PYNQ-ZU Bare Metal SD Workflow

Last Updates: June 2025

E-Elements Technology Co., Ltd.



Table of Contents

Tal	ole of Cor	ntents		1
1.	Overview			2
2.	Environment Setup			3
	2.1.	Prerequisites		3
	2.2.	Equipment		3
3.	Vitis Al Runtime (VART) for Edge		錯誤! 尚未定義書館	籖。
	3.1.	VART - Setup the Host	錯誤! 尚未定義書館	籖。
	3.2.	VART - Setup the Target		5
	3.3.	Running VART Examples	錯誤! 尚未定義書館	籖。
	3.4.	Running VART Examples	錯誤! 尚未定義書鏡	籖。
4.	VART Samples for Edge		錯誤! 尚未定義書籍	籖。
	4.1.	resnet-50	錯誤! 尚未定義書館	籖。
	4.2.	resnet50_mt_py	錯誤! 尚未定義書館	籖。
	4.3.	inception_v1_mt_py	錯誤! 尚未定義書館	籖。
	4.4.	pose_detection	錯誤! 尚未定義書館	籖。
	4.5.	video_analysis	錯誤! 尚未定義書館	籖。
	4.6.	adas_detection	錯誤! 尚未定義書館	籖。
	4.7.	segmentation	錯誤! 尚未定義書館	籖。
	4.8.	squeezenet_pytorch	錯誤! 尚未定義書館	籤。
5.	Running Examples in Vitis Al Library		錯誤! 尚未定義書館	籖。
	5.1.	Running Vitis Al Library Samples	錯誤! 尚未定義書館	籖。
	5.2.	Running Vitis Al Library Apps	錯誤! 尚未定義書館	籖。
6.	Reference		錯誤! 尚未定義書館	籖。
	6.1.	Xilinx Zynq MPsoC	錯誤! 尚未定義書館	籖。
	6.2.	Utilities	錯誤! 尚未定義書釒	籖。

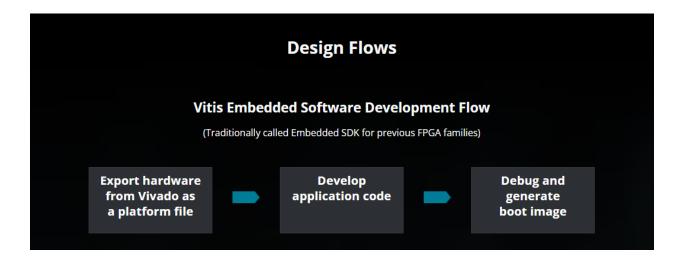




1. Overview

The PYNQ-ZU is not only a Python-based prototyping platform, it also supports full bare-metal development, offering developers low-level control over the Zynq UltraScale+ MPSoC hardware without relying on an operating system.

By targeting the ARM Cortex-A53, Cortex-R5, or Cortex-M4 cores directly, users can implement deterministic, real-time, and highly efficient embedded systems on the PYNQ-ZU board. This is ideal for applications such as motor control, real-time signal processing, or custom embedded firmware development.



Reference: AMD Vitis-Unified Software Platform Website [link]



2. Environment Setup

2.1. Prerequisites

- Install Vitis Unified Platform 2024.2 [link] if Vitis not installed on your machine yet
- Install Serial Terminal we use the MobaXterm here [link]

2.2. Equipment

- One computer
- One PYNQ-ZU
- One micro-USB to USB-A cable
- One SD card



3. Workflow

3.1. Build hardware platform

Build your own FPGA hardware platform

- 1. Open the Vivado Design Suite
- 2. Click "Create Project" > "Next".
- 3. Provide a project name and location for the new project.
- 4. Select "RTL Project" and "Do not specify sources" currently.
- 5. Choose your FPGA Board or the special FPGA part number.
- 6. Finish the new project creating.
- 7. In the Flow Navigator tab, click "Create Block Design".
- 8. Provide the BD name and then click "OK" to create a new one.
- 9. In the Diagram view, click "Add IP" or press (Ctrl + I) to add the "Zyng UltraScale+ MPSoC" IP.
- 10. Click "Run Block Automation" > "OK".
- 11. Connect "pl_clk0" and "maxihpm0_fpd_aclk" and "maxihpm1_fpd_aclk".
- 12. Click "Validate Design" or press "F6".
- 13. Click "OK" after you see Validation successful.
- 14. In the BLOCK DESIGN/Sources/Design Sources tab, right click your BD design and "Create HDL Wrapper".
- 15. In Flow Navigator tab, click "Generate Bitstream".
- 16. After the bitstream file is generated, click File > Export > Export Hardware.
- 17. Select "include bitstream".
- 18. Provide the Xilinx Support Archive file name and location.



3.2. Create a booting image

Create a booting bare metal application from SD card

- 1. Open the Vitis Unified IDE.
- 2. In "VITIS EXPLORER" tab, click "Set Workspace".
- 3. Click Create Platform Component.
- 4. Provide the platform name and location.
- 5. Browse your XSA file created in the previous stage.
- 6. Select "standalone" and "psu_cortexa53_0".
- 7. Open platform/Settings/vitis-comp.json.
- 8. In platform/psu_cortexa53_0/standalone_psu_cortexa53_0/Board Support Package, select "xilffs" library.
- 9. In the Flow tab, build the platform project.
- 10. In the Example tab, open "Hello World" example.
- 11. In the Flow tab, build the Application project (Hello World example).
- 12. Click "Create Boot Image".
- 13. Provide BIF Path and Image path.
- 14. Copy BOOT.bin to your ext32 SD card.

