

Demystify Vitis Embedded Acceleration Platform Creation Version: 2020.2

Dec. 2020 Xilinx Adapt Conf



Vitis Unified Software Platform



Vitis Target Platform

Base Hardware, Software Architecture



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-Al 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



Use a Pre-built Platform for Evaluation



Download Pre-built Vitis Embedded Platforms

www.xilinx.com/download



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



Use a Pre-Built Platform

GUI

	New Application	Project		
form			•••	
ose a platform for your project. You can al A)' tab.	o create an application from	NXSA through th	he 'Create a new platform from hardware	
Select a platform from repository	Create a new platform fr	om hardware	(XSA)	
nd:			🕂 Add 🏟 Manage	
xiinx_vckiau_base_202010_1 [cui vck	290 Embedded Accel D6 Embedded Accel	xiiinx	лргојлкранову, чих гиу сиси. 1. чови 2. 1430/mternal_pray /proj/xbuilds/CustTA/2020.1_0602_1458/internal_plat	
kilinx_zcu102_base_202010_1 [cu: zcu	102 Embedded Accel	xilinx	/proj/xbuilds/CustTA/2020.1_0602_1458/internal_plat	
xilinx_zcu102_base_dfx_202010_1 zcu	L02 Embedded Accel	xilinx	/proj/xbuilds/CustTA/2020.1_0602_1458/internal_platf	
xilinx_zcu104_base_202010_1 [cus zcu	L04 Embedded Accel	xilinx	/proj/xbuilds/CustTA/2020.1_0602_1458/internal_plati	
advantech_vega-4002_xdma_2018 veg	a-4002 DataCenter Accel	advantech	/proj/xbuilds/custin/2020.1_0602_1458/internal_platt	
silinx aws-vu9p-f1 shell-v042618 aws	-vu9p-f DataCenter Accel	•	New Application Project	
xilinx aws-vu9p-f1 shell-v042618 aws	-vu9p-f DataCenter Accel	Domain		1
		🔒 Sysroot is no	at specified. Embedded acceleration templates require a valid sysroot location	
atform Info				
eneral Info	Acceleration Resources	Select the dom	main that the application would link to or create a new domain	
Name: xilinx_zcu102_base_202010	Clock Frequencies			
Part: xczu9eg-ffvb1156-2-e	Clock	Note: New dor	main created by this wizard will have all the requirements of the application template selected in the next step	
Family: zynguplus	PLO		Domain details	
Description:	PL 1			
A basic platform targeting the ZCU102	PL 2	xit	Name: xrt	
evaluation board, which includes 4GB	PL 4			
of DDR4 for the Processing System, GEM_USB_SDIO card interface and	PL 5		Display Name: xrt	
Course for a	PEO		Operating System: linux 💌	
			Processor: psu_cortexa53	
			Application settings	
			Sysroot path:	Browse.
			Root FS:	Browse.
			Kernel Image:	Browse.
		0	< Back Next > Cancel	Fini

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



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 <u>http://petalinux.xilinx.com</u>
- 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>



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

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	



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]
```

Diagram x Address Editor x Address Map x Platform Setup σĽ Clock Settings AXI Port ¥ ♦ AXI Stream Port Name Enabled ID Is Default Proc Sys Reset Status Freque.. v # clk wiz 0 (Clocking Wizard:6.0) Clock 📑 clk out1 ✓ proc_sys_reset_0 fixed \sim 150 MHz Interrupt clk_out2 proc sys reset 1 300 MHz clk out3 \bigcirc 2 /proc sys reset 2 75 MHz fixed Memory clk out4 3 /proc sys reset 3 100 MHz fixed Platform Name clk out5 4 /proc sys reset 4 200 MHz clk out6 /proc sys reset 5 400 MHz clk out7 6 /proc sys reset 6 600 MHz fixed # ps_e (Zynq UltraScale+ MPSoC:3.3) pl clk0

Vivado GUI

- Window -> Platform Setup
- Enable the clock signals

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

Settings	AXI Port						
 AXI Port 							
AXI Stream Port	Name	Enabled	Memport		SP Tag		Memo
	> 🖲 axi_interconnect_lpd (AXI Inte	erconnect:2.1)					
CIOCK	> 💽 interconnect_axifull (AXI Interconnect:2.1)						
Interrupt	> 📧 interconnect_axihpm0fpd (AXI Interconnect:2.1)						
/ Memory	> 💽 interconnect_axilite (AXI Inter	connect:2.1)					
	✓ ≢ ps_e (Zynq UltraScale+ MPSc	oC:3.3)					
 Platform Name 	M_AXI_HPM1_FPD		M_AXI_GP	~		~	
	S_AXI_HPC0_FPD	\checkmark	S_AXI_HPC	~	HPC0	\mathbf{v}	ps_e H
	S_AXI_HPC1_FPD	\checkmark	S_AXI_HPC	~	HPC1	\sim	ps_e H
	S_AXI_HP0_FPD	\checkmark	S_AXI_HP	~	HP0	\checkmark	ps_e H
	S_AXI_HP1_FPD		S_AXI_HP	~	HP1	\sim	ps_e H
	S AXI HP2 FPD		S AXI HP	~	HP2	~	ns e F

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

Diagram x Address Editor x Address Map x Platform Setup x									
Settings AXI Port									
v 1	AXI Port	¥ ♦ 🔳							
1	AXI Stream Port	Name	Enabled	Memport		SP Tag		Memory	
1	Clock	> 📧 axi_interconnect_lpd (AXI Interconnect	2.1)						
		> 📳 interconnect_axifull (AXI Interconnect:2.1)							
 ✓ 1 	Interrupt	> 📧 interconnect_axihpm0fpd (AXI Interconnect:2.1)							
1	Memory	> 💌 interconnect_axilite (AXI Interconnect:	2.1)						
		✓ ₱ ps_e (Zynq UltraScale+ MPSoC:3.3)							
~ 1	Platform Name	🖻 m axi hpm1 fpd		M AXI GP	~		\mathbf{v}		
		S_AXI_HPC0_FPD		S_AXI_HPC	~	HPC0	~	ps_e H	
		S_AXI_HPC1_FPD		S_AXI_HPC	~	HPC1	\sim	ps_e H	
		S_AXI_HP0_FPD		S_AXI_HP	~	HP0	\sim	ps_e H	
		S_AXI_HP1_FPD		S_AXI_HP	~	HP1	\sim	ps_e H	
		S_AXI_HP2_FPD		S_AXI_HP	~	HP2	¥	ps_e H	
		 S_AXI_HPC0_FPD S_AXI_HPC1_FPD S_AXI_HP0_FPD S_AXI_HP1_FPD S_AXI_HP2_FPD 	> > > >	S_AXI_HPC S_AXI_HPC S_AXI_HP S_AXI_HP S_AXI_HP	* * * *	HPC0 HPC1 HP0 HP1 HP2	* * * *	ps_eH ps_eH ps_eH ps_eH ps_eH	

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

Diag	jram 🗙 Address E	ditor x Address Map x Platform Se	etup ×
Set	ttings	Interrupt	
~	AXI Port	¥ ♦ I	
~	AXI Stream Port	Name	Enabled
1	Clock	✓ ₱ axi_intc_0 (AXI Interrupt Controller:4.1)	
-	SIDER	📄 intr	
~	Interrupt		

CLI

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



Platform Block Diagram Example AXI Interfaces zyng ultra ps e 0 ps8 0 axi periph Mandatory M AXI HPM0 FPD + + S00_AXI maxihpm0_fpd_aclk M AXI HPM0 LPD + maxihpm0_lpd_aclk ACLK axi intc 0 pl_resetn0 o Block pl_ps_irq0[0:0] pl_clk0 ARESETN UltraSCALE⁻ S00_ACLK ■ ← ■ M00_AXI + -= + s_axi Zyng UltraScale+ MPSo S00 ARESETN ∎Ă∎ s axi aclk irq M00_ACLK s_axi_aresetn M00_ARESETN intr[0:0] Interrupt rst ps8 0 100M AXI Interconnect AXI Interrupt Controller slowest_sync_clk bus_struct_reset[0:0] ext_reset_in aux reset in peripheral reset(0:0) proc_sys_reset_100m debug_sys_rst interconnect_aresetn[0:0] 🍈 locked peripheral aresetn[0:0] dcr slowest_sync_clk mb_reset bus_struct_reset[0:0] ext_reset_in Processor System Reset peripheral_reset[0:0] aux_reset_in clk wiz 0 interconnect_aresetn[0:0] mb_debug_sys_rst dcm_locked peripheral_aresetn[0:0] 🕒 clk_100m Processor System Reset clk_200m resetn clk in1 clk 400m proc sys reset 200m locked slowest sync clk mb reset Clocking Wizard bus_struct_reset[0:0] ext reset in peripheral_reset[0:0] aux reset in interconnect_aresetn[0:0] 🕩 mb_debug_sys_rst dcm_locked peripheral_aresetn[0:0] Processor System Reset proc sys reset 400m slowest_sync_clk mb_reset ext_reset_in bus_struct_reset[0:0] aux reset in peripheral_reset[0:0] **Clock and** mb_debug_sys_rst interconnect_aresetn[0:0] dcm_locked peripheral_aresetn[0:0] Reset Processor System Reset



Vitis Linked Vector Addition Block Diagram Example



EXILINX.

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 = \langle 0x0 \rangle;
      xlnx,num-intr-inputs = <0x20>;
      interrupt-parent = <&gic>;
      interrupts = <0 89 4>;
};
```

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/systemuser.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



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

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>

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



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



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

Dosu cortexa53	OS:	linux				
▼ Talinux on psu_cortexa53	Processor:	psu_cortexa53				
📄 Libraries	Supported Runtimes:	OpenCL 🔻				
	Display Name:	linux on psu_cortexa53				
	Description:	linux_domain				
						1
	Bif File:	/zcu104_custo	m_pkg/boot/linux.bif	<u>B</u> rowse	Q	10
	Boot Components Directory:	/zcul	04_custom_pkg/boot	B <u>r</u> owse	Q	
	Linux Image Directory:	/zcul04	L_custom_pkg/image	Br <u>o</u> wse	Q	
	Linux Rootfs:			Browse	Q	
	Bootmode	SD 🔻				
	Sysroot Directory:			Bro <u>w</u> se	Q	
	QEMU Data:			Browse	Q	
	QEMU Arguments:			Brows <u>e</u>	Q	
	PMU QEMU Arguments:			Browse	Q.	

► 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.





Summary

E XILINX.

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





Happy Vitising

