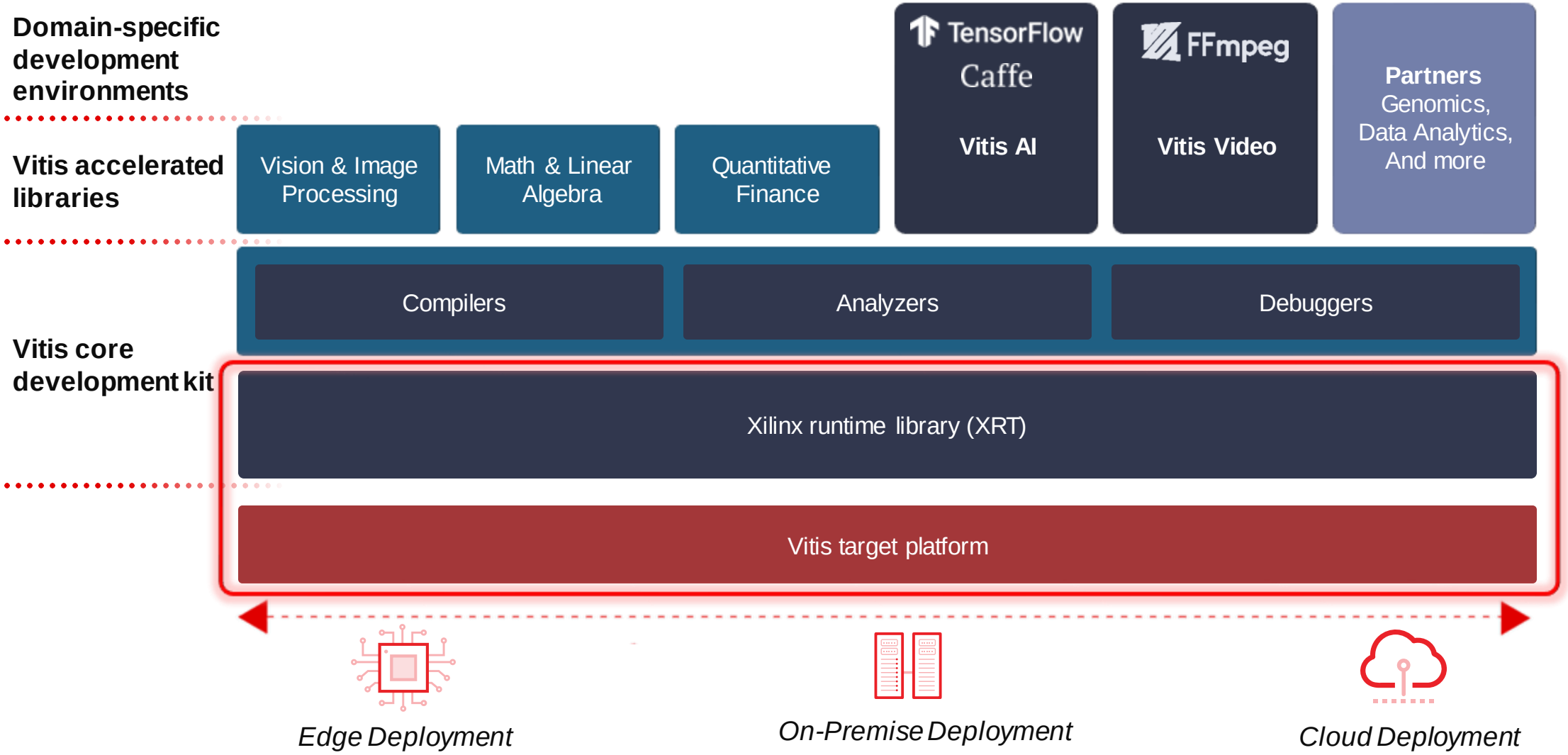
Demystify Vitis Embedded Acceleration Platform Creation

Version: 2020.2

Dec. 2020

Xilinx Adapt Conf

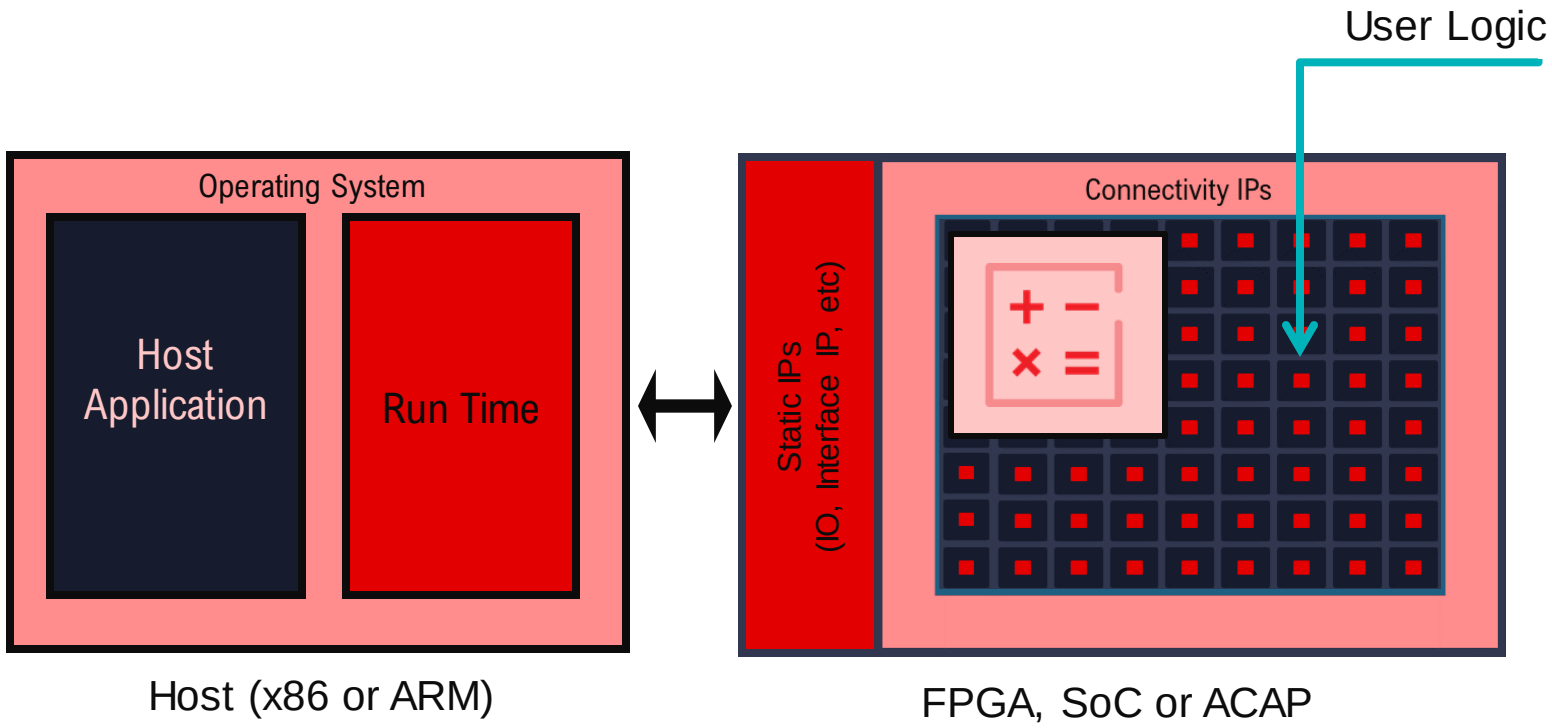
Vitis Unified Software Platform

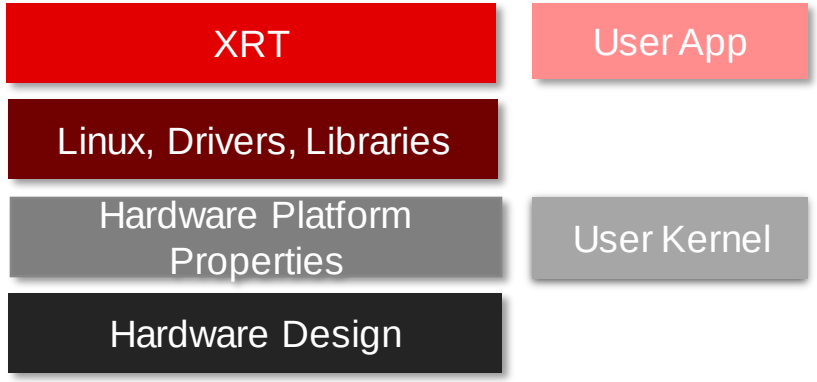


XILINX

Vitis Target Platform

Base Hardware, Software Architecture





XILINX

Do I Need to Create My Own Platform?

- Begin with a Xilinx pre-configured platform For evaluation or PoC.
- Customize platforms when need advanced features or production.

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 - Add VCU, adjust DDR config - Design Final Product	Xilinx Demo Board Custom Board	Custom Platforms

XILINX

Use a Pre-built Platform for Evaluation

Download Pre-built Vitis Embedded Platforms

- www.xilinx.com/download

Vivado (HW Developer) | Vitis (SW Developer) | Vitis Embedded Platforms | PetaLinux

ZCU102 Base 2020.1 (ZIP - 208.01 MB)

MD5 SUM Value : 20618aa28663fe0e528c72146483cdee

ZCU102 Base DFX 2020.1 (ZIP - 220.84 MB)

MD5 SUM Value : 2642571f5d406348425a3c08b78e3f11

ZYNQMP common image (TAR/GZIP - 977.56 MB)

MD5 SUM Value : 8bcf744c9c998f1ec3f9edb7cfd810

ZYNQMP common target licenses and sources (TA

MD5 SUM Value : 6adb9c8f24d3e8c2897a0f5c65656

ZYNQ Third Party Sources (ZIP - 201.83 MB)

MD5 SUM Value : 70ec8188c7a38a7d272f04b047bb8927

ZYNQ common target licenses and sources (TAR/GZIP - 2.29 GB)

MD5 SUM Value : 9a6b678c5ced8101af49f31ff4118a16

ZYNQMP Third Party Sources (ZIP - 230.95 MB)

MD5 SUM Value : b0621ff1cad9c5e54525dc27fc48150d

Base Platform

- Static HW platform

DFX Platform

- Dynamic HW platform

Common Image

- Kernel
- RootFS
- SYSROOT

3rd Party Sources

- Source files used to generate RootFS and SYSROOT
- Mainly for reference. They are not needed to use the platform.

XILINX

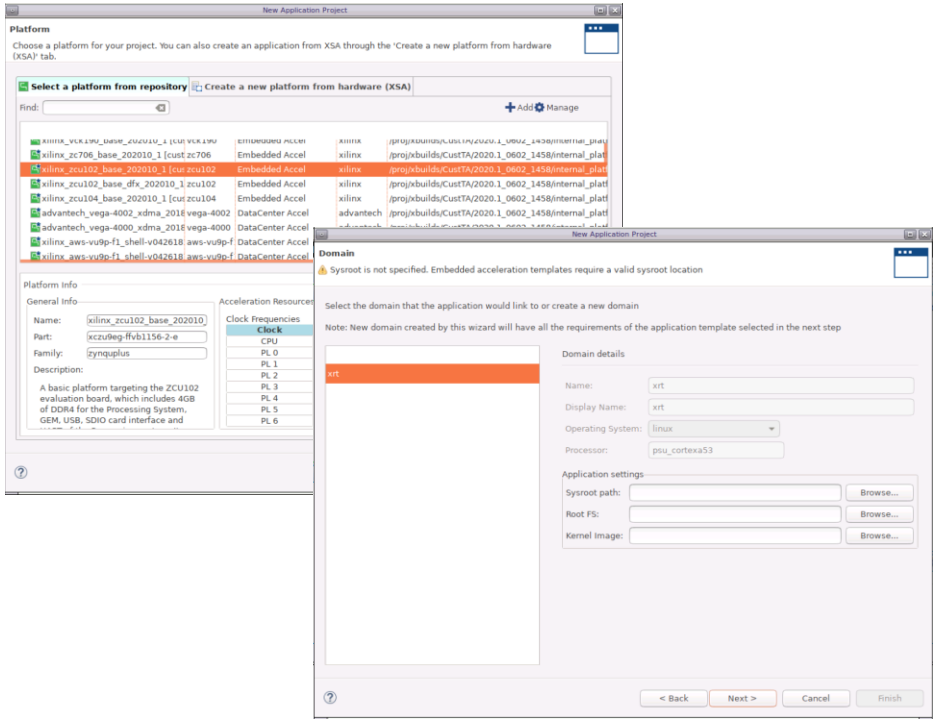
Install Platforms

- Install Platform
  - Extract to platform search path
- Platform Search Path
  - */opt/xilinx/platforms*
  - *$XILINX_VITIS/platforms*
  - *$PLATFORM_REPO_PATHS*
- Extract Common Images
- Install SYSROOT
  - extract sdk.sh
  - Point SYSROOT parameter in Vitis application to the extracted directories
    aarch64-xilinx-linux
    cortexa9t2hf-neon-xilinx-linux-gnueabi

XILINX

Use a Pre-Built Platform

- GUI



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

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/xilinx2020_1/vitis_doc/vitiscommandcompiler.html

XILINX

Install Additional Software with Package Feed

- What is a package feed?
  - Install software packages on-the-fly
  - Like *apt* for Ubuntu, *yum* 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?
  - *dnf* is pre-installed in rootfs of common images
  - Set up package feed URL

```
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, versal, etc.
  - Install packages like using apt/yum

```
dnf install <package name>
```


XILINX

Create Custom Embedded Acceleration Platforms

High Level Workflow for Platform Creation

Step 0: Base Bootable Design

Platform creation preparation

Boot testing

Peripheral function testing

Various ways to create this design

For Xilinx boards, Vivado preset and PetaLinux BSP can be used

For custom board, please setup pinout and PS settings according to your board

XILINX

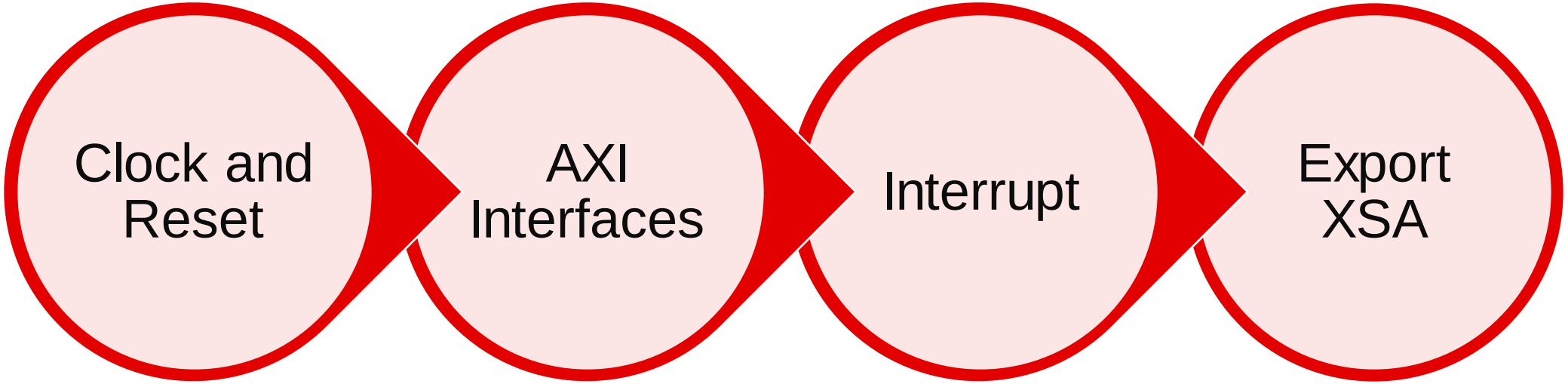
Step 0: Base Bootable Design

- Plan which components should be in your platform

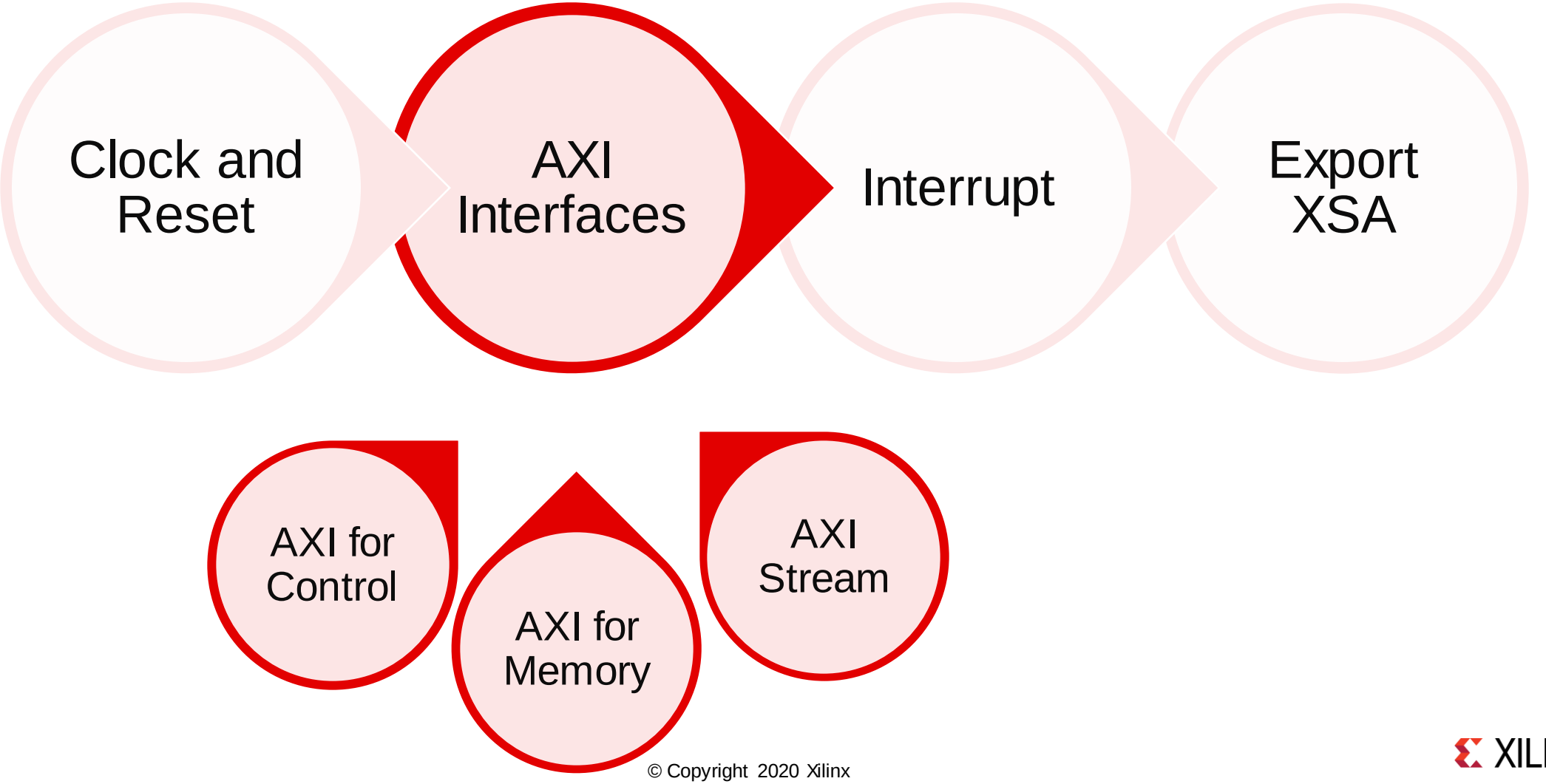
Platform	Kernel
Interface IO (Clock, GPIO)	Memory mapped acceleration kernels
Interface IP that needs system driver (EMAC, MIPI)	Streaming interface acceleration kernels
ARM Processors	Free-running kernels
Non-AXI Interface IP	

XILINX

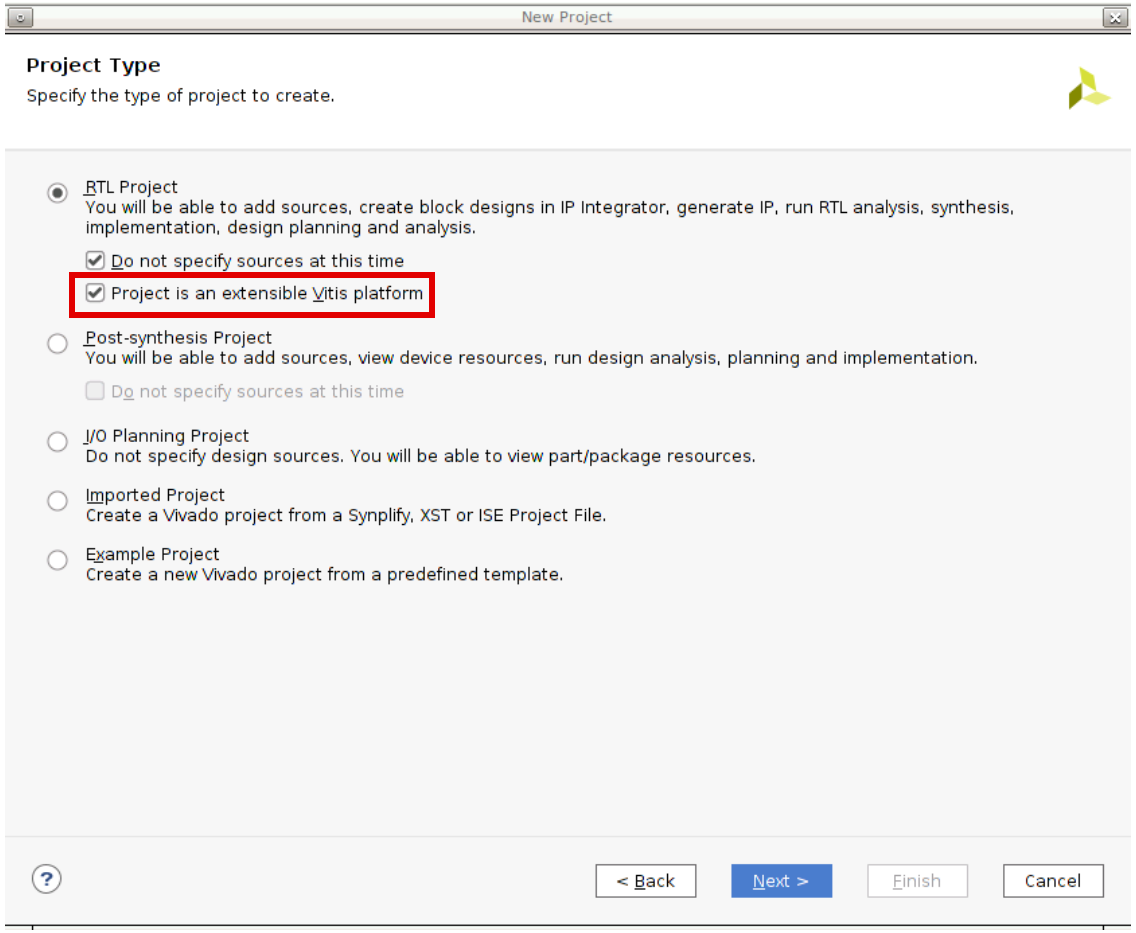
Step 1: Prepare Hardware Design in Vivado

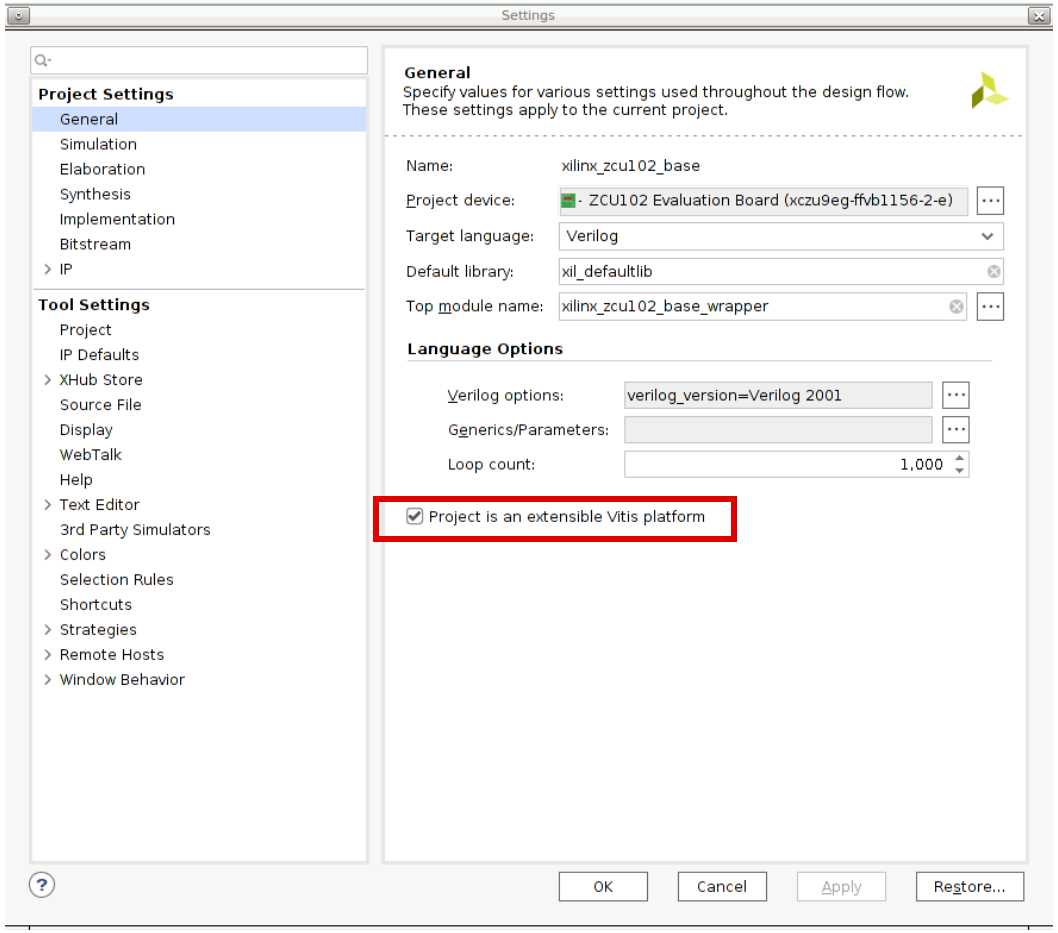


Step 1: Prepare Hardware Design in Vivado



A. Mark Project as Extensible Platform Project



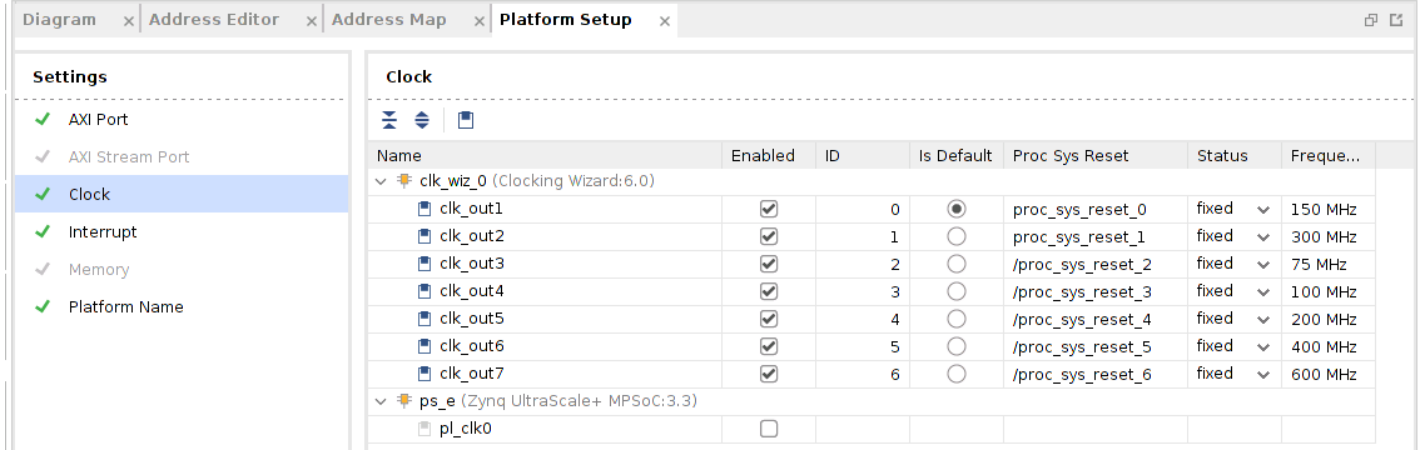


B. Clock and Reset Settings

- General Rules
  - Each clock needs one associated reset signal synchronous with this clock
  - Each platform must have one and only one default clock
- CLI

```
set_property PFM.CLOCK
{clk_out1 {id "0" is_default "true"
proc_sys_reset "/proc_sys_reset_0" status
"fixed"}
clk_out2 {id "1" is_default "false"
proc_sys_reset "/proc_sys_reset_1" status
"fixed"}
...} [get_bd_cells /clk_wiz_0]
```

- Vivado GUI
  - Window -> Platform Setup
  - Enable the clock signals



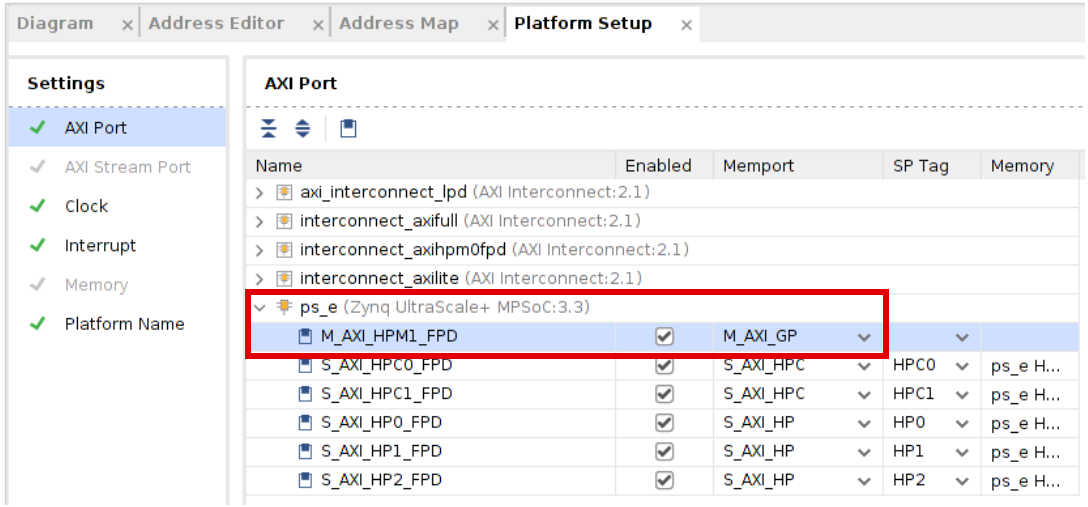
XILINX

B. AXI Interfaces for Kernel Control

- General Rules
  - Needed for AXI-MM kernel control
  - It can be PS AXI Master port
  - It can be master port of Interconnect or Smartconnect
- Vivado GUI
  - Window -> Platform Setup
  - Enable the Interfaces
  - Sptag for doesn't take effect
- CLI

```
set_property PFM.AXI_PORT \
{M_AXI_HPM1_FPD {memport "M_AXI_GP" }} [get_bd_cells
/zynq_ultra_ps_e_0]
```

- GUI



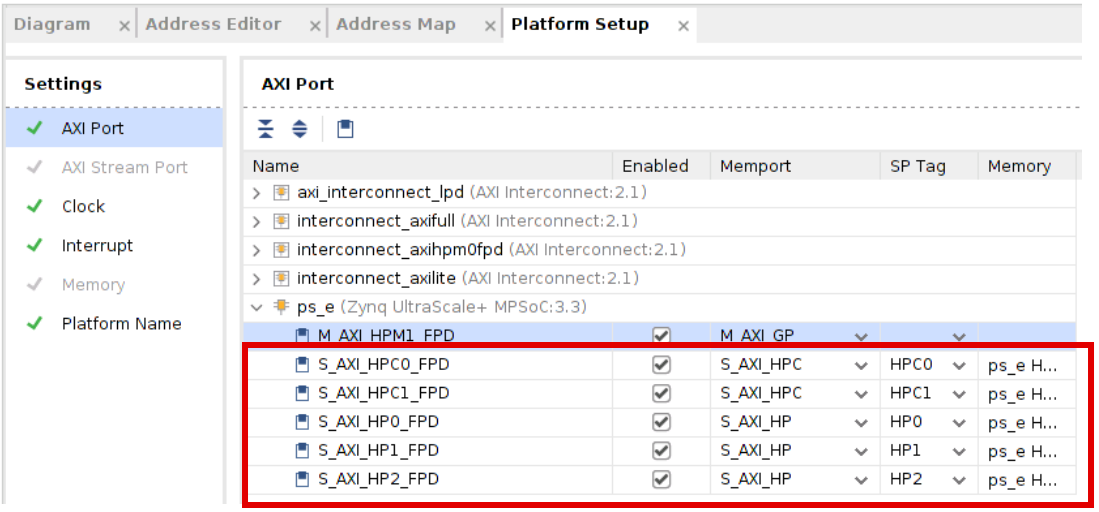
C. AXI Interfaces for Memory Access

- General Rules
  - A platform needs to define one or more memory interface for memory-mapped kernel to access DDR memory
- Vivado GUI
  - Window -> Platform Interfaces
  - Enable the Interfaces
  - Set sptag for Interface name (optional)
    - A symbolic identifier that represents a class of platform port connections
    - Multiple block design platform ports can share the same sptag
    - Used by v++ link.
  - Set memory for memory subsystem identifier(optional)
    - Cell name and Base Name columns in the IP integrator Address Editor

- CLI

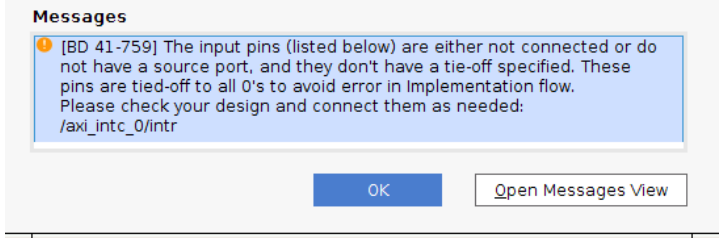
```
set_property PFM.AXI_PORT \
S_AXI_HP0_FPD {memport "S_AXI_HP" sptag "HP0" memory "ps_e
HP0_DDR_LOW"} [get_bd_cells /zynq_ultra_ps_e_0]
```

- GUI



D. Interrupt Settings

- General Rules
  - A platform needs to define how kernel interrupt signal can be connected.
  - AXI Interrupt Controller is needed for v++ linker to link interrupt signals automatically.
- Note
  - Safe to ignore *intr* floating critical warning because v++ linker will make connections.

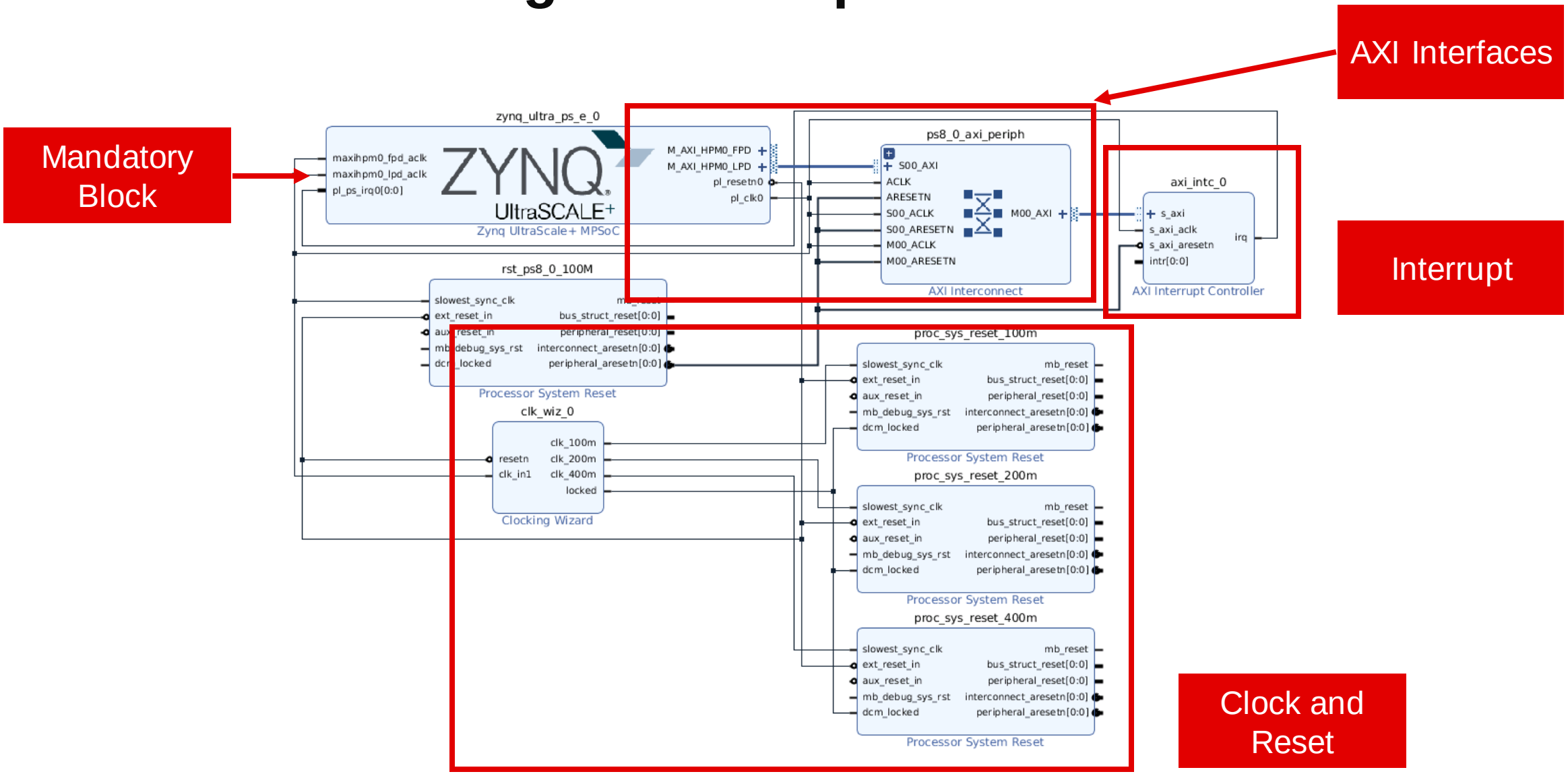


- Vivado GUI

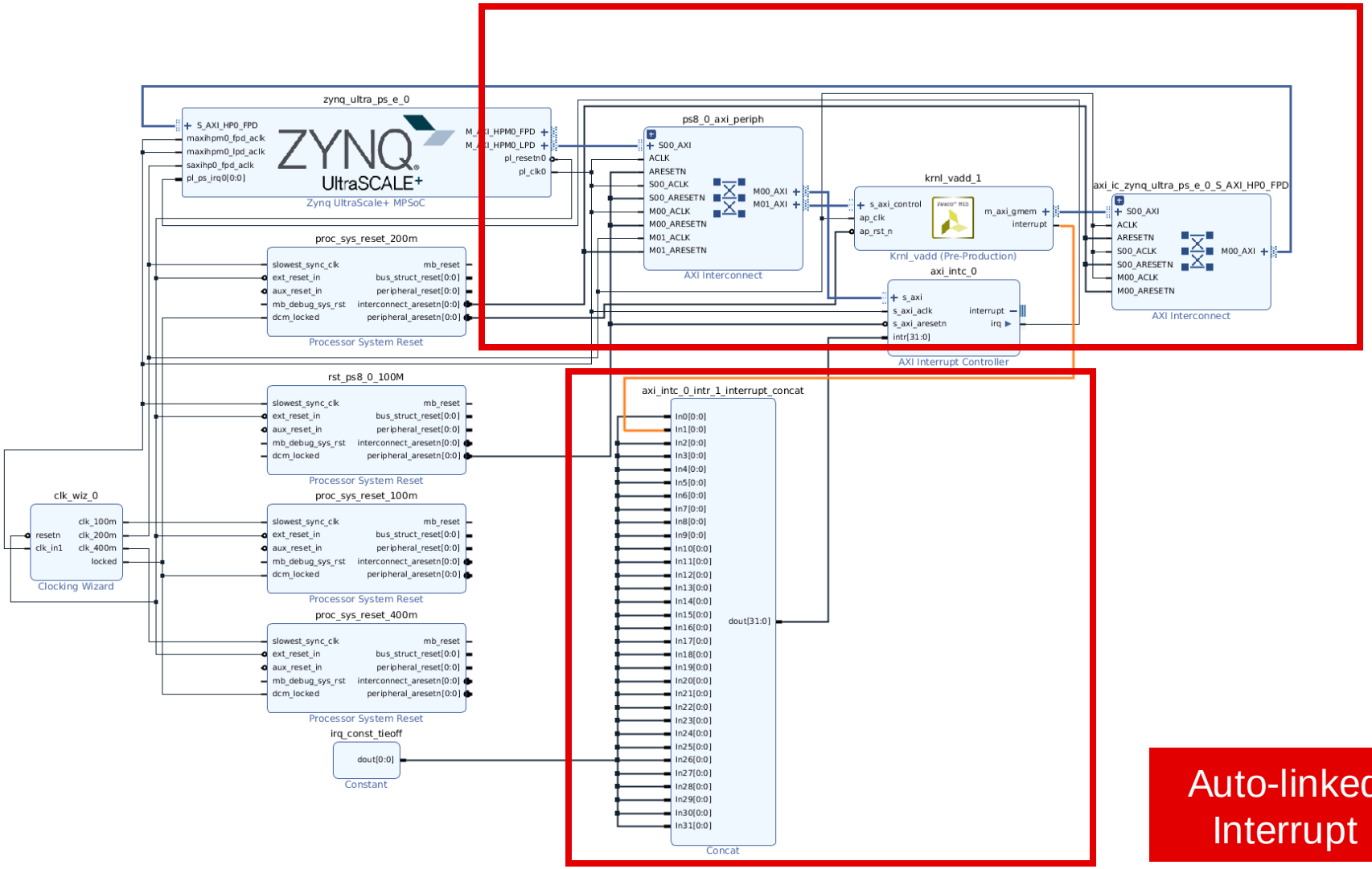
- CLI

```
set_property PFM.IRQ {intr {id 0 range 32}} [get_bd_cells
/axi_intc_0]
```

Platform Block Diagram Example



Vitis Linked Vector Addition Block Diagram Example

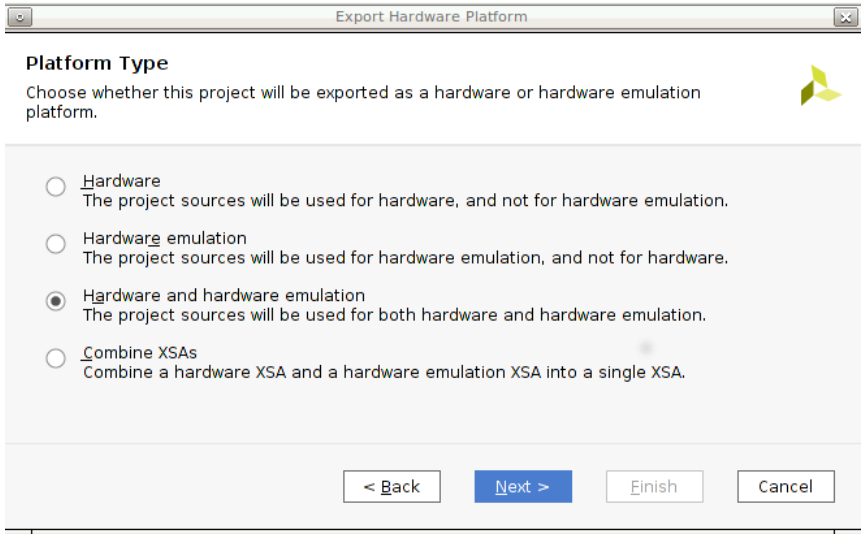


Auto-linked kernel

Auto-linked Interrupt

E. Export XSA

- 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



- CLI

```
# Setup Platform Name
set_property PFM_NAME {xilinx:zcu102:zcu102_base:1.0}
[get_files [current_bd_design].bd]

# Generate block design and optionally implement the design

# Export Acceleration Platform
write_hw_platform ./zcu102_base.xsa

# Validate Platform
validate_hw_platform ./zcu102_base.xsa
```

Step 2: Prepare Software Environment

- Supported Software Environments

Software Environment	Supported?
Linux	Yes
Standalone	No
RTOS	Roadmap

- Linux Components Requirements

Linux Component	Requirements
Kernel Image	-
Root FS	xrt zocl } packagegroup-petalinux-xrt (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>;
};
```


XILINX

Step 2: Prepare Software Environment

Build from Scratch

- Create PetaLinux Project from XSA
  - `petalinux-create -t project --template zynqMP`
  - `petalinux-config --get-hw-description=<XSA_DIR>`
- Update Device Tree
  - *project-spec/meta-user/recipes-bsp/device-tree/files/system-user.dtsi*
- Customize Kernel and RFS
  - `petalinux-config -c kernel`
  - `petalinux-config -c rootfs`
- Build Kernel, RFS and device-tree
  - `petalinux-build`
- Build SYSROOT
  - `petalinux-build --sdk`
  - `./images/linux/sdk.sh`

UG1144

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*
- Generate Platform with prebuilt_linux mode
  - New in 2020.2 base platforms
  - Define `COMMON_RFS_KRNL_SYSROOT`
  - Generate Platform `make all`

UG1393

XILINX

Enable Package Feed in RootFS

- Install dnf to rootfs
  - `petalinux-config -c rootfs`
  - `[*]Image Feature -> Package Management`

- Add feed URL
  - On Board Preparation

```
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
```

  - Install packages like using apt/yum
    - *dnf install <package name>*

XILINX

Step 3. Create Vitis Platform - Prepare Contents

- Boot directory
  - BIF and the components used by BIF
    - fsbl.elf
    - pmufw.elf
    - bl31.elf
    - u-boot.elf
- BIF example

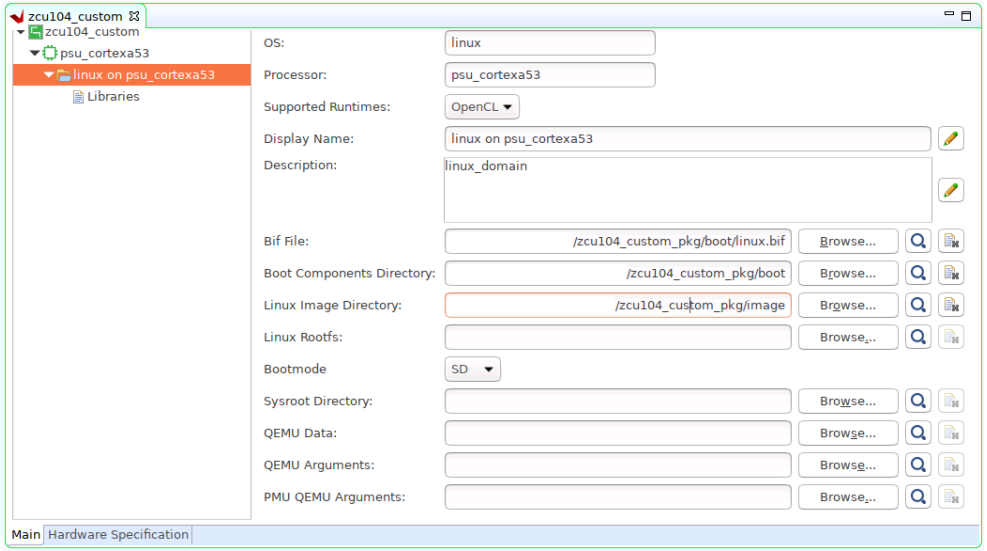
```
/* linux */
 the_ROM_image:
 {
          [fsbl_config] a53_x64
          [bootloader] <fsbl.elf>
          [pmufw_image] <pmufw.elf>
          [destination_device=pl] <bitstream>
          [destination_cpu=a53-0, exception_level=el-3, trustzone] <bl31.elf>
          [destination_cpu=a53-0, exception_level=el-2]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

- Image directory
  - Vitis packager will add all files in image directory to fat32 partition of SD card

XILINX

Step 3. Create 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
```

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.

XILINX

Summary

High Level Workflow for Platform Creation

Reference

- UG1416: Vitis Unified Software Platform User Guide
  - https://www.xilinx.com/html_docs/xilinx2020_2/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-In-Depth-Tutorial/tree/master/Vitis_Platform_Creation

XILINX

Happy Vitising