DPU Integration for Embedded ML

Vivado/Petalinux/SDK

Machine Learning Live

March 12, 2019 | Detroit, MI

Terry O'Neal – ML Specialist FAE – South Central US

XILINX.

Agenda

- Session Overview
- DPU Integration Walkthrough

Session Overview

Introduction

- **The DNNDK Evaluation package is a GREAT tool for evaluating pre-built models – BUT:**
  - It doesn't allow you to use your own hardware
  - You're limited to a pre-built Linux image
  - You can't easily experiment with different DPU sizes and configurations
- **DPU Targeted Reference Design (TRD) Released for ZCU102 at xilinx.com**
  - This is a great introduction to building a custom DPU-based system
  - Vivado->Petalinux->Yocto SDK Application
  - Includes:
    - Pre-release DPU IP v1.3.0
    - Example Vivado project for ZCU102
    - Petalinux BSP including all necessary components needed for the DPU
    - A resnet50 example application

Edge AI Targeted Reference Designs (TRD)

Product	Documentation	Image Download	File Size	MD5 Checksum
DPU TRD	DPU IP Product Guide (PG338)	zcu102-dpu-trd-2018-2-190306.zip	4 MB	c075965f7a391fa0ec15765d6e8ae87e

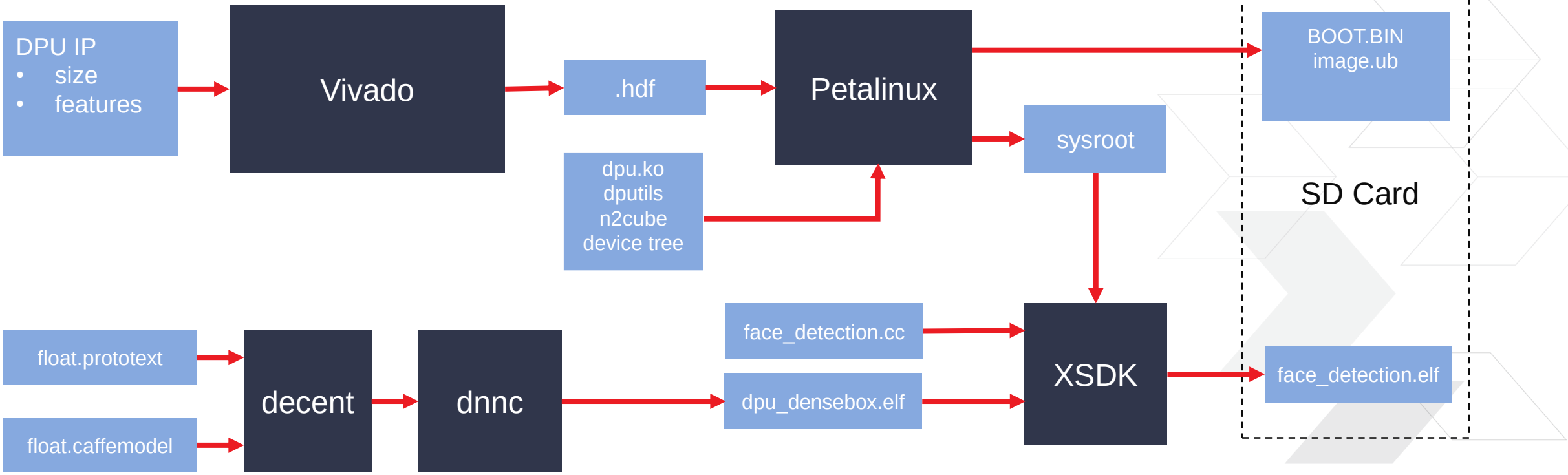
Introduction

- **We'll go through a quick overview of how to create your own design for:**
  - A different evaluation or custom board (Ultra96 Used as example)
  - A custom Vivado project
- **All tools and build steps will be covered**
  - Vivado project entry
  - Petalinux configuration & build
  - DPU application development in Xilinx SDK.
- **This will help you understand what's going on "under the hood" in the TRD.**
- **The full lab will be available soon:**
  - https://github.com/Xilinx/Edge-AI-Platform-Tutorials/

DPU Integration Walkthrough

High-level Tool Flow

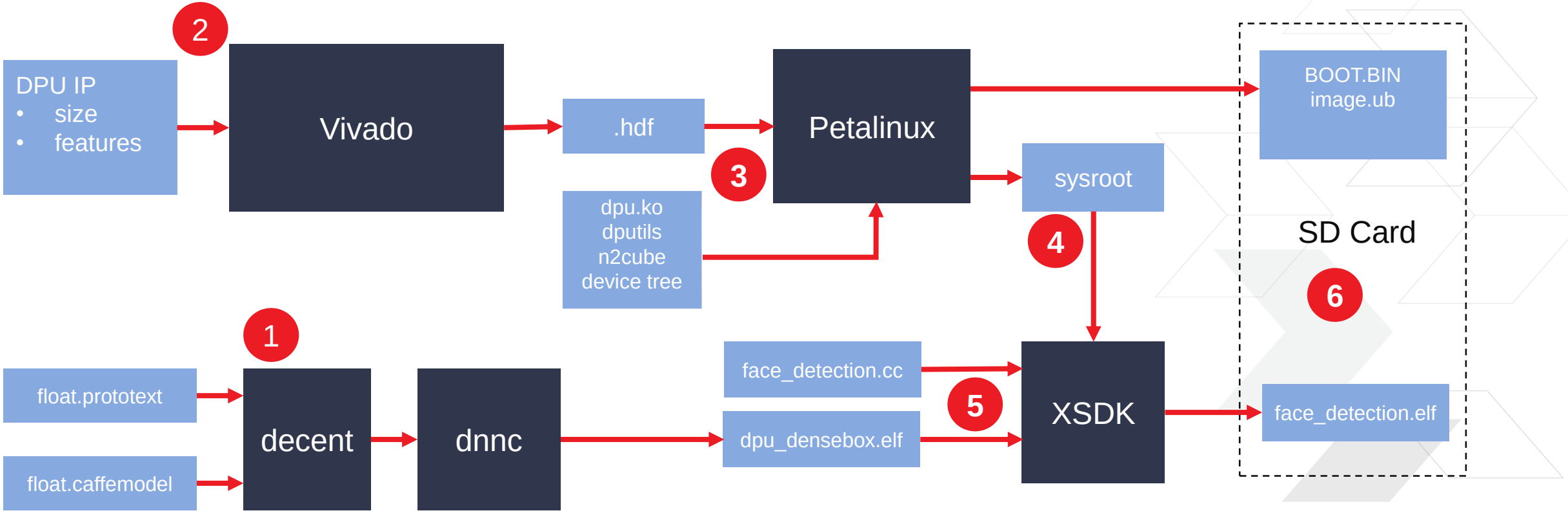
Face Detection Application Example



XILINX.

