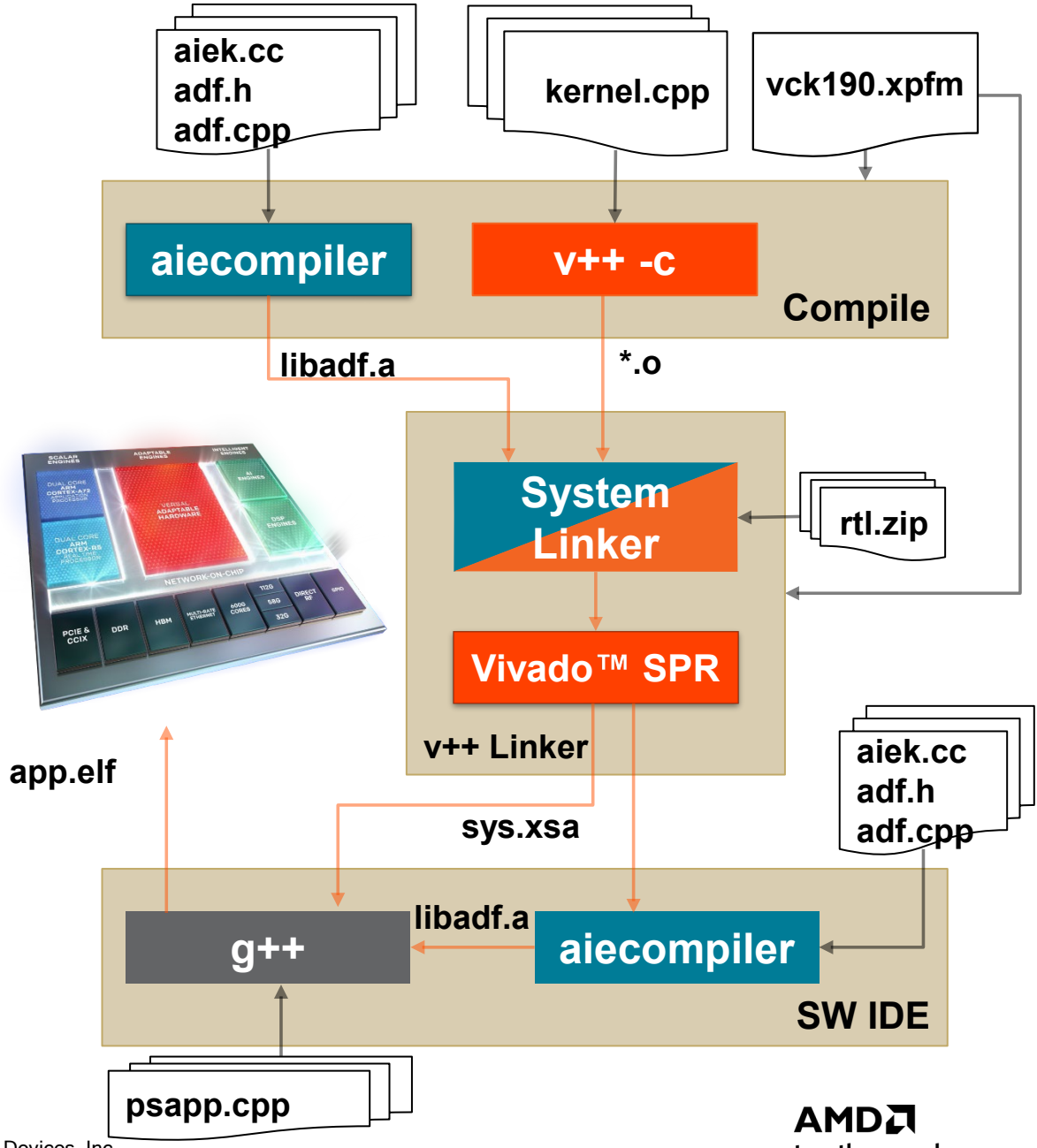
- Hardware build takes more time
- Recommended to perform all optimization in the hardware emulation configuration stage
- Compilation and integration of AIE subsystem and accelerator network into hardware platform
  - Flow automation through HLS, AI Engine compiler, and Vivado™ Design Suite
  - Complete hardware/software system generated by Vitis™ compiler



AMD
together we advance_

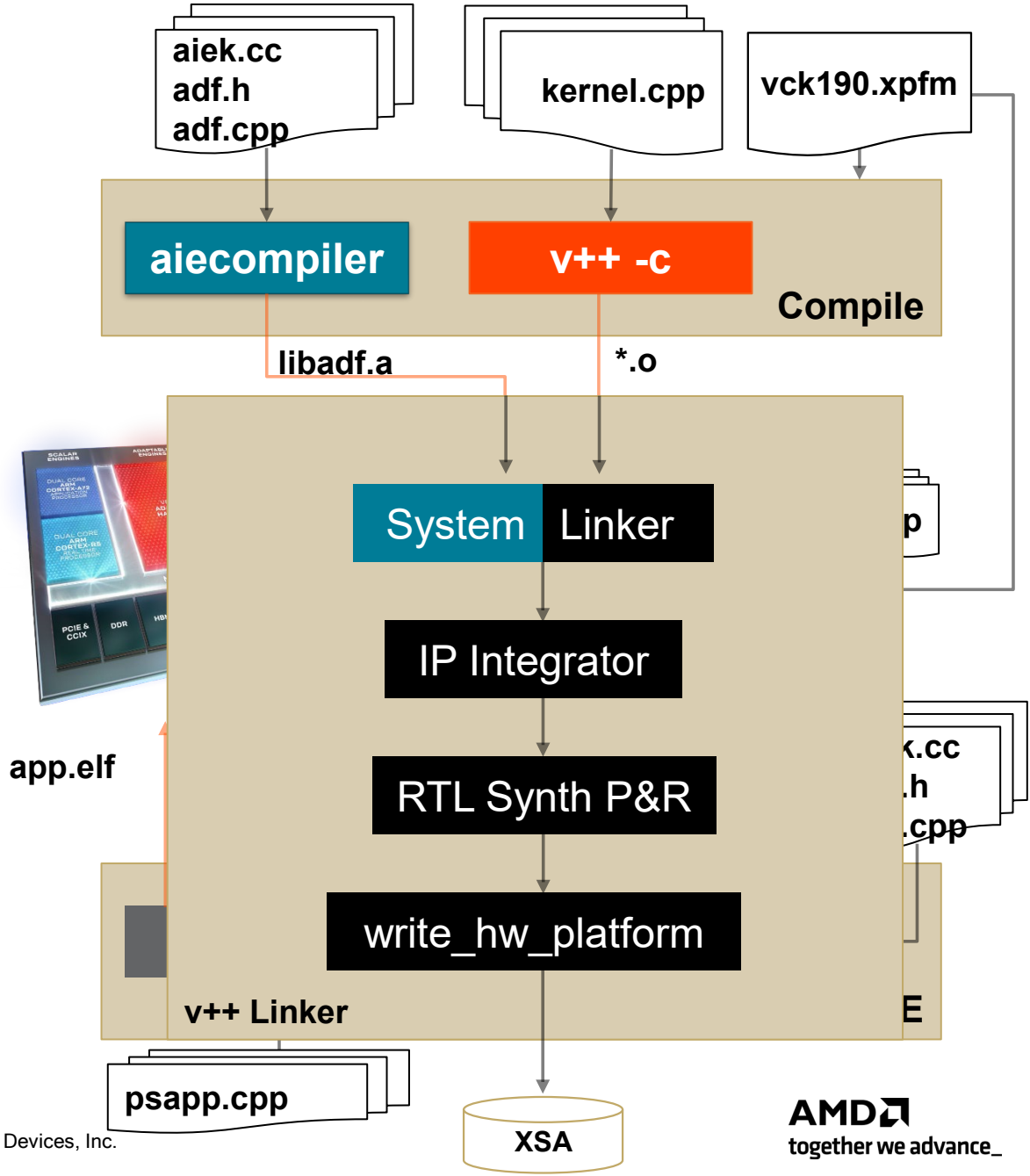
Vitis Tool Hardware Build (2)

- Clock on AI Engine: 1GHz
- Clock on PL region: lower frequency
- Difference between the data throughput of the AI Engine kernels and the PL kernels
- Vitis™ compiler inserts can match the throughput capacities of the PL and AI Engine regions
- Vitis tool generates the hardware and software system using the Vitis compiler
- Software development against fixed PL design
  - PS code iterations
  - AIE kernel and graph iterations subject to fixed PL constraint



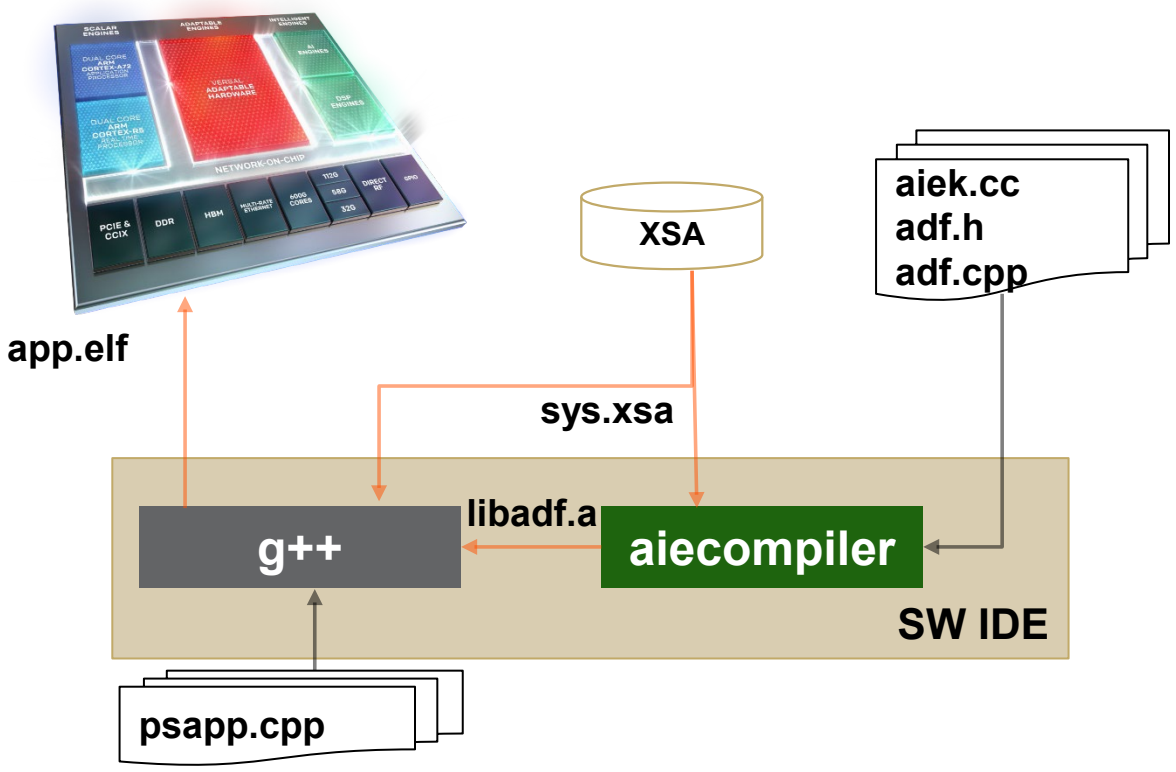
Vitis Tool Hardware Build (3)

- Compilation and integration of AIE subsystem and accelerator network into hardware platform
  - Flow automation through HLS, aiecompiler, and Vivado™ Design Suite
  - Complete hardware/software system generated by Vitis™ compiler
- Software development against fixed PL design
  - PS code iterations
  - AIE kernel and graph iterations subject to fixed PL constraint

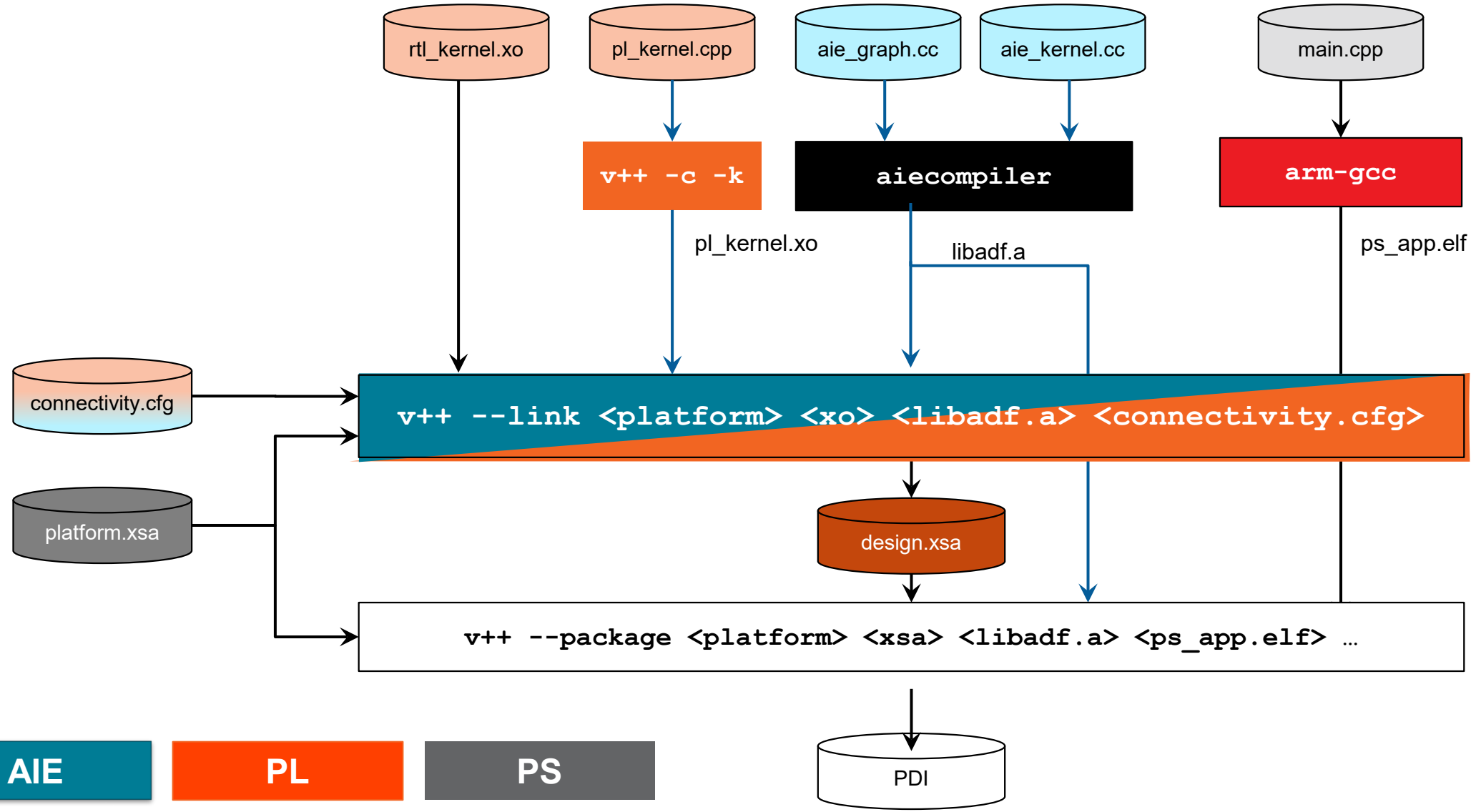


Software Development

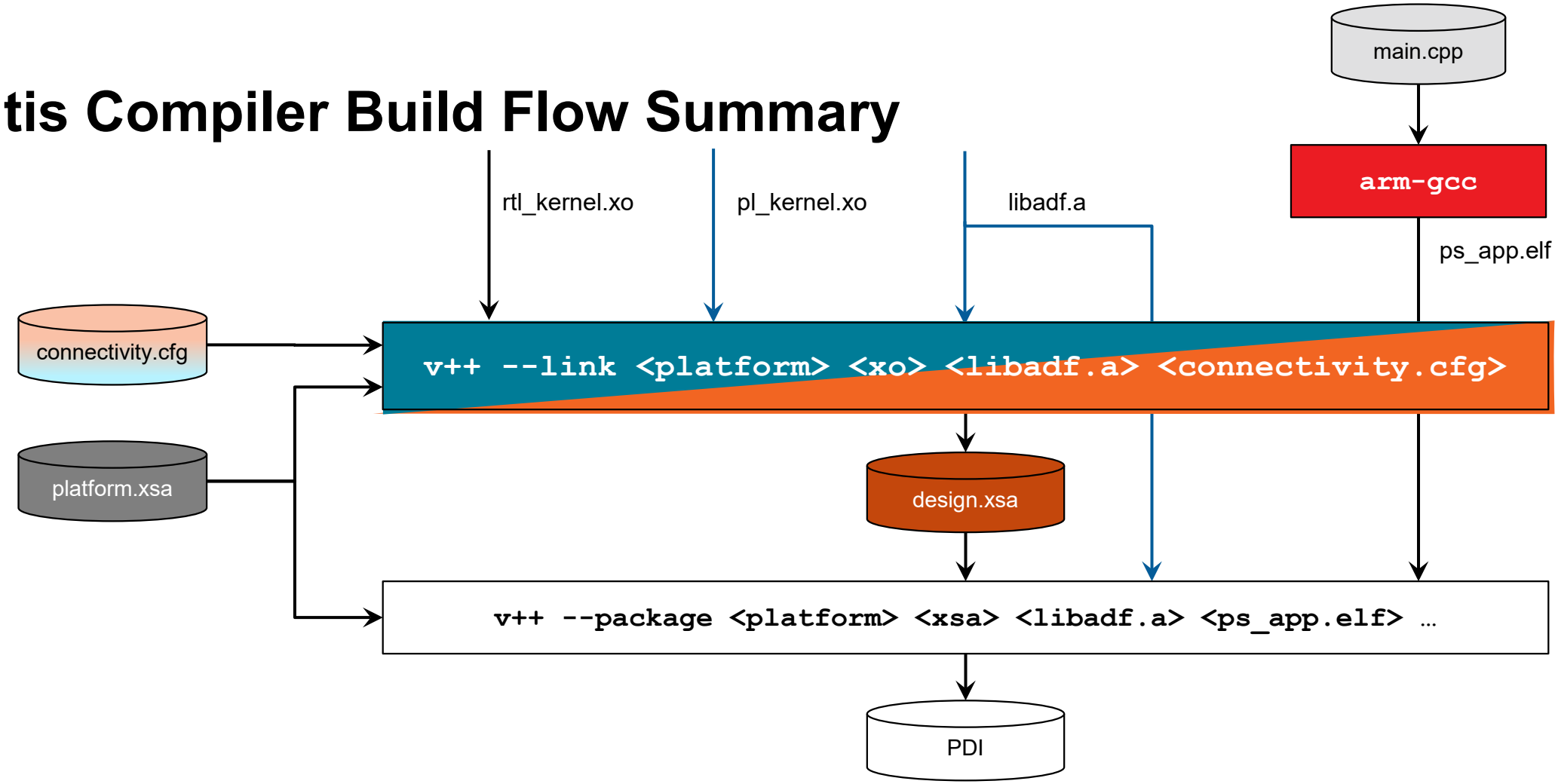
- Iteratively build application against the platform
- Host code development
- Verify on hardware
- Software performance tuning



Vitis Compiler Build Flow Summary



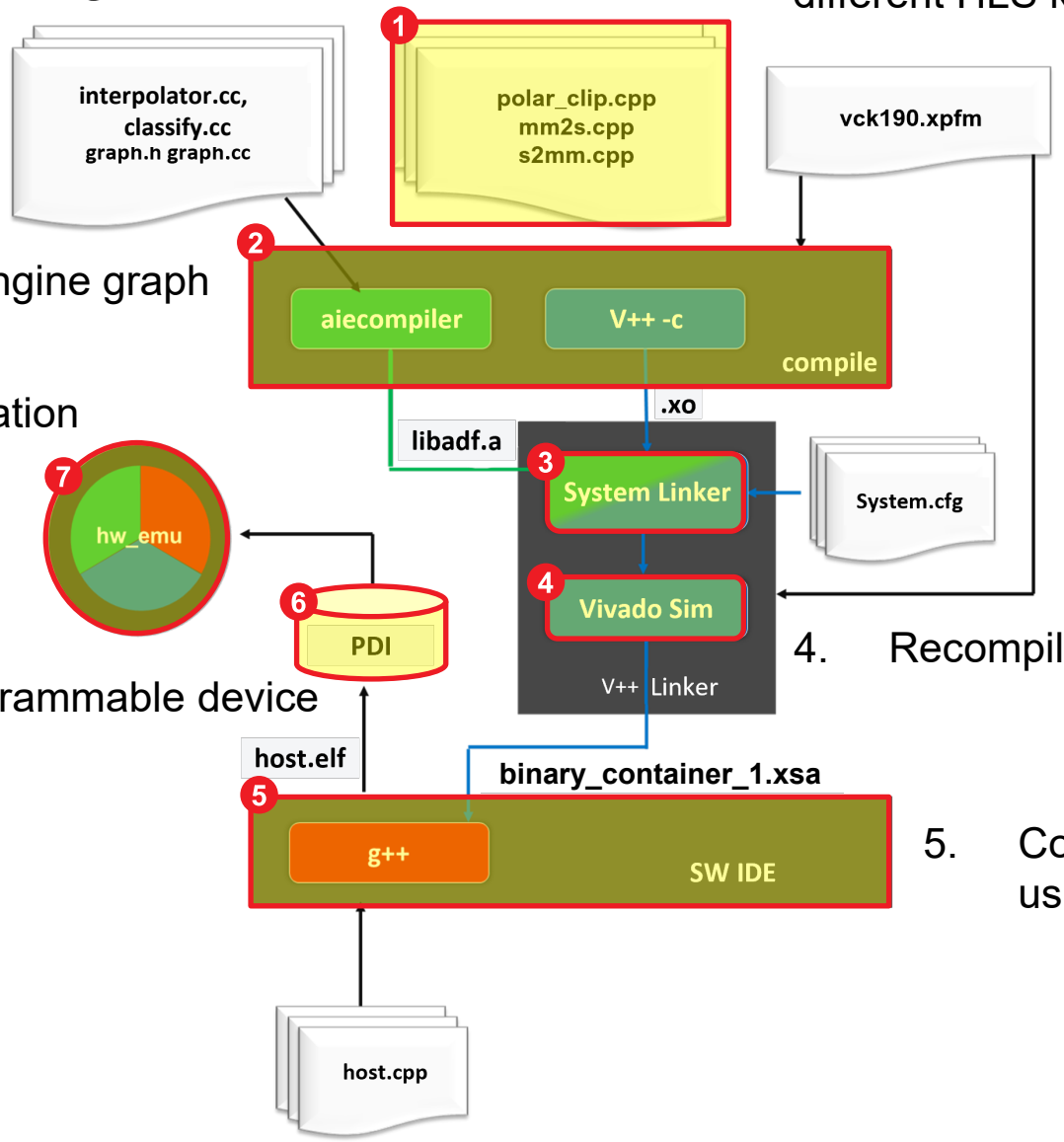
Vitis Compiler Build Flow Summary



- Vitis™ packager:
  - Add the PS application and firmware
  - Generate the required setup to run hardware emulation
- PS application controls the AI Engine graph
  - Graph APIs generated by the AI Engine compiler or the standard XRT APIs to control the AI Engine graph
  - Control the PL kernels, it is recommended to use the standard XRT APIs

Example Design Flow

1. Build and compile the different HLS kernels



2. Compile the AI Engine graph

3. Use the Vitis™ compiler to link the AIE, HLS, and RTL kernels with the platform
   - Generate a fixed platform for software development

4. Recompile the AIE graph

5. Compile the A72 host application using the new fixed platform

6. Generate the programmable device image (PDI)

7. Run hardware emulation

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 material, 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 material, 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’s products are as set forth in a signed agreement between the parties or in AMD's Standard Terms and Conditions of Sale. GD-18.


AMD