XILINX

使用 Vitis 平台实现自适应计算

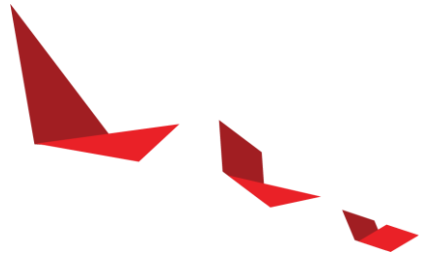
Version: 2021.1

2021年8月

SW&AI Technical Marketing

After This Session...

You will be able to answer these questions



- Why Vitis + Platform can be beneficial?
- What's the architecture of a Vitis platform?
- When should I create a custom platform?
- What are the steps of platform creation?
  - What are the changes in Vitis 2021.1?

All Developers Can Build and Deploy on All Platforms

Build

Embedded Developers

Enterprise Application Developers

Enterprise Infrastructure Developers

Data & AI Scientists

XILINX VITIS™

Deploy

Zynq-7000

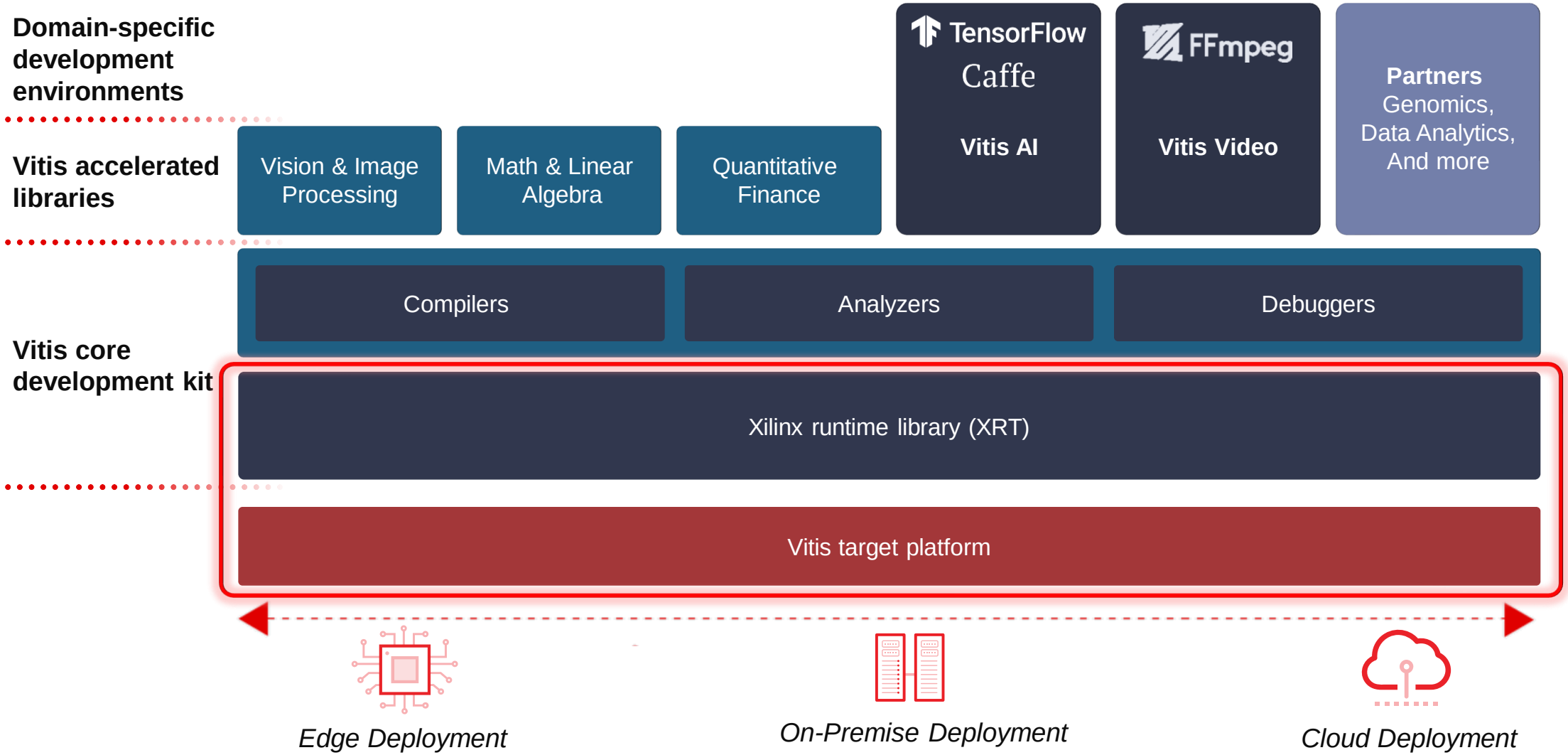
Zynq UltraScale+ MPSoC

Alveo

Versal ACAPs

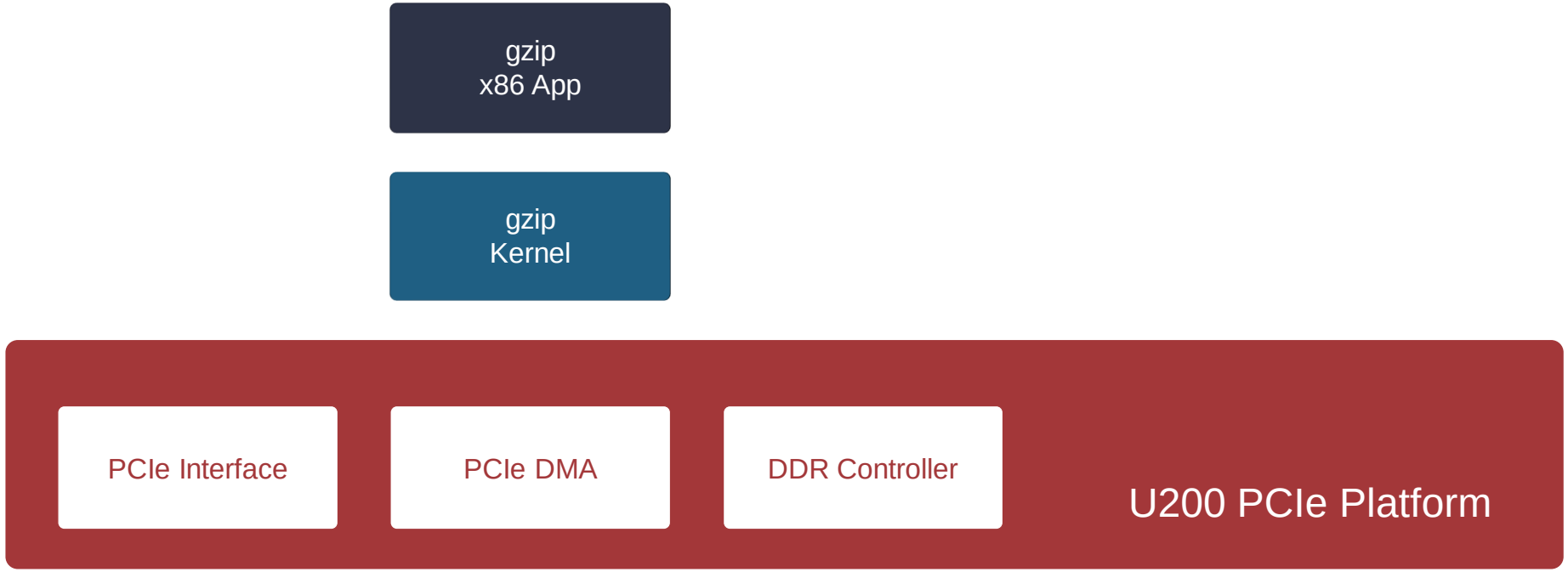
XILINX

Vitis Unified Software Platform

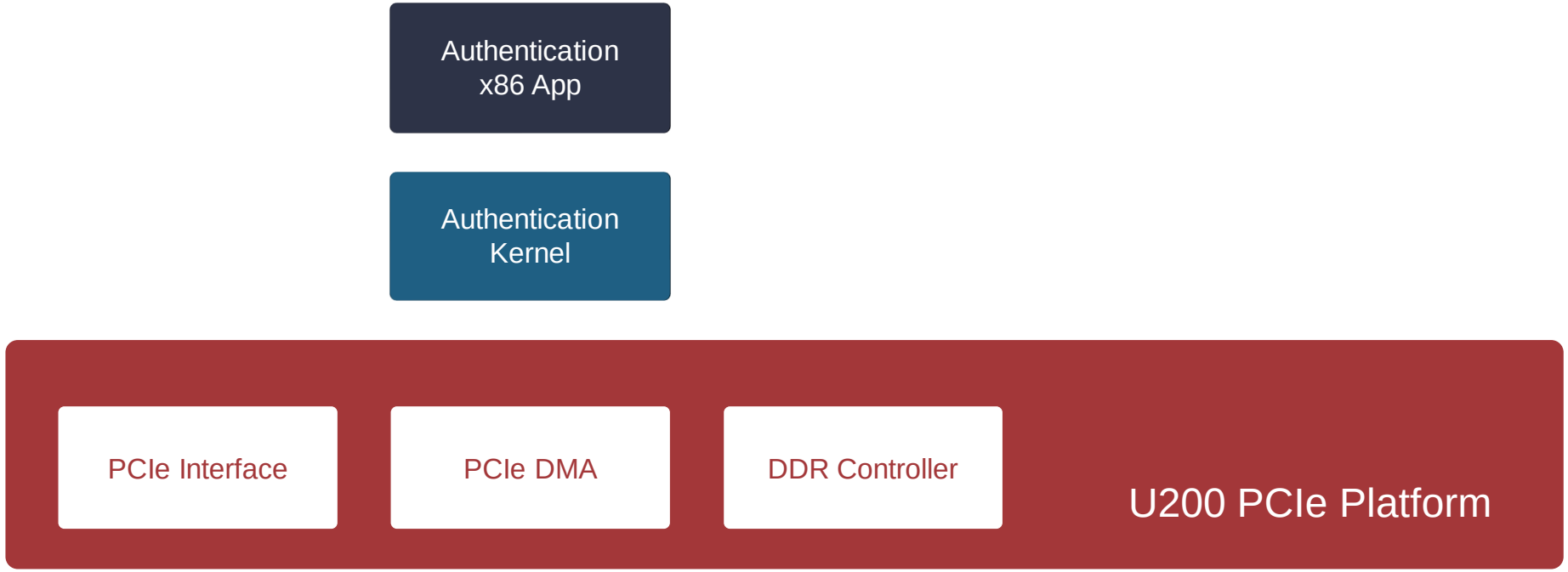


XILINX

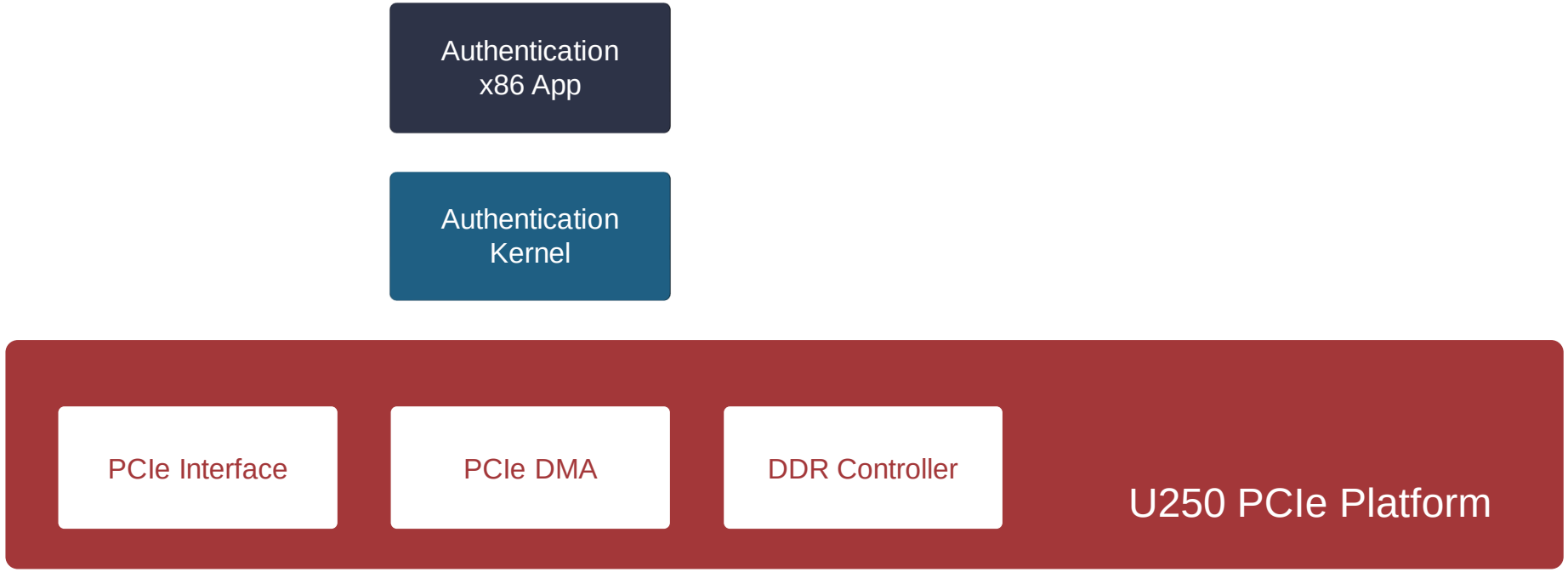
Platform - Kernel Example 1: PCIe Platform



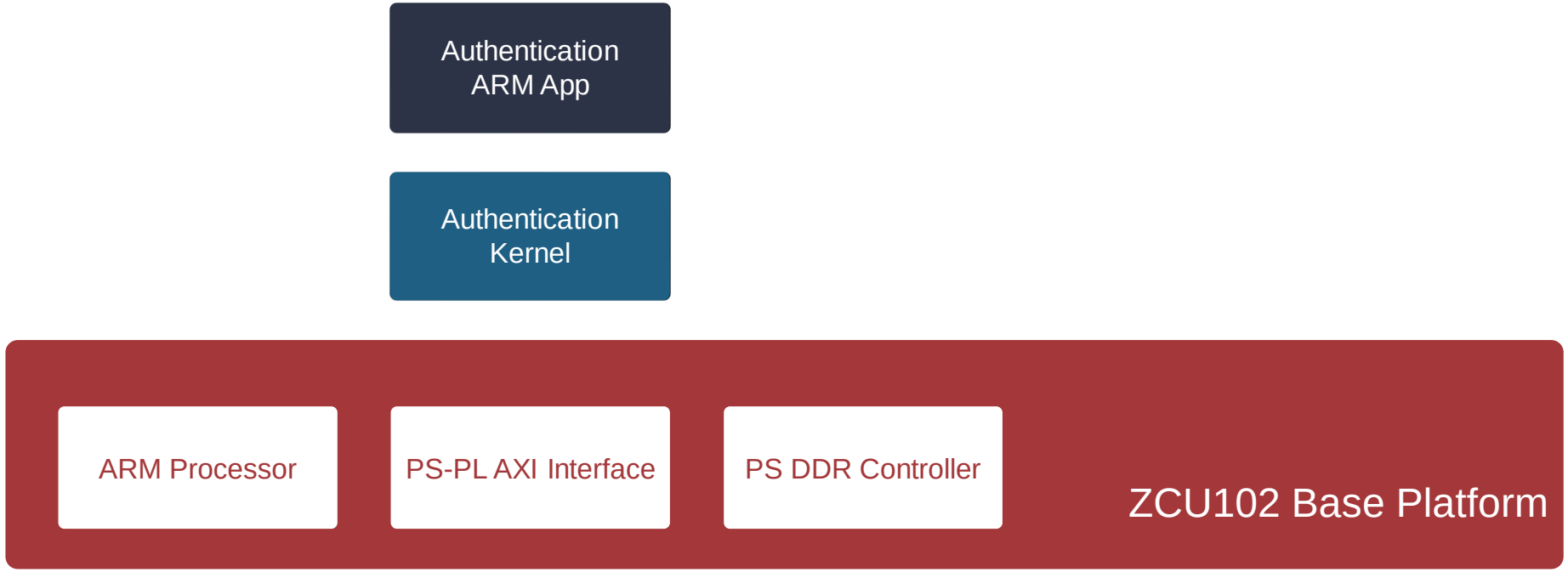
Platform - Kernel Example 1: PCIe Platform



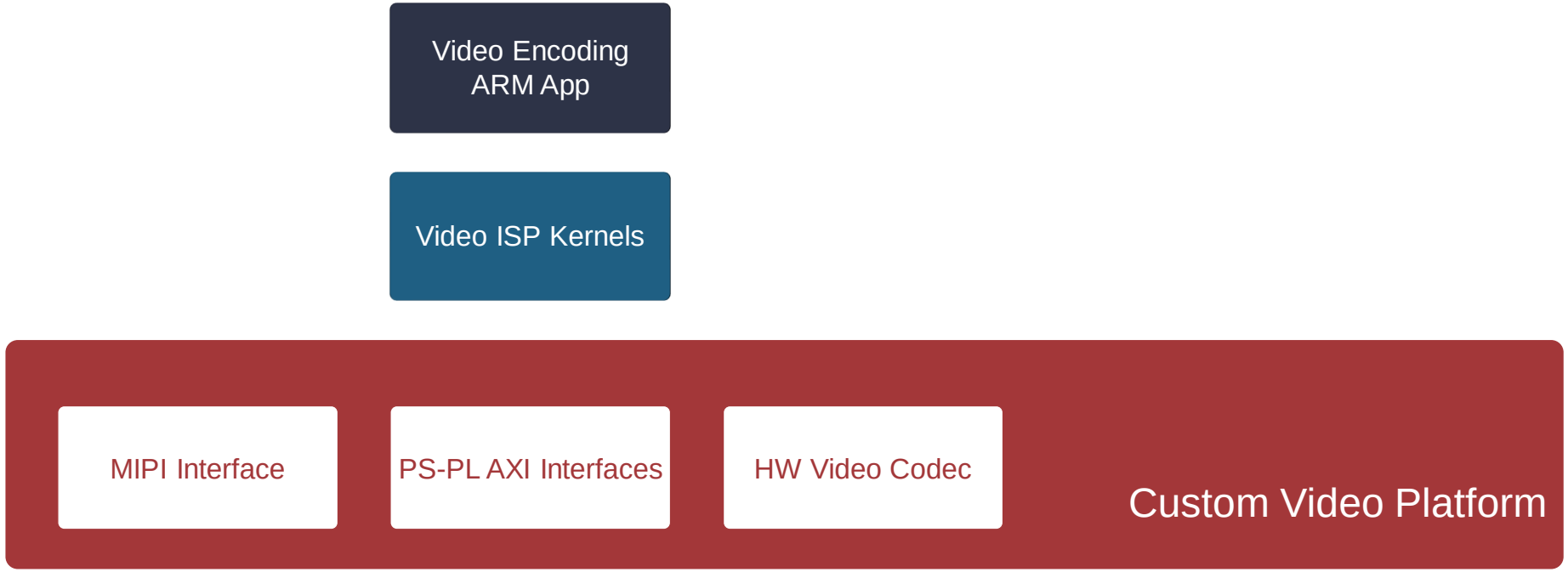
Platform - Kernel Example 1: PCIe Platform



Platform - Kernel Example 1: PCIe Platform

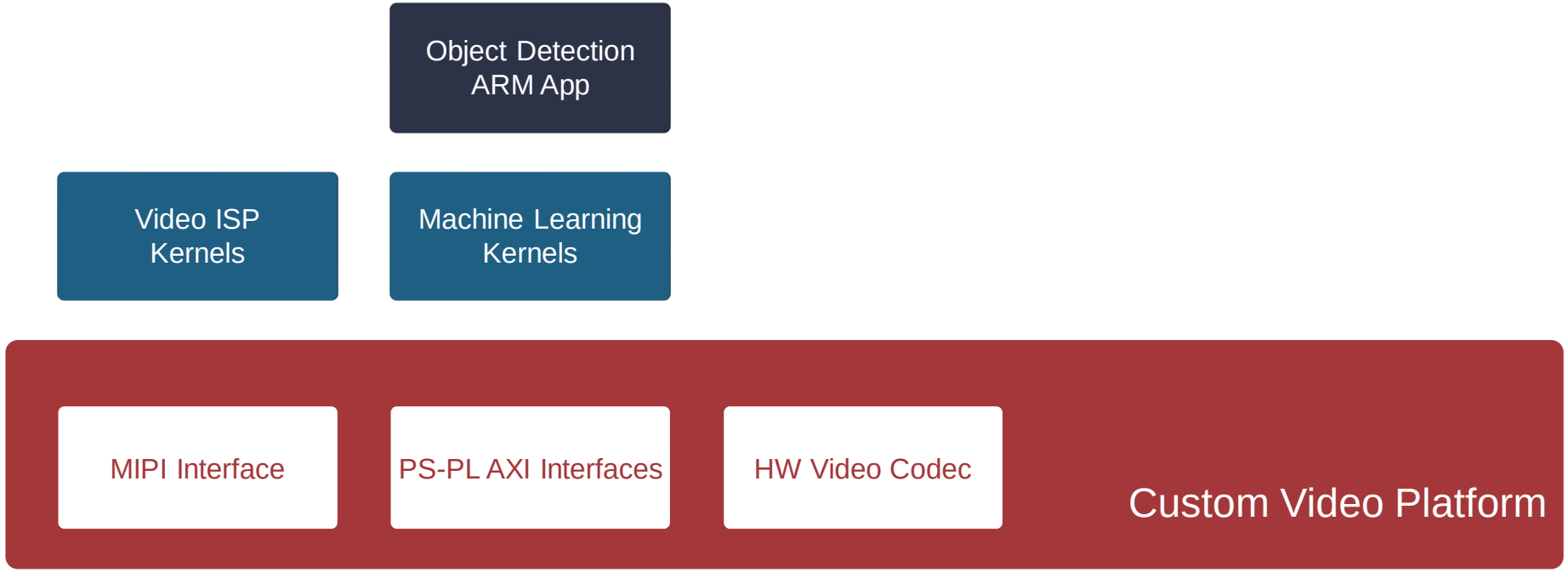


Platform - Kernel Example 2

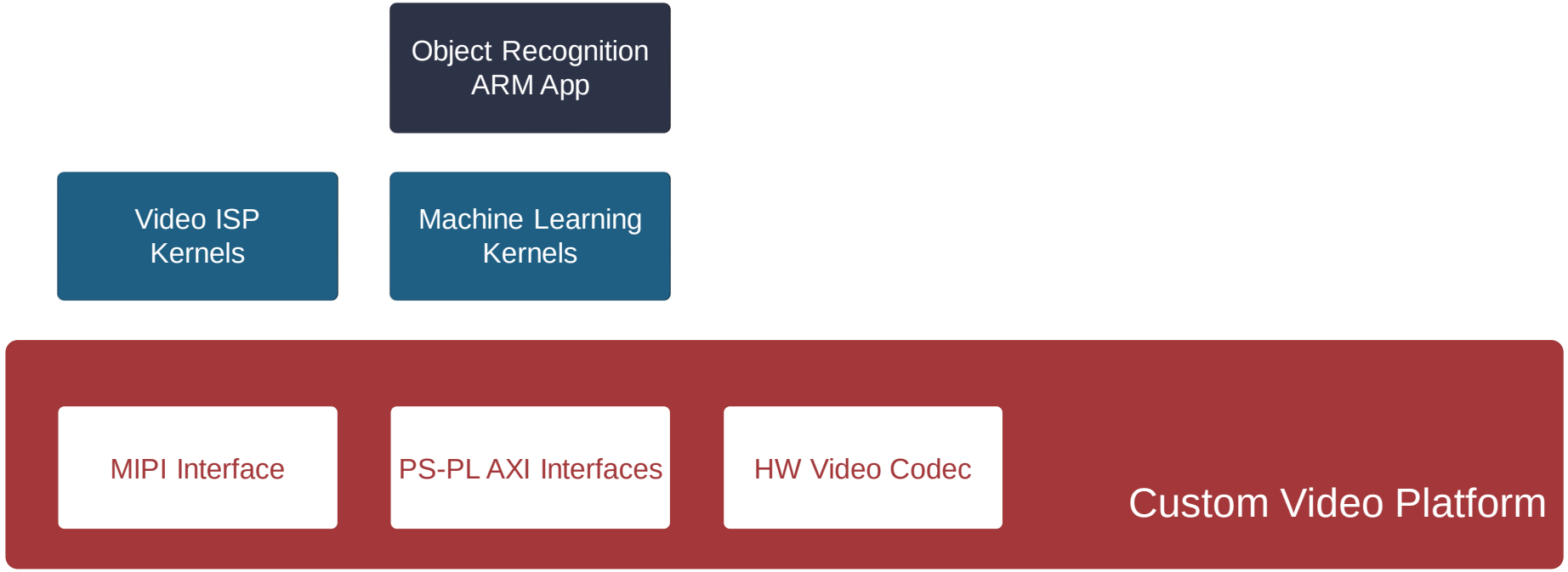


XILINX.

Platform - Kernel Example 2



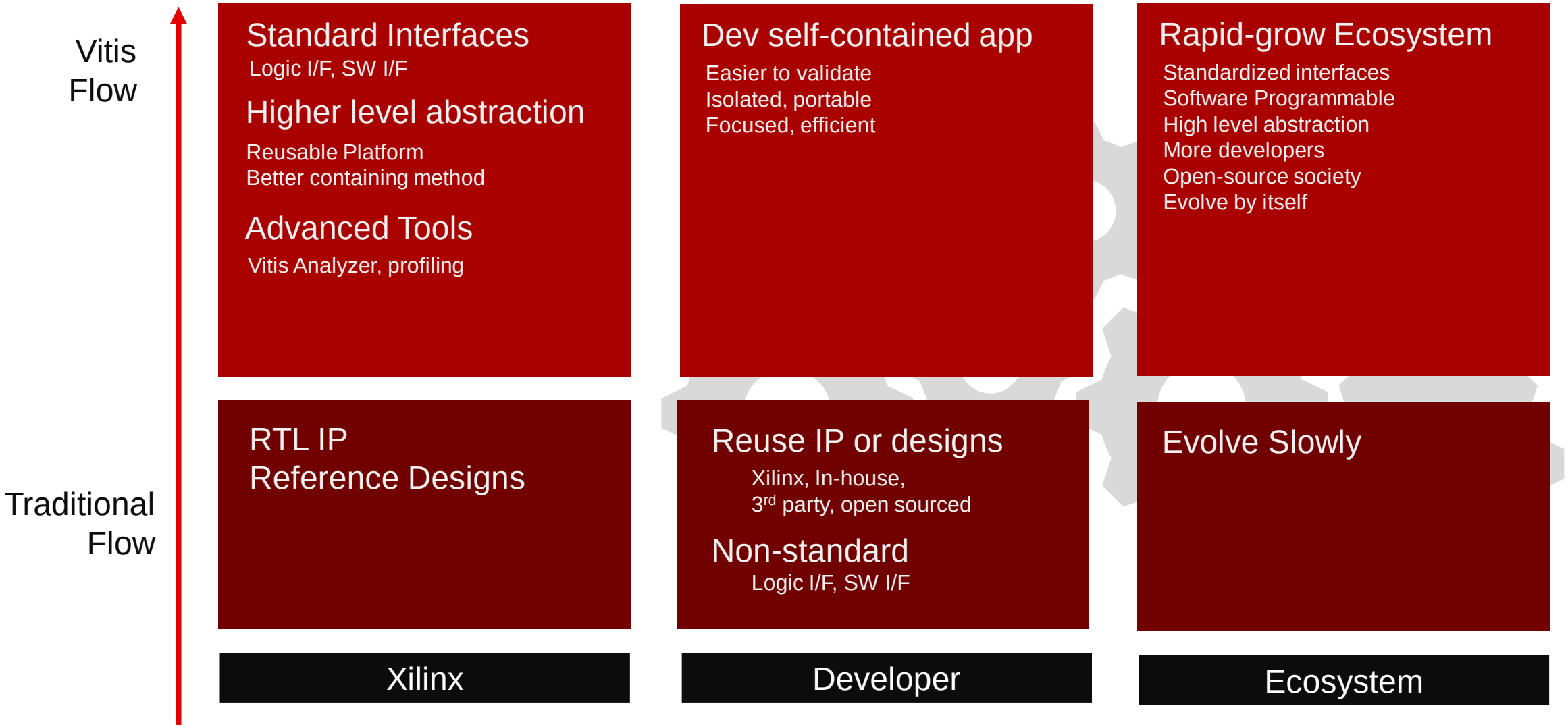
Platform - Kernel Example 2



Platform Common Rules

Platforms Provide	Interface to Kernels
Clock, Reset, Interrupt	Clock, Reset, Interrupt
Interface IO (GPIO, SPI, IIC)	N/A (PS control) or AXI-MM
Interface IP that needs system driver (EMAC, MIPI)	AXI-MM or AXI Stream
ARM Processors	AXI-MM
DDR Controllers	AXI-MM
Non-AXI Interface IP	

From Traditional Flow to Vitis Platform Flow



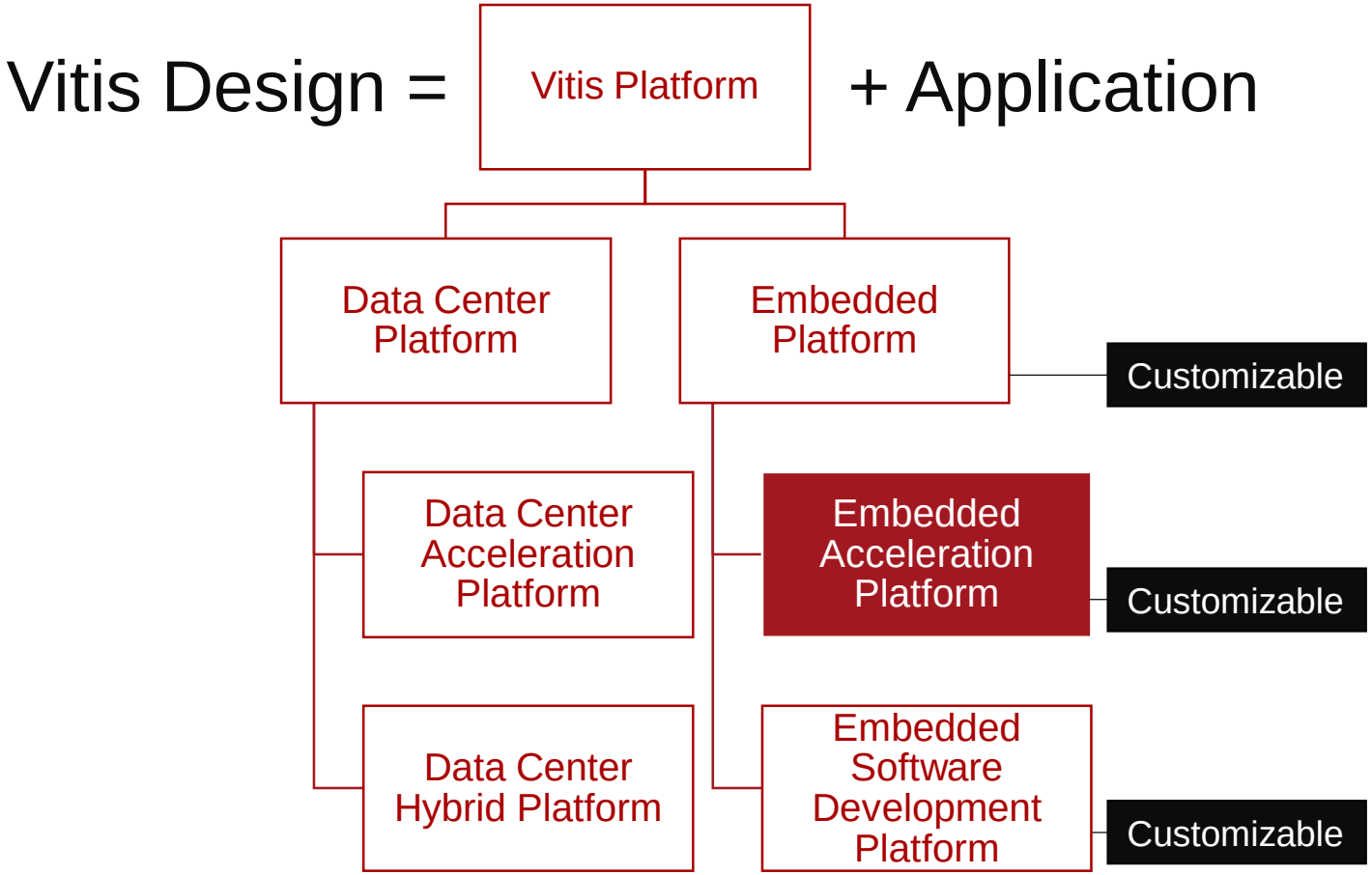
XILINX.

Platform Architecture 平台架构

Types, Components and Relationship

XILINX.

Vitis Platform Types

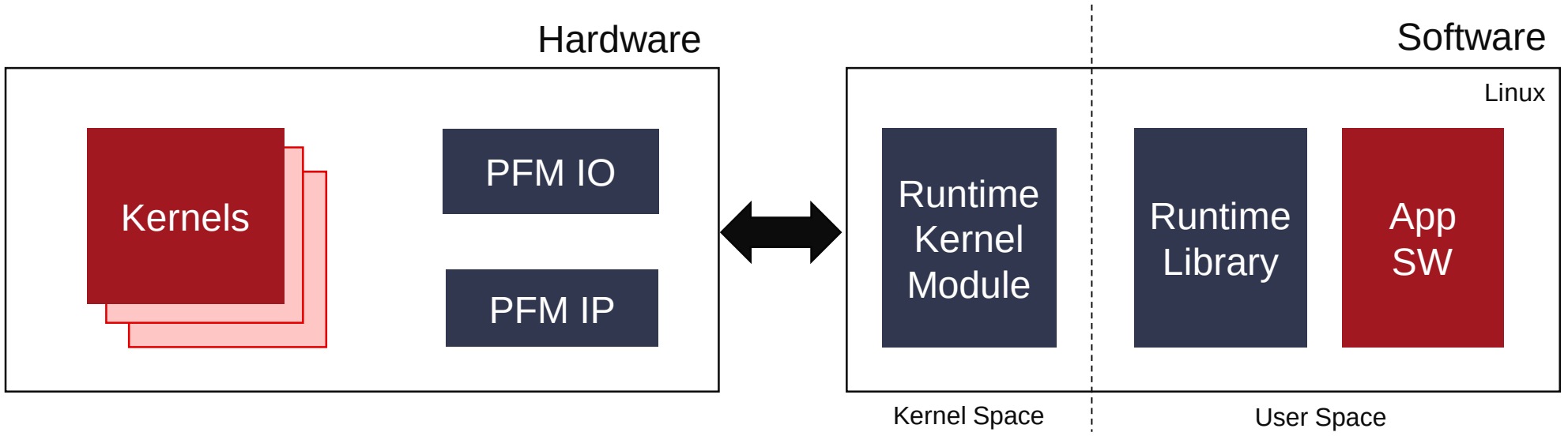


XILINX.

Platform-based Acceleration Application Development

Embedded Acceleration Use Case

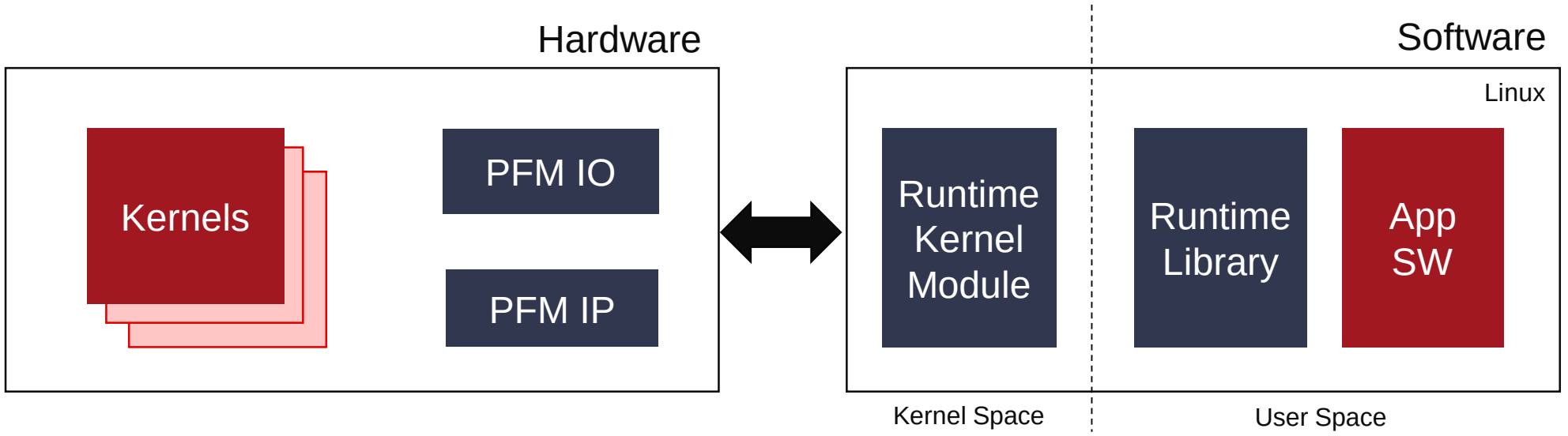
- Platform provides
  - Hardware
    - A pre-configured “wrapper” containing I/O, and other functionality.
  - Software (embedded only)
    - A bootable system with XRT installed
- User develops
  - Acceleration kernels
    - Vitis connects kernels to the platform hardware.
  - Software applications
    - Control acceleration kernels
    - Vitis packages everything to a bootable image.



XILINX.

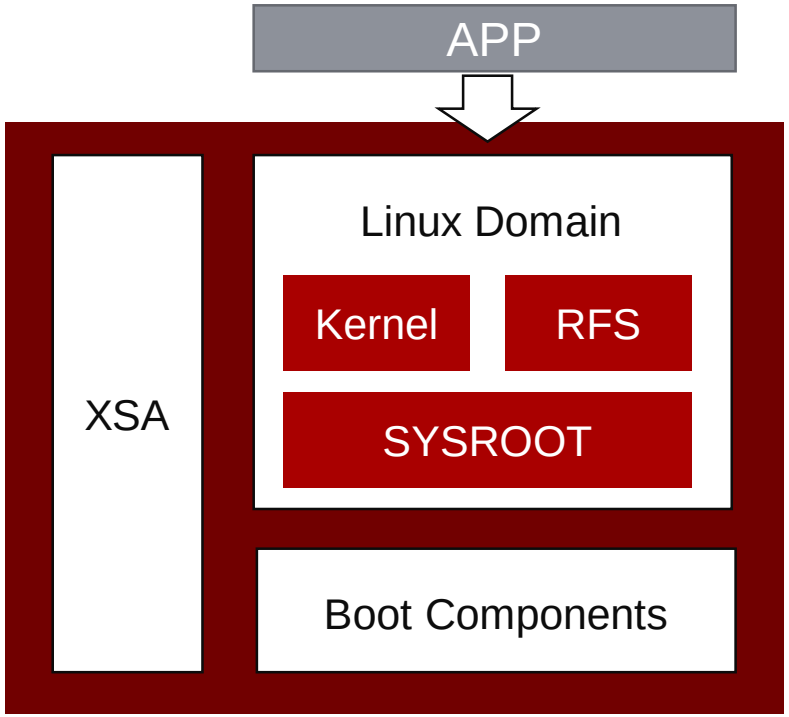
Platform and Application Components

- Platform provides (XPFM)
  - Hardware Platform (HPFM)
    - The hardware design including IO definition, IP definition and interface properties
  - Software Platform (SPFM)
    - Linux domain (Kernel, RootFS with XRT, Sysroot)
    - Boot Images (BIF, FSBL, PMUFW, U-boot, Device Tree)
- User develops
  - Acceleration kernels (XO)
  - Link Hardware with Platform (XCLBIN)
  - Software applications (ELF)
  - Package Boot Image (sd_card.img)

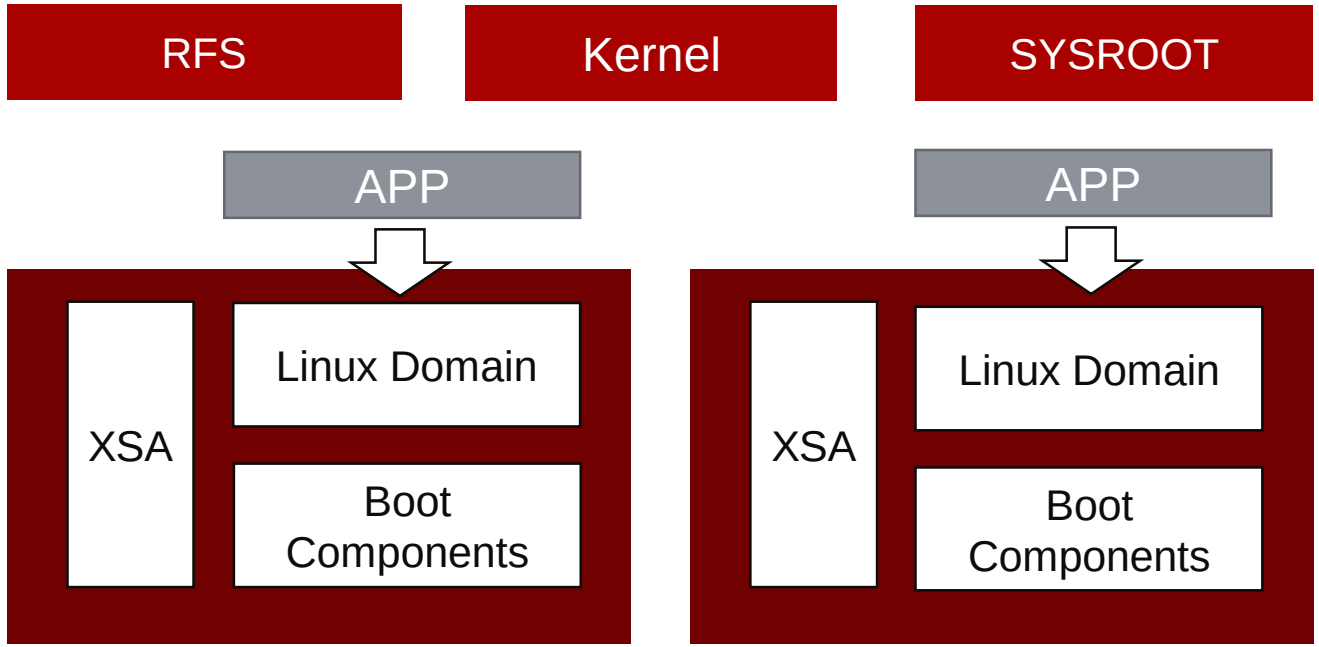


Two ways to organize software components

- Pack into Platform



- Assign with System Project
  - RFS, Kernel and SYSROOT can share among platforms



XILINX.

Vitis makes designs ~~complicated?~~ simplified with pre-built platforms

XILINX.

Examples for Platform Selection

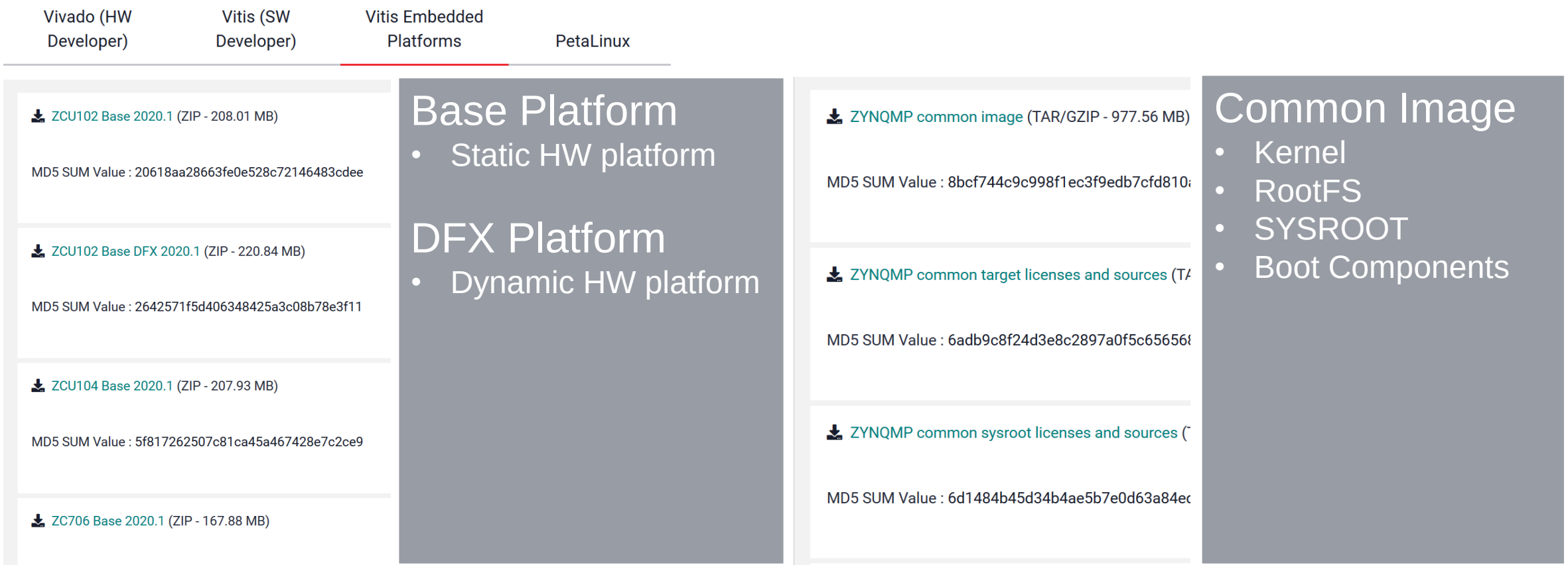
Design Step	Purpose	Target Board	Platform
Evaluation	- Learn Vitis Acceleration Flow - Evaluate Vitis Libraries - Evaluate Vitis-AI Performance	Xilinx Demo Board	Xilinx Pre-built Platforms
Develop/PoC	- Build Custom Kernel - Add Custom Kernel to Acceleration Pipeline	Xilinx Demo Board	Xilinx Pre-built Platforms
Customization	- Add Custom IO Interfaces - Design Final Product	Xilinx Demo Board Custom Board	Custom Platform

Start with a Pre-built Platform
从预编译的平台开始设计

XILINX.

Download Pre-built Vitis Embedded Platforms

- Xilinx Download Center www.xilinx.com/download



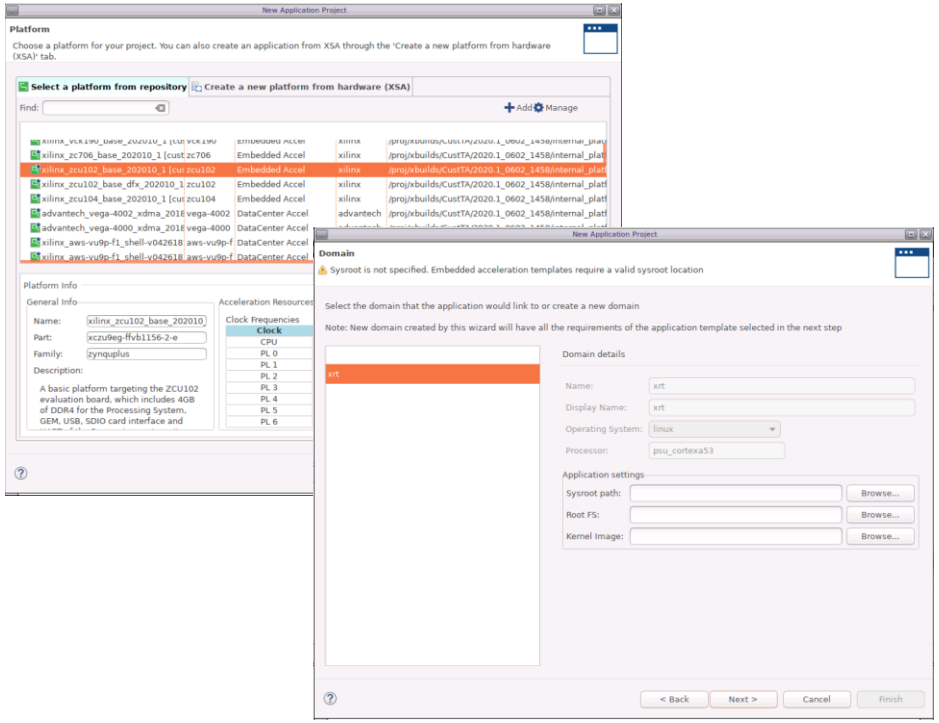
XILINX.

Install Platforms

- Install Platform
  - Extract to platform search path
- Platform Search Path
  - /opt/xilinx/platforms
  - $XILINX_VITIS/platforms
  - $PLATFORM_REPO_PATHS
- Install Common Image
  - extract common image to any path
- Install SYSROOT
  - Run sdk.sh: self extracted package

Use a Platform

- GUI



- CLI
  - v++ compiling and linking options
    - *v++ -c --platform=<platform_name>*
    - Can be embedded in config file
  - Host app cross compile environment
    - *export SYSROOT=<sysroot_path>*
  - v++ packaging options
    - *--package.kernel_image <arg>*
    - *--package.rootfs*

Example
https://github.com/Xilinx/Vitis-Tutorials
- Getting_Started/Vitis/example/zcu102
- /hw/Makefile
- src/zcu102.cfg

V++ Reference Manual
https://www.xilinx.com/html_docs/xilinx2021_1/vitis_doc/vitiscommandcompiler.html

Install Additional Software with Package Feed

- What is a package feed?
  - Install software packages on-the-fly
  - Like *apt* for Ubuntu, *yum/dnf* for CentOS
  - PetaLinux uses *dnf*

```
dnf install git
```

- What are the benefits?
  - Skip PetaLinux rootfs recompilation
- Who provides these packages?
  - Xilinx hosts pre-compiled packages on http://petalinux.xilinx.com
- Which packages are included?
  - All packages available with PetaLinux

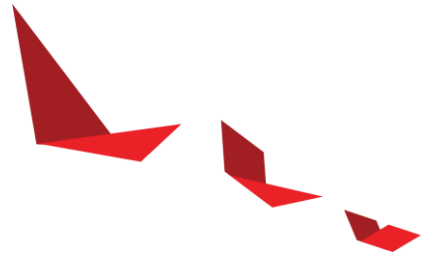
- How to use a package feed?
  - Enable `dnf` in rootfs
    - dnf is pre-installed in rootfs of common images
  - Set up package feed URL on board

```
wget http://petalinux.xilinx.com/sswreleases/
rel-v2020/generic/rpm/repos/zynqmp_generic_eg.repo
cp zynqmp_generic_eg.repo /etc/yum.repos.d/
dnf clean all # Clean dnf local cache
```

    - More archs are supported: zynq, ev, cg, dr, etc.
  - Install packages on board
    - *dnf install <package name>*

```
Doc: UG1393 Using Vitis Platforms
```

Agenda Checkpoint



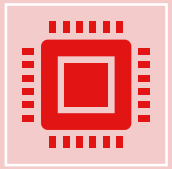
- Why Vitis + Platform can be beneficial? ✓
- What's the architecture of a Vitis platform? ✓
- When should I create a custom platform? ✓
  - Base platform usage
- What are the steps of platform creation?

Create Embedded Acceleration Platform from Scratch
创建嵌入式加速平台

XILINX.

Step 0: Base Bootable Design

Step 0: Base Bootable Design



Boot to Linux

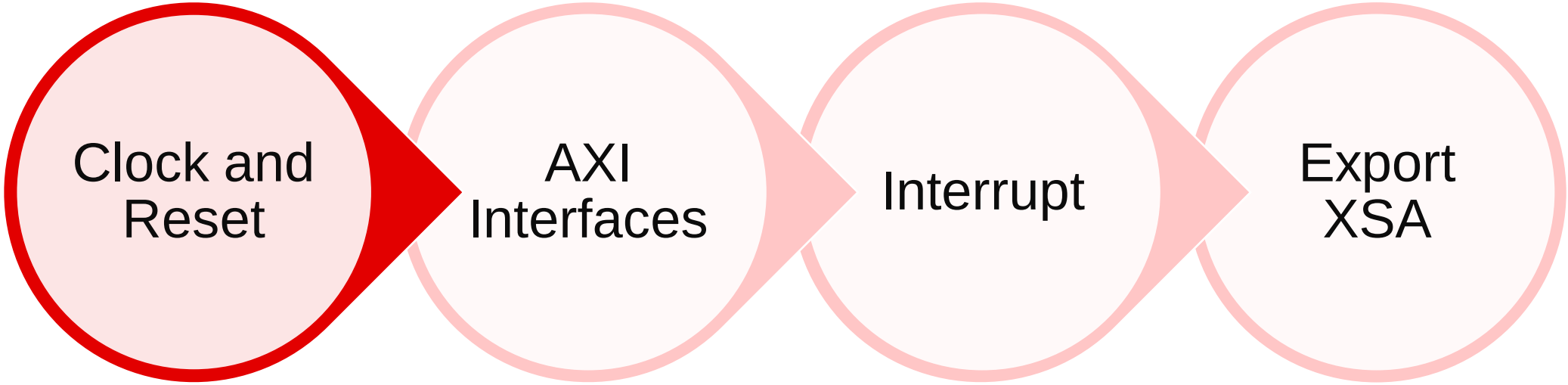
- Verify PS Configuration
- Verify DDR
- Verify Peripheral functions

Design Methods

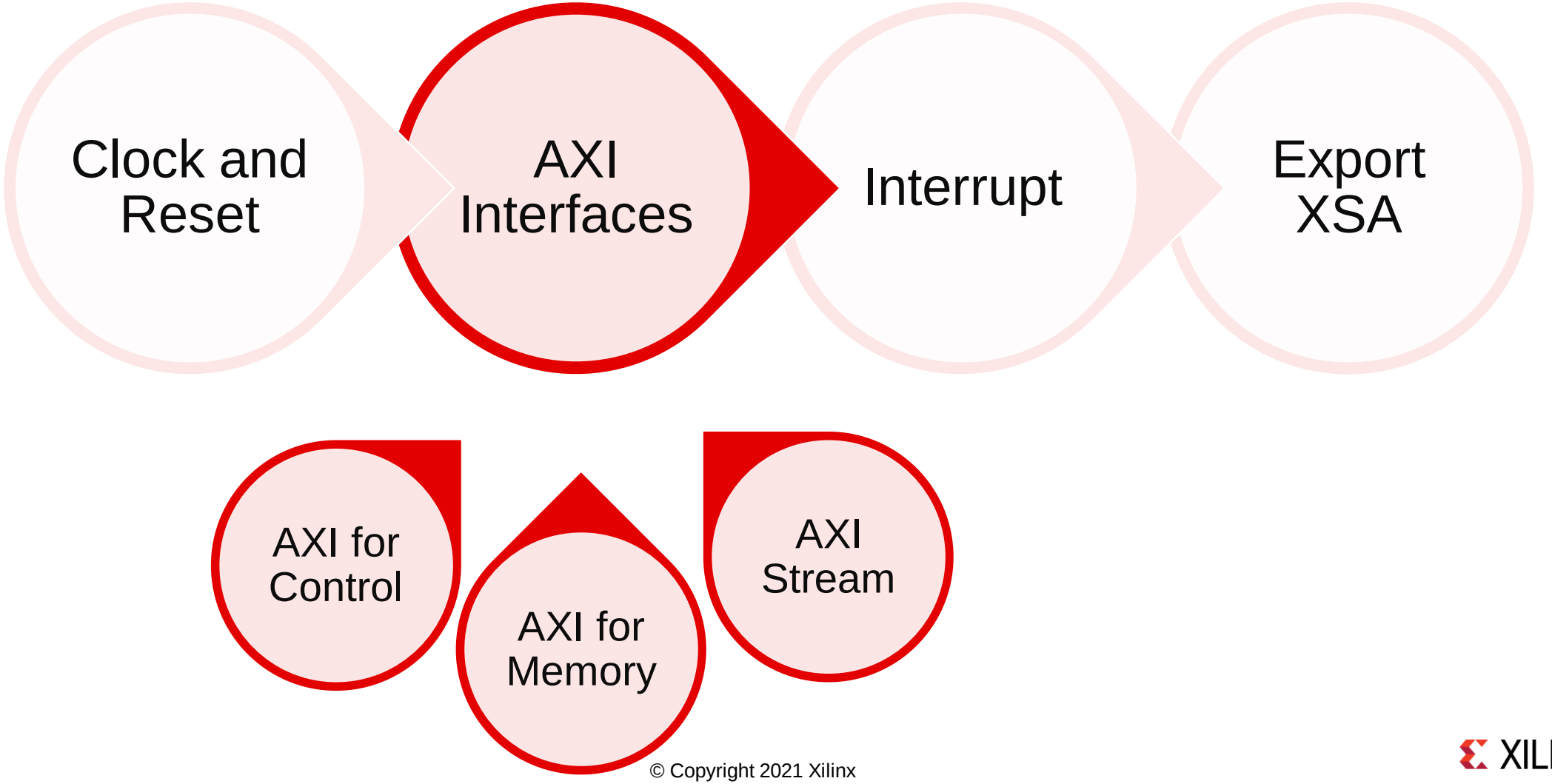
- Xilinx boards: Vivado preset and PetaLinux BSP can be used
- Custom boards: Setup pinout and PS settings according to your board

Step 1: Update Hardware Design in Vivado

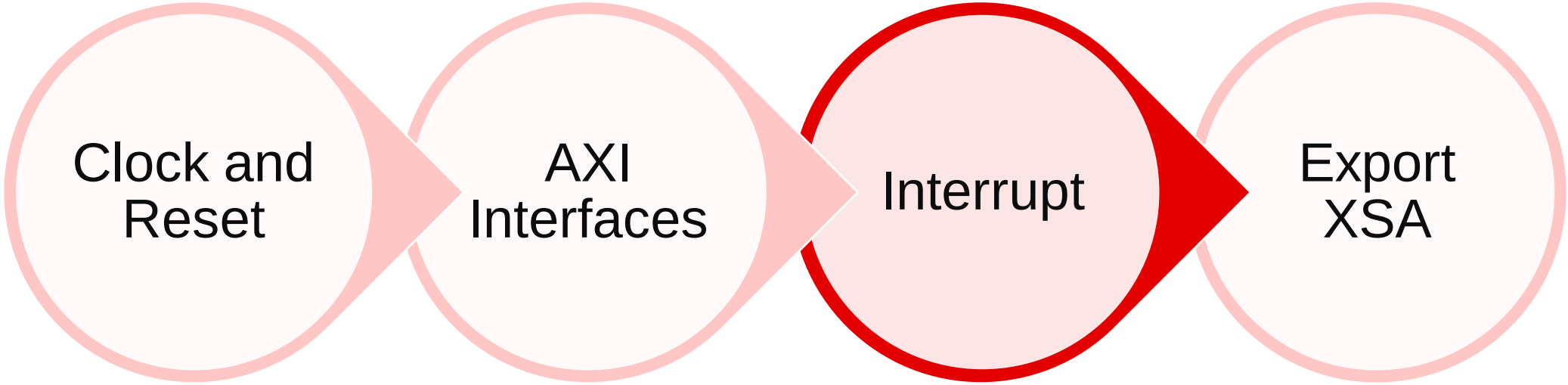
Step 1: Update Hardware Design in Vivado



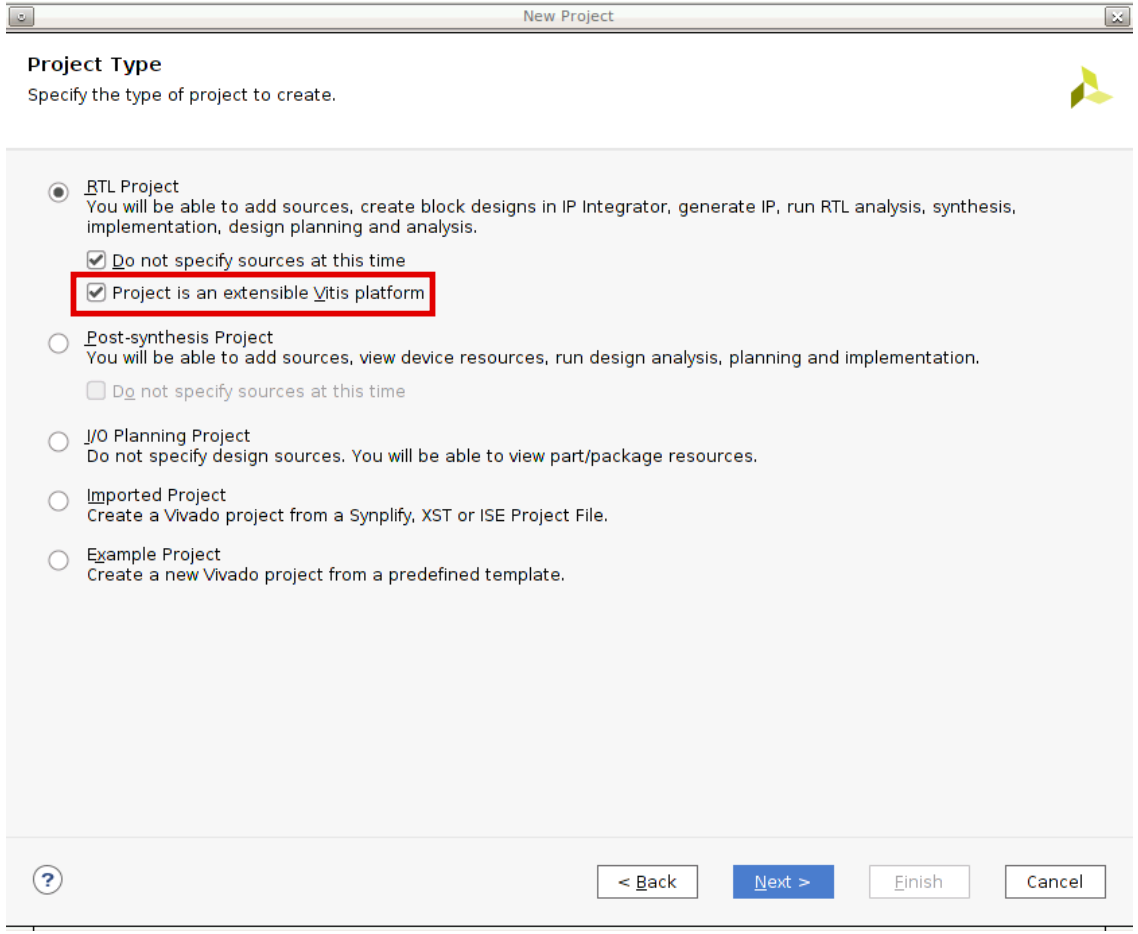
Step 1: Prepare Hardware Design in Vivado

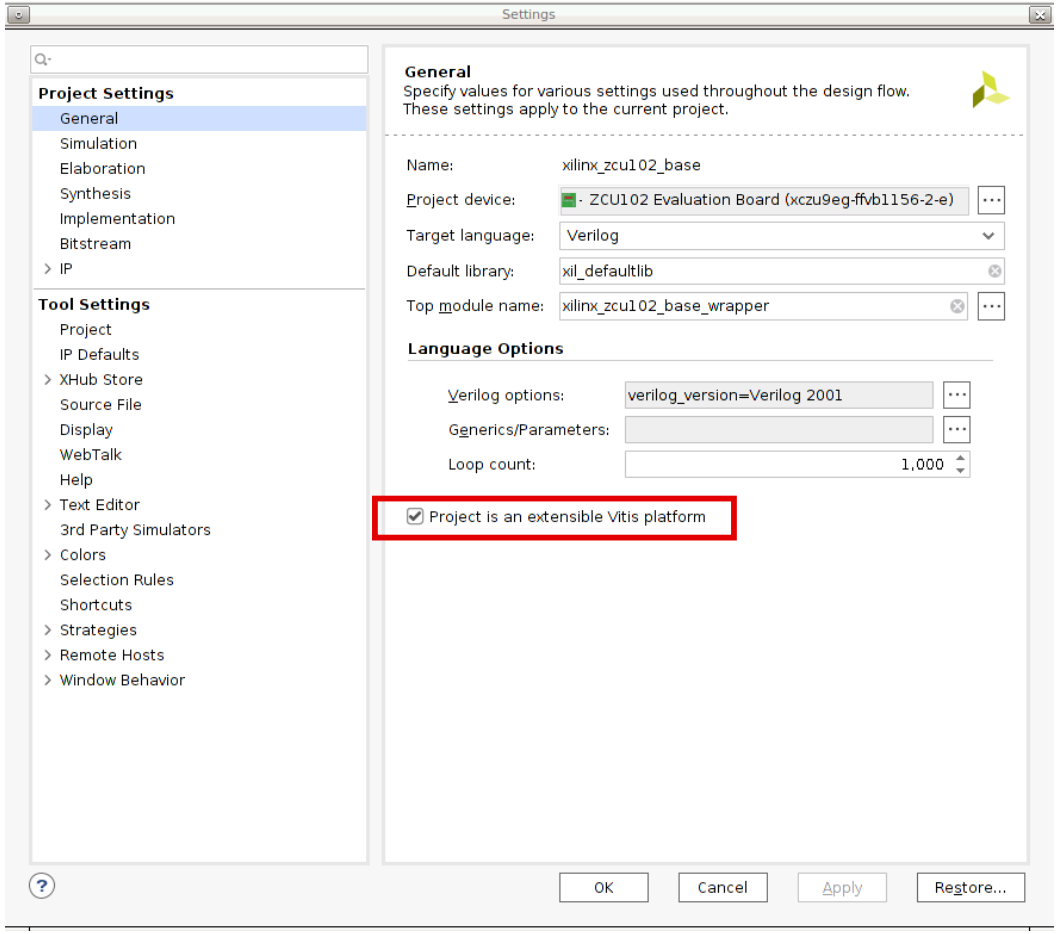


Step 1: Update Hardware Design in Vivado



A. Mark Project as Extensible Platform Project

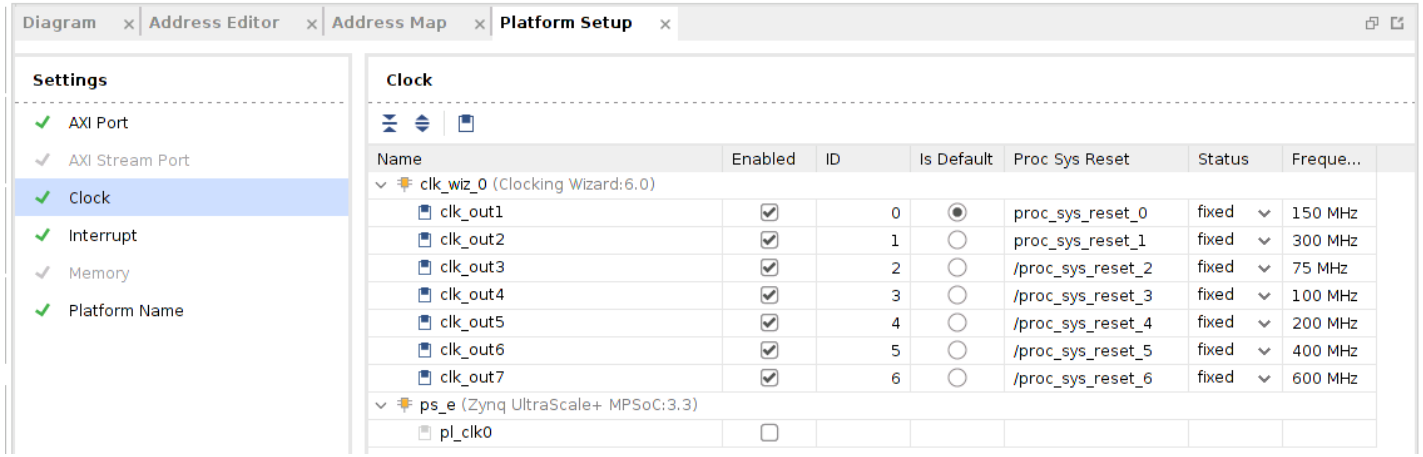




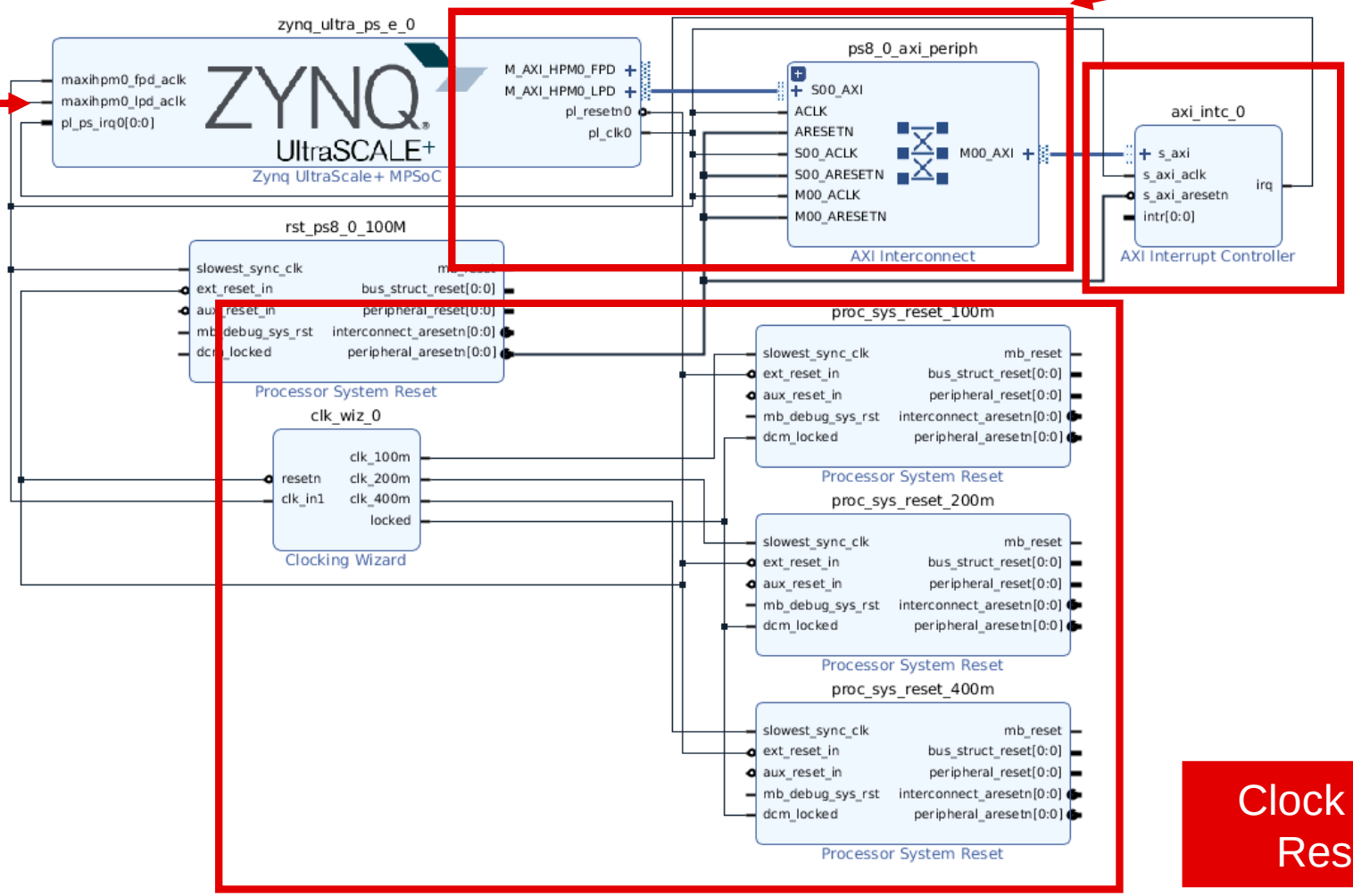
XILINX.

B. Platform Setup

- Vivado GUI
  - Window -> Platform Setup
  - Enable the AXI, clock and interrupt resources

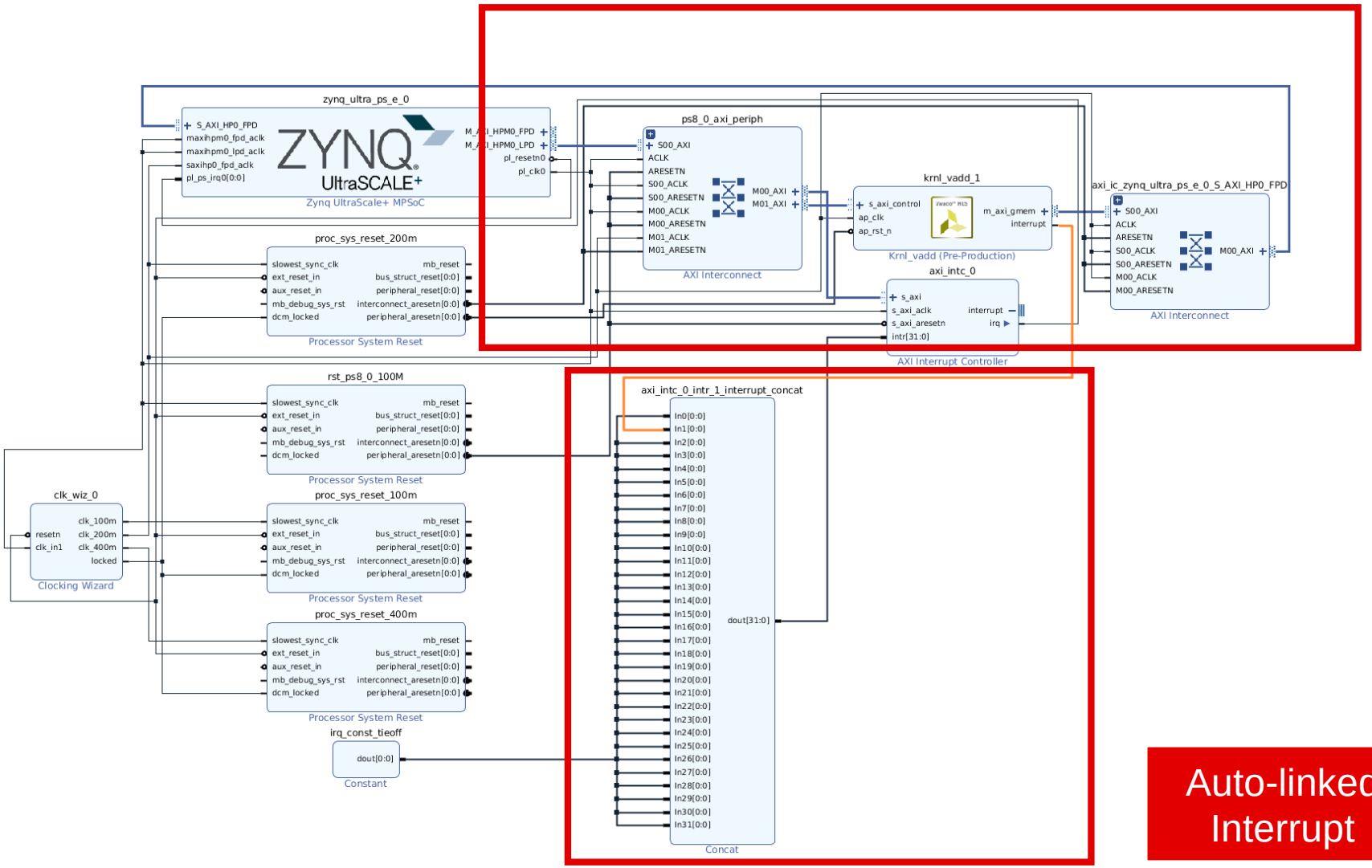


Platform Block Diagram Example



XILINX.

Vitis Linked Vector Addition Block Diagram Example

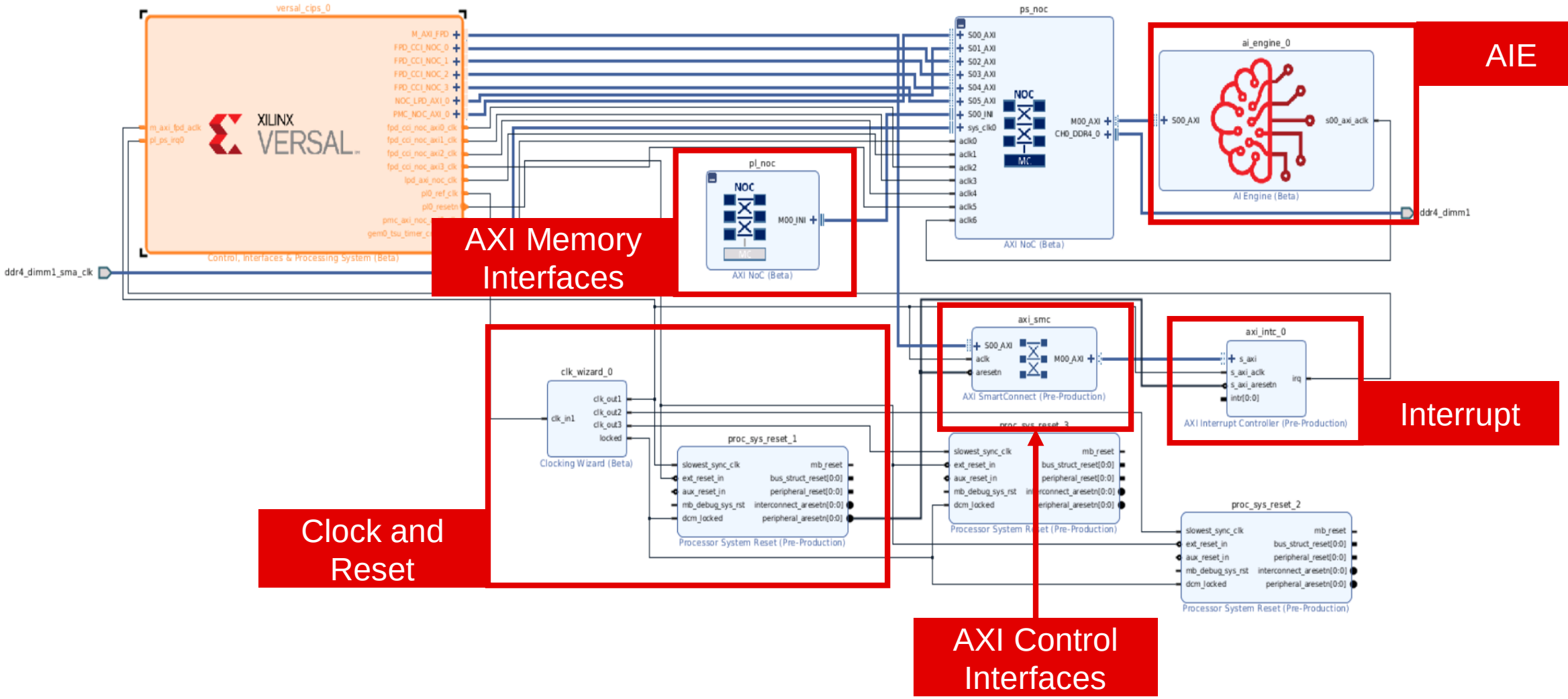


Auto-linked kernel

Auto-linked Interrupt

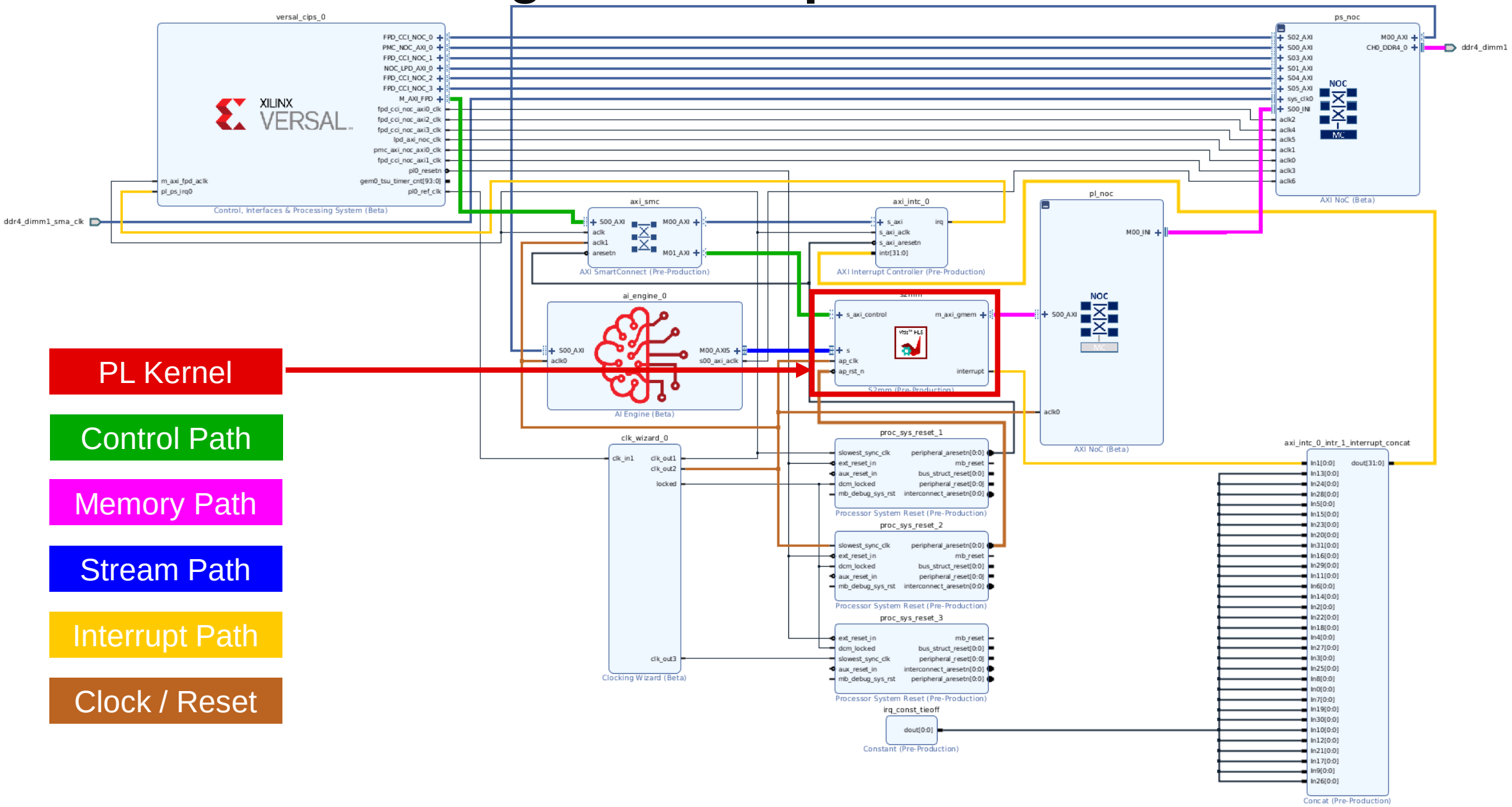
XILINX.

Platform Block Diagram Example



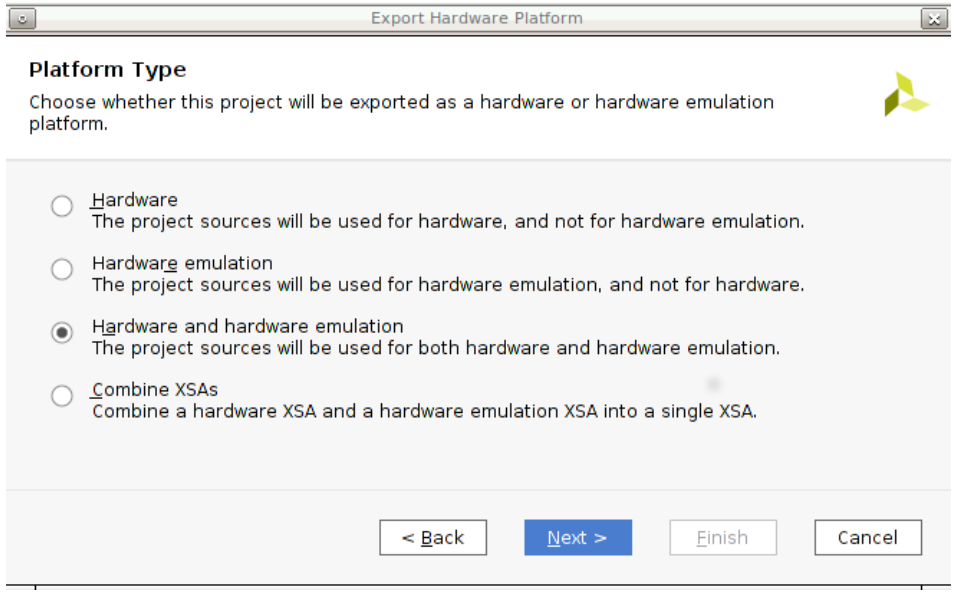
XILINX

Platform Block Diagram Example



C. Export XSA for Software Design

- Vitis GUI
  - Create HDL Wrapper
  - Generate Block Diagram
  - File -> Export -> Export Platform
    - Select Platform Type
    - Select Pre-synthesis
    - Input platform name and description
    - Generate XSA



XILINX.

Step 2: Update Software Environment

Step 2: Update Software Environment

- Supported Software Environments

Software Environment	Supported?
Linux	Yes
Standalone	No
RTOS	Roadmap

- Linux Components Requirements

Linux Component	Requirements
Linux Kernel	-
Root FS	xrt (zocl) (xrt-dev)
Device tree	zocl node override intc `num-intr-inputs` to 32

- Device tree example

```
&amba {
    zyxclmm_drm {
        compatible = "xlnx,zocl";
        status = "okay";
        interrupt-parent = <&axi_intc_0>;
        interrupts = <0  4>, <1  4>, <2  4>, <3  4>,
                     <4  4>, <5  4>, <6  4>, <7  4>,
                     <8  4>, <9  4>, <10 4>, <11 4>,
                     <12 4>, <13 4>, <14 4>, <15 4>,
                     <16 4>, <17 4>, <18 4>, <19 4>,
                     <20 4>, <21 4>, <22 4>, <23 4>,
                     <24 4>, <25 4>, <26 4>, <27 4>,
                     <28 4>, <29 4>, <30 4>, <31 4>;
    };
};

&axi_intc_0 {
      xlnx,kind-of-intr = <0x0>;
      xlnx,num-intr-inputs = <0x20>;
      interrupt-parent = <&gic>;
      interrupts = <0 89 4>;
};
```

Step 2: Update Software Environment

Build from Scratch

- Create PetaLinux Project from XSA
  - `petalinux-create -t project --template zynqMP`
  - `petalinux-config --get-hw-description=<XSA_DIR>`
- Customize Kernel and RFS
  - `petalinux-config -c kernel`
  - `petalinux-config -c rootfs` -> Add XRT
- Build Kernel, RFS and device-tree
  - `petalinux-build`
- Build SYSROOT
  - `petalinux-build --sdk`
  - `./images/linux/sdk.sh`

Use Common Image

- Download Pre-built Common Image

Kernel Image	Image
Root File System	rootfs.ext4
SYSROOT	sdk.sh

- Update User Device Tree
  - *sw/prebuilt_linux/user_dts/system-user.dtsi*
  - Use Device tree generator to generate device tree from XSA

Step 3. Generate Vitis Platform

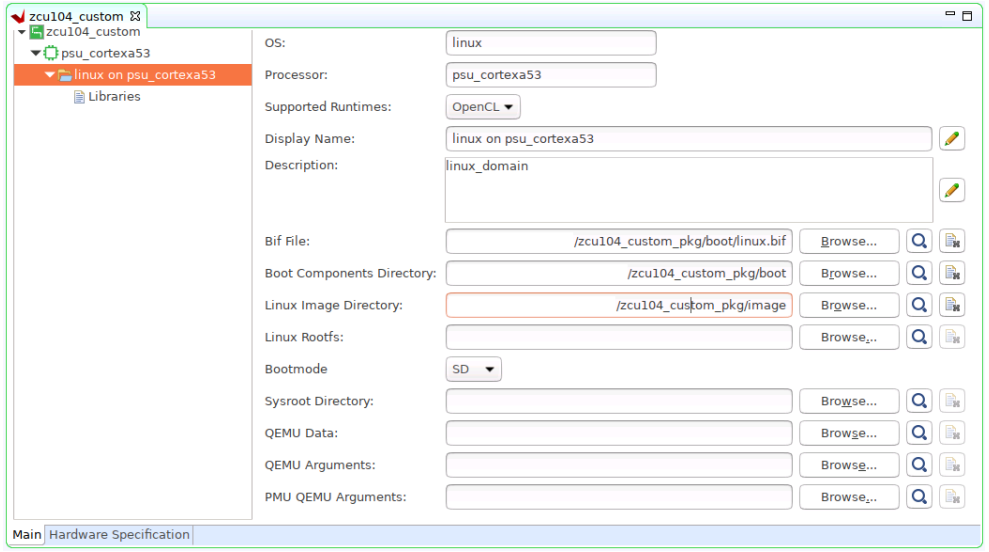
Step 3. Generate Vitis Platform - Prepare Contents

- Hardware: XSA
- Software
  - Boot directory
    - BIF and the components used by BIF
    - fsbl.elf, pmufw.elf, bl31.elf, u-boot.elf
  - SD Card FAT32 directory
    - u-boot.scr
    - Any other custom files
  - Optional Linux components
    - Vitis 2021.1 can generate a default BIF
    - Example BIF

```
/* linux */
 the_ROM_image:
 {
          [fsbl_config] a53_x64
          [bootloader] <fsbl.elf>
          [pmufw_image] <pmufw.elf>
          [destination_device=pl] <bitstream>
          [destination_cpu=a53-0, exception_level=el-3, trustzone] <bl31.elf>
          [destination_cpu=a53-0, exception_level=el-2] <u-boot.elf>
 }
```

Step 3. Generate Vitis Platform

- Vitis GUI
  - New -> Platform Project
  - Use XSA that was exported in step 1
  - Create Linux domain
  - Setup BIF, boot dir and image dir
  - Generate platform by clicking build icon



- XSCT CLI

```
platform -name $platform_name -hw
$xsa_path/$platform_name.xsa -out ./$OUTPUT -no-boot-bsp

domain -name xrt -proc psu_cortexa53 -os linux -image
$SW_COMP/src/a53/xrt/image
domain config -boot $SW_COMP/src/boot
domain config -bif $SW_COMP/src/a53/xrt/linux.bif
domain -runtime opencl

platform -generate
```


XILINX.

Step 4: Verify the Platform

Step 4: Verify the Platform

- Check platforminfo report
  - `platforminfo <Platform_NAME>.xpfm`
  - Check clock information, memory information are reported as expected

```
==================
Clock Information
==================
  Default Clock Index: 0
  Clock Index:         2
    Frequency:         100.000000
  Clock Index:         0
    Frequency:         200.000000
  Clock Index:         1
    Frequency:         400.000000

==================
Memory Information
==================
  Bus SP Tag: HP0
  Bus SP Tag: HP1
  Bus SP Tag: HP2
  Bus SP Tag: HP3
  Bus SP Tag: HPC0
  Bus SP Tag: HPC1
```

- Run ***Vector Addition*** example on this platform
  - It can be in the same workspace if the platform is created in Vitis GUI.
  - Set *PLATFORM_REPO_PATHS* environment variable to allow Vitis to get the platform if not working on the same workspace or working with command line flow.

Summary

Review the Platform Creation Flow

Reference

- UG1416: Vitis Unified Software Platform User Guide
  - https://www.xilinx.com/html_docs/xilinx2021_1/vitis_doc/
  - Creating Embedded Platforms in Vitis
  - Using Embedded Platforms
- Xilinx Platform Source Code
  - https://github.com/Xilinx/Vitis_Embedded_Platform_Source
- Vitis Platform Creation Tutorial
  - https://github.com/Xilinx/Vitis-Tutorial/tree/master/Vitis_Platform_Creation

Thank You