High-level Tool Flow

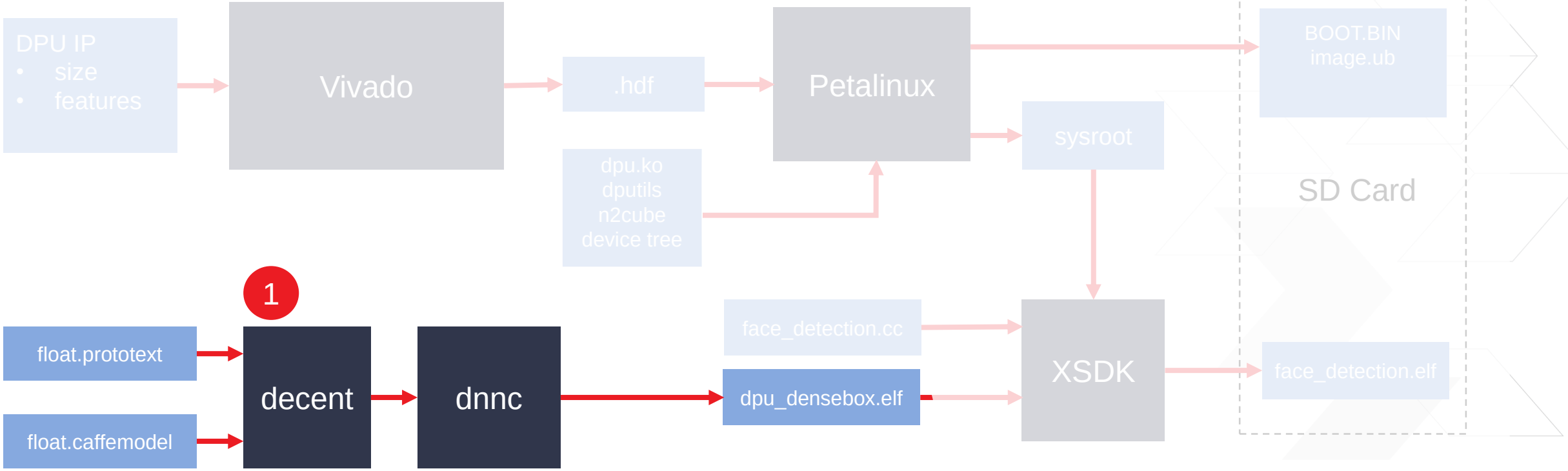
Face Detection Application Example



XILINX.

High-level Tool Flow

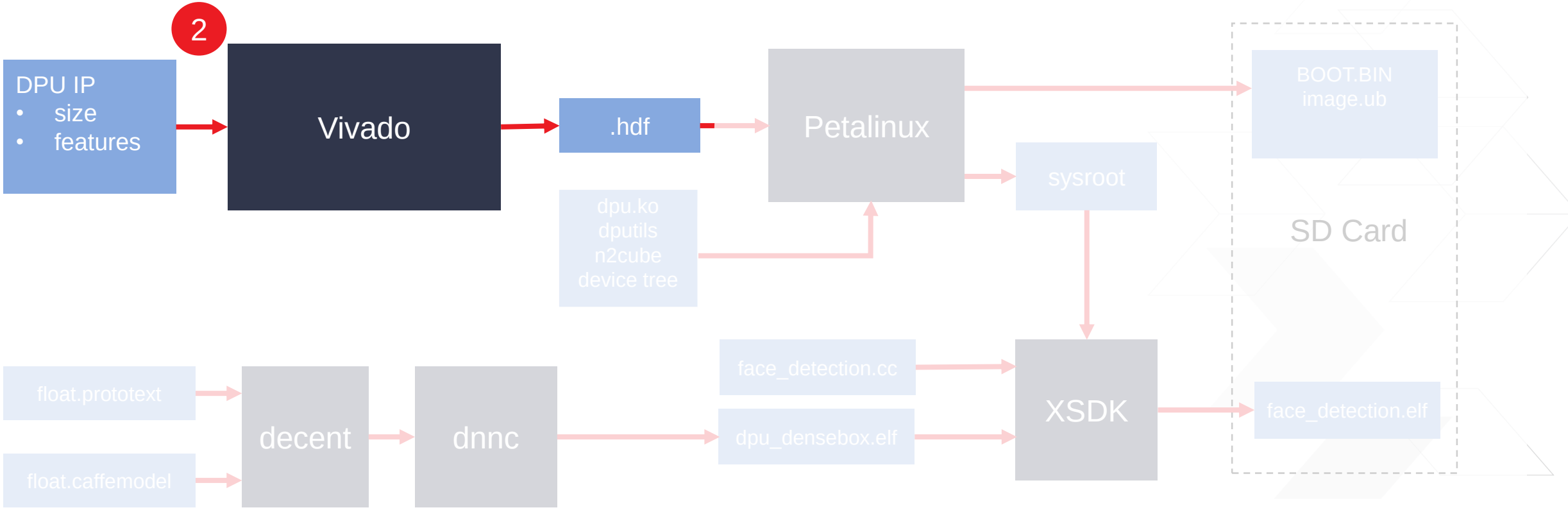
Face Detection Application Example - DNNDK



XILINX.

High-level Tool Flow

Face Detection Application Example - Vivado



XILINX.

Vivado Overview

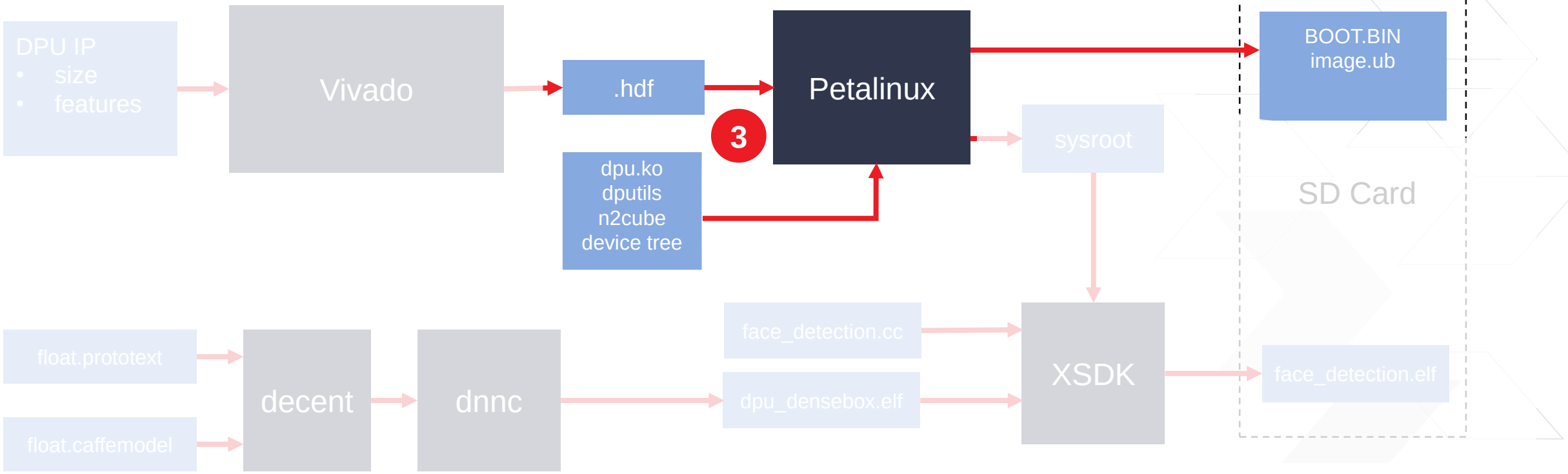
- **Create a new project for the Ultra96**
- **Add the DPU IP to the IP Catalog**
  - Note: This is a v1.3.0 version of the DPU – compatible with v1.3.0 of DNNC
- **Use a .tcl script to hook up the block design in IPI**
- **Examine the DPU configuration and connections**
- **Examine the clocking structure**
- **[Optional] Copy the pre-built .hdf to the Petalinux project**
- **Generate the bitstream**
- **Export the .hdf**

For this session, we'll take the "Optional" path and skip the bitstream generation

Live Vivado Demo

High-level Tool Flow

Face Detection Application Example - Petalinux



XILINX.

Petalinux Project Overview

- Create a new Petalinux project with the "Template Flow" i.e. No BSP
- Add some new Yocto Recipes and recipe modifications
- Import the .hdf from Vivado
- Configure some Ultra96-specifc hardware options
- Add some necessary packages to the root filesystem
- Update the device tree to add the DPU
- Build the project
- Create a boot image

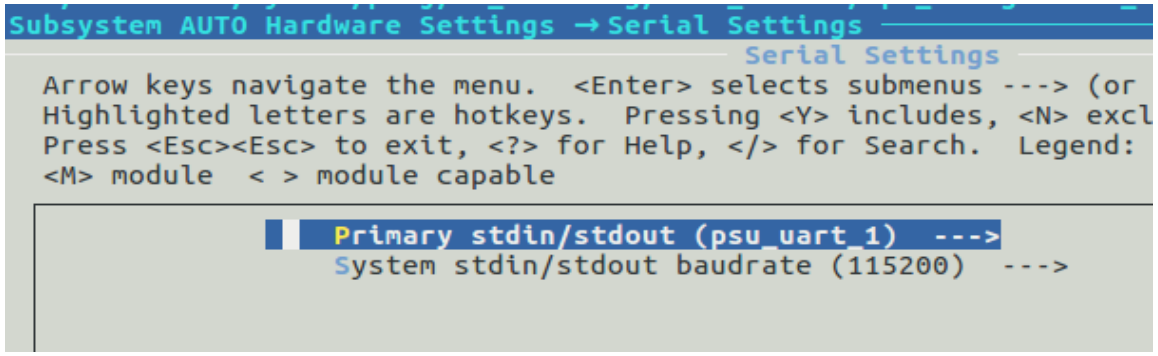
Live Petalinux Demo

Yocto Recipe Additions/Modifications

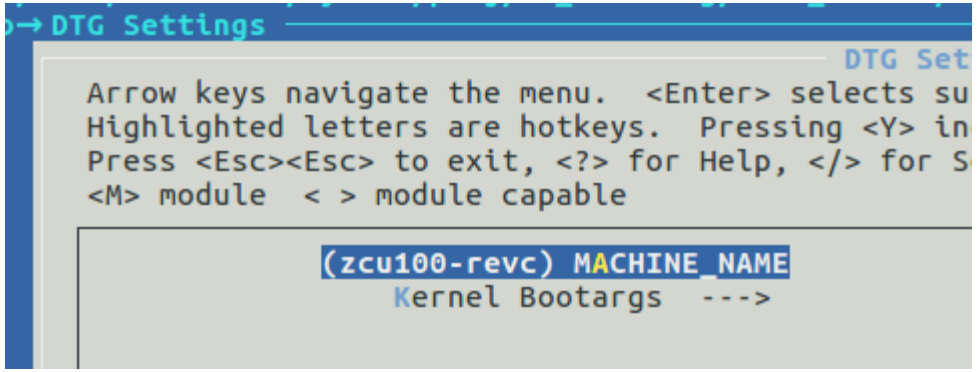
- **Add a recipe for OpenCV v3.1.**
  - This is the version needed by the DPU libraries, but Petalinux builds v3.3 by default.
- **Modify the Petalinux Yocto configuration to use OpenCV v3.1 instead of v3.3**
- **Add a bbappend for the protobuf package to change the branch that its source is pulled from.**
  - This is needed due to the OpenCV v3.1 change.
- **Add a bbappend to modify the LINUX_VERSION_EXTENSION of the kernel.**
  - This is needed to make the pre-built dpu kernel module (dpu.ko) “version magic” match the kernel that we build. Without this change, dpu.ko will fail to be inserted at boot.
- **Add a recipe to add the DPU driver, utilities, libraries, and header files into the root file system.**
- **Add a bbappend for the base-files recipe to do various things like auto insert the DPU driver, auto mount the SD card, modify the PATH, etc.**

Board Config – Ultra96

- Import the hardware description file from Vivado
  - `petalinux-config --get-hw-description=../hsi`
- Change the serial port to PSU_UART1



- Select the Ultra96 Machine (Ultra96 was originally called zcu100)



Device Tree

- The device tree generator does not yet support the DPU – must be added manually
- **reg:** Slave Register address MUST be mapped to 0x8f000000 in this release
- **interrupts:** can be connected to any PL->PS interrupt.
- **core-num**: must match number of interrupt tuples and must match the hardware

PS Interface	GIC IRQ #	Linux IRQ #
PL_PS_IRQ1[7:0]	143:136	111:104
PL_PS_IRQ0[7:0]	128:121	96:89

To get the interrupt number to put in the device tree, subtract 32 from GIC IRQ to get Linux IRQ

For example, in our Vivado project, we connected to PL_PS_IRQ0[0] which is GIC IRQ# 121 (per TRM).
121-32 = **89 (0x59)**

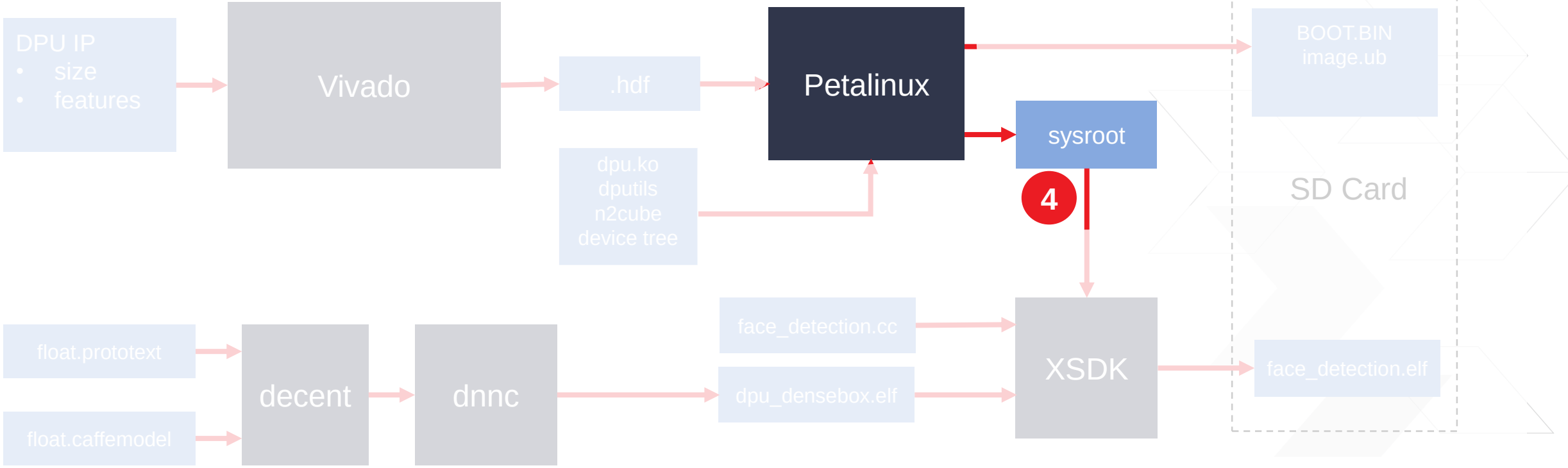
```
&amba{
        dpu@8f000000 {
                compatible = "deephi, dpu";
                interrupt-parent = <&gic>;
                interrupts = <0x0 0x59 0x1 >;
                reg = <0x0 0x8f000000 0x0 0x700>;
                memory = <0x60000000 0x80000000>;
                core-num = <0x1>;
        };
 };
```

3 DPU Core Example:

```
interrupts = <0x0 0x59 0x1 0x0 0x5a 0x1 0x0 0x5b 0x1 >;
core-num = <0x3>;
```

High-level Tool Flow

Face Detection Application Example - sysroot



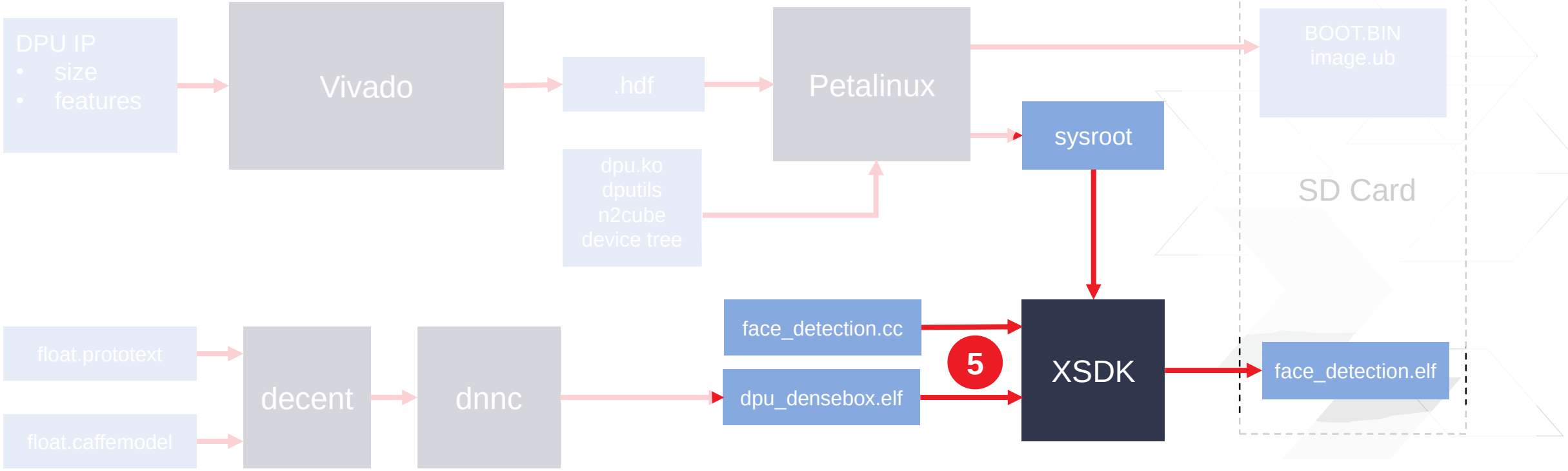
XILINX.

Sysroot generation

- **We need "sysroot" so we can build applications against the libraries/header files**
  - These files are included in the root filesystem, but we need them available at application build time
- **To build it, you use the following commands:**
  - `petalinux-build --sdk`
    - This builds a Yocto SDK and deploys it at <project dir>/images/linux/sdk.sh
  - `petalinux-package --sysroot`
    - This command installs the SDK at images/linux/sdk/sysroots/aarch64-xilinx-linux
    - sysroot can be found at : images/linux/sdk/sysroots/aarch64-xilinx-linux

High-level Tool Flow

Face Detection Application Example - XSDK

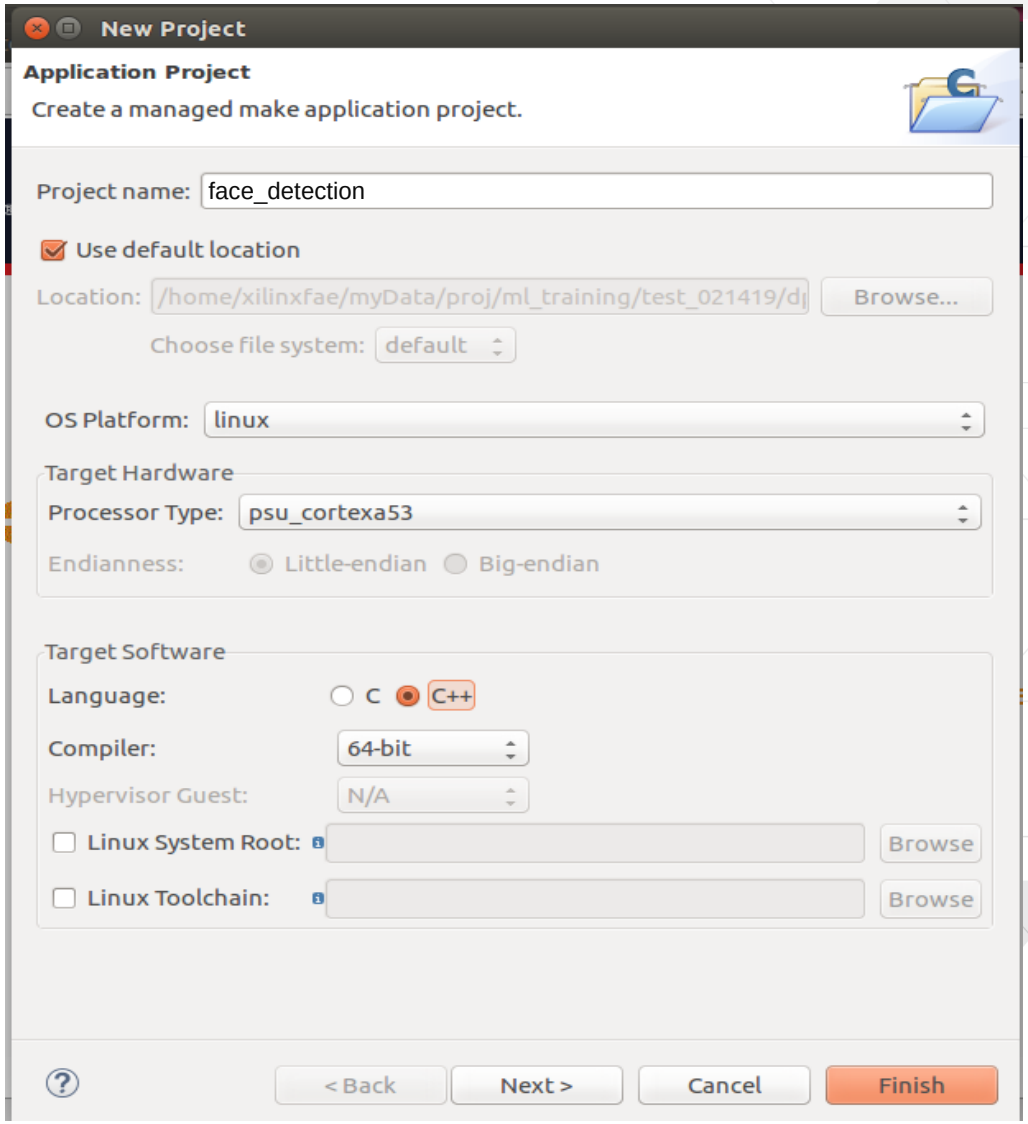


XILINX.

Live XSDK Demo

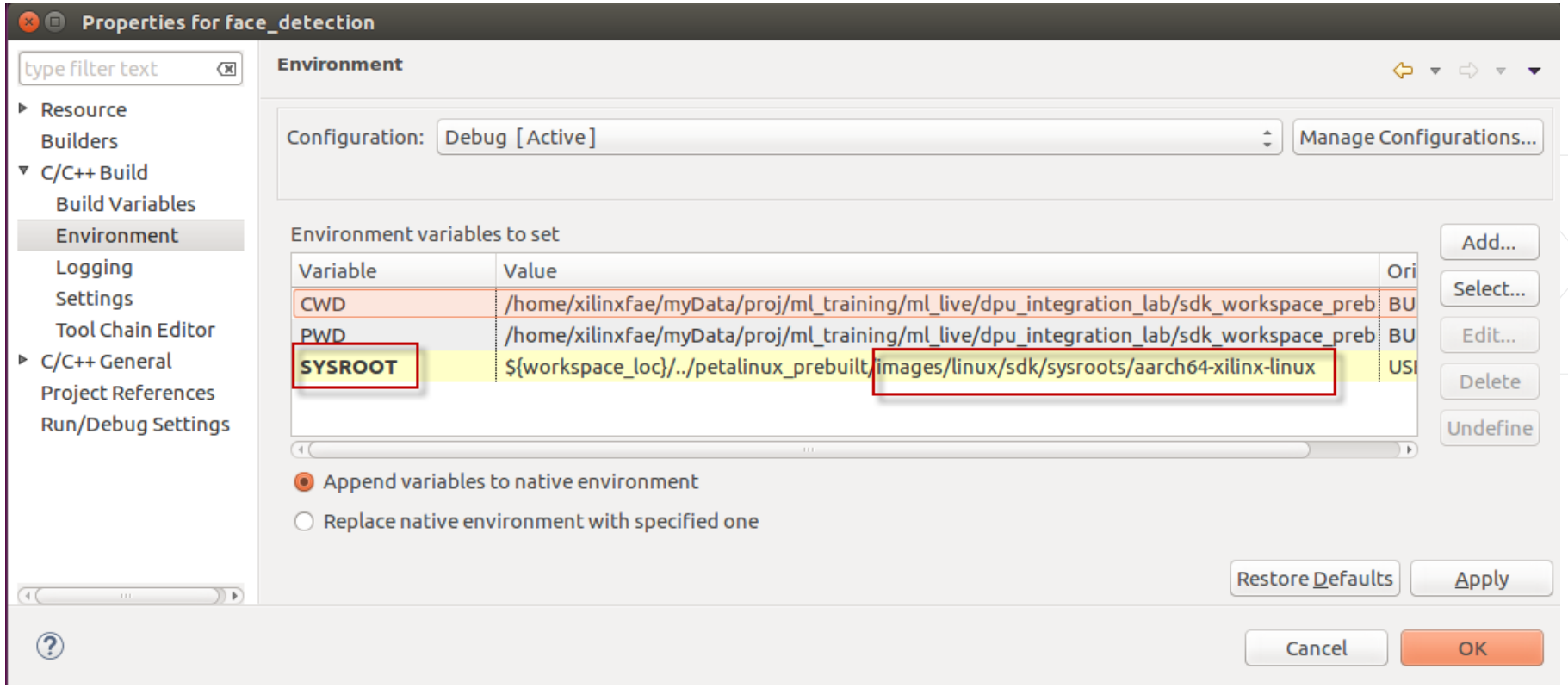
XSDK Application Creation

- File->New Application Project
- **Name**: face_detection
- **OS Platform**: Linux
- **Processor Type**: psu_cortexa53
- **Language**: C++
- Click Next
- Choose "Empty Application"
- Click Finish



XSDK Project Overview

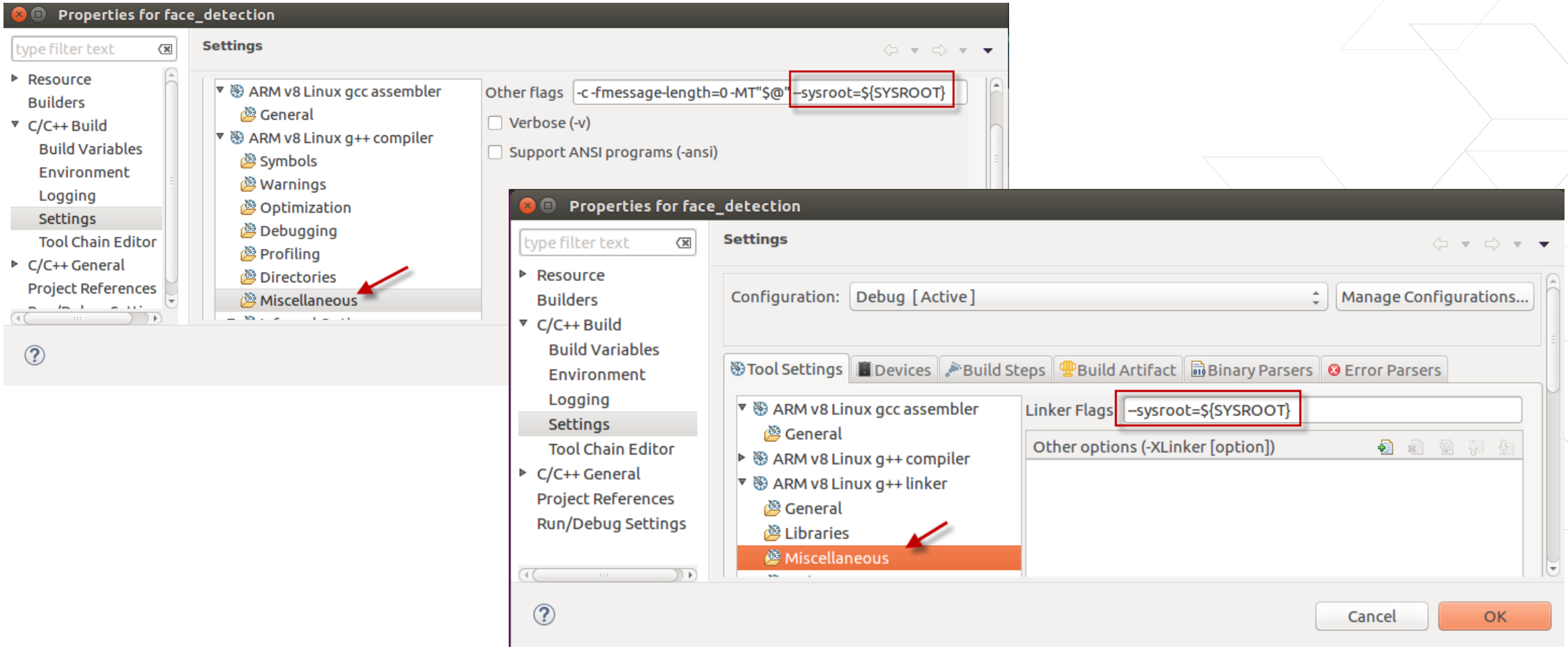
> Create a SYSROOT environment variable



XILINX.

XSDK Project Overview

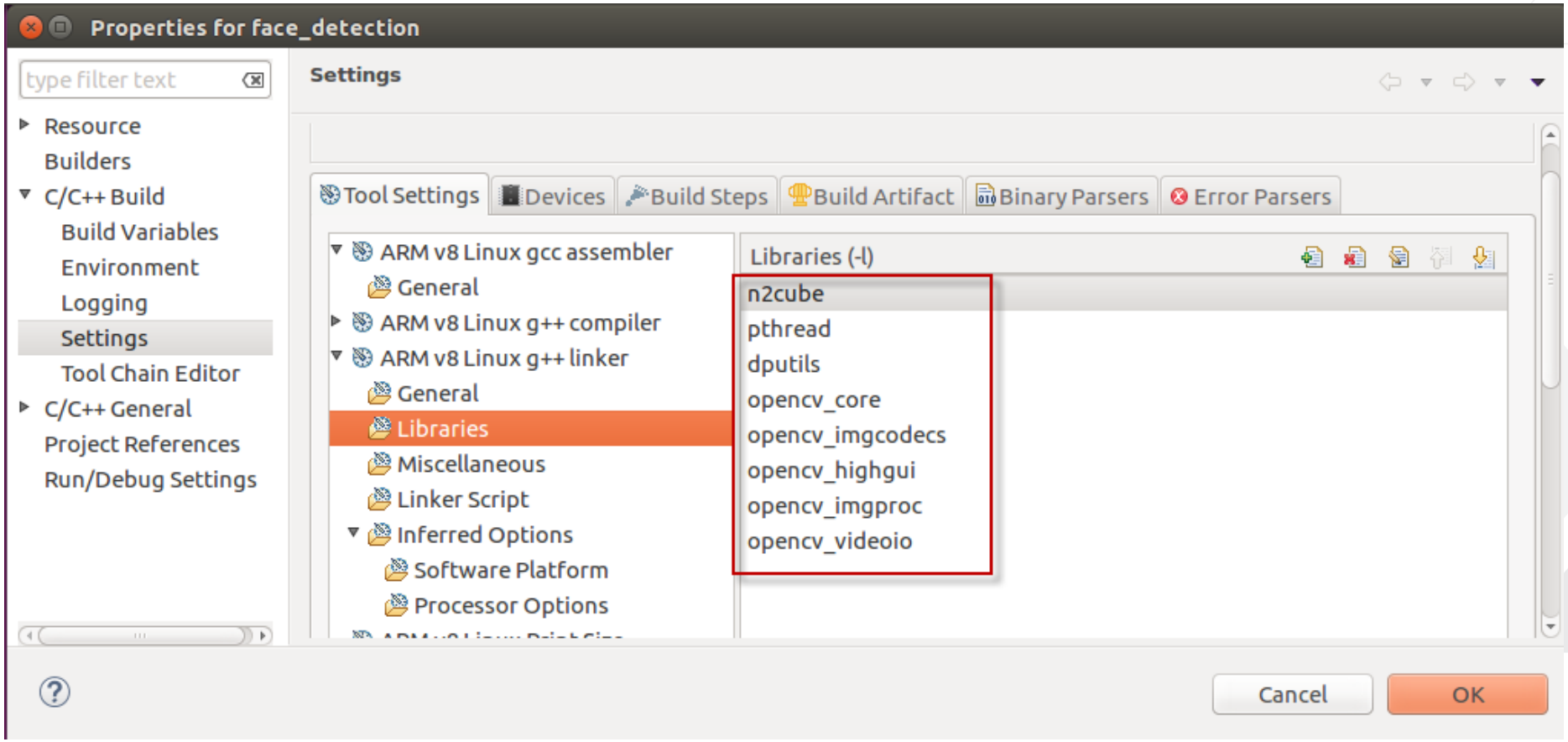
> Point the Compiler and Linker to SYSROOT



XILINX.

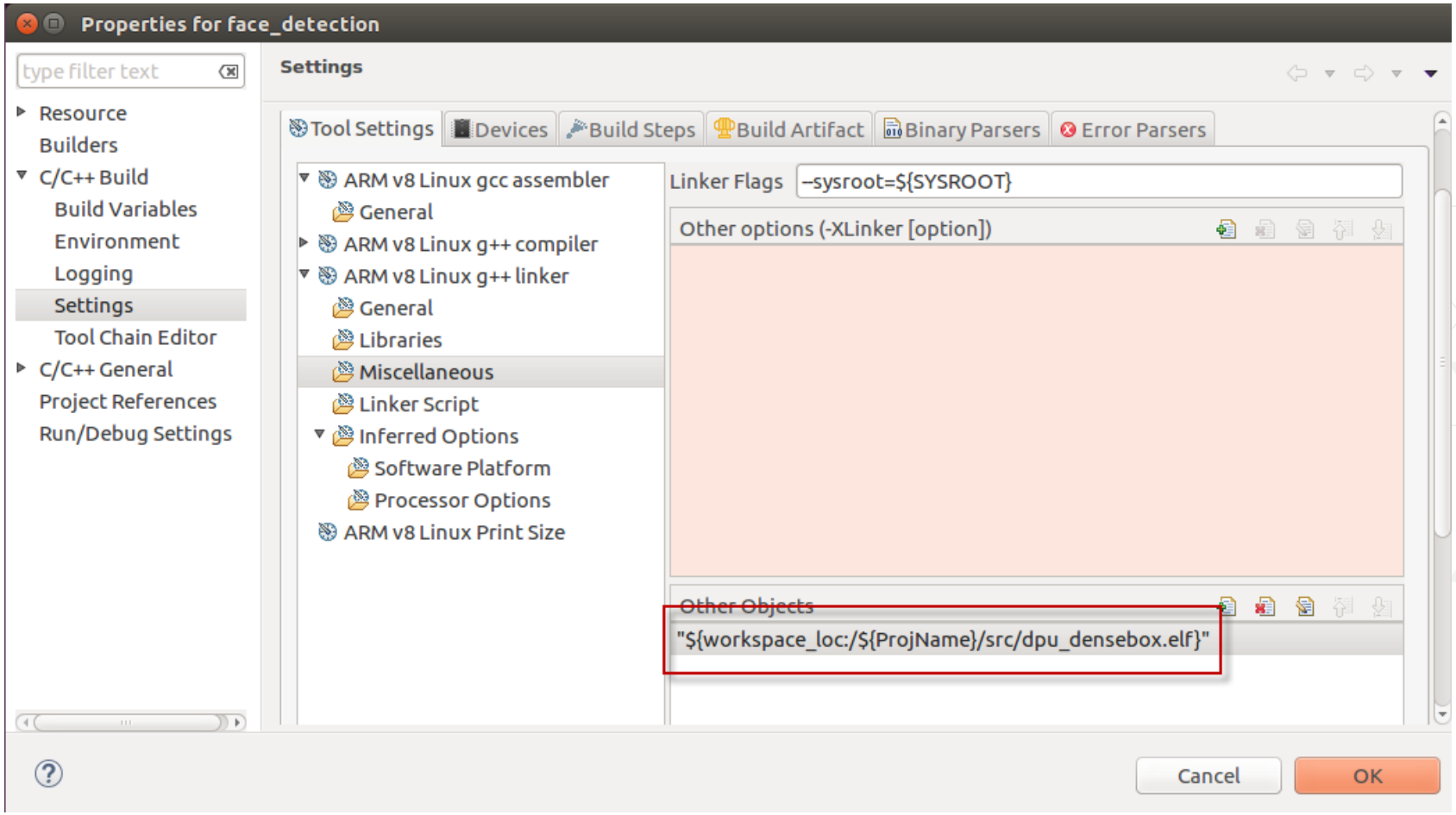
XSDK Project Overview

> Add needed libraries



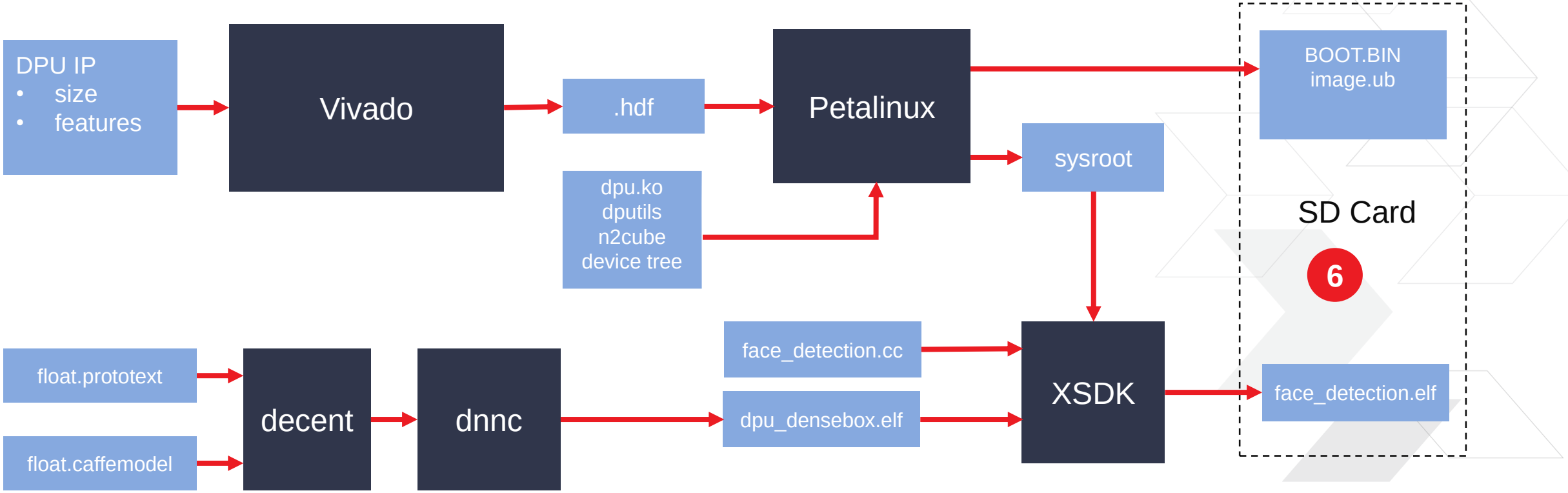
XSDK Project Overview

> Link in the Model .elf from DNNDK



High-level Tool Flow

Face Detection Application Example – Package Images



XILINX.

Package the Images

- **Package the boot images:**
  - `petalinux-package --boot --fsbl zynqmp_fsbl.elf --u-boot u-boot.elf --pmufw pmufw.elf --fpga system.bit --force`

- **Copy images to the SD card:**
  - **Kernel+rootfs+device tree:** `petalinux/images/linux/image.ub`
  - **Zynq Boot Image:** `petalinux/images/linux/BOOT.BIN`
  - **Face Detection App:** `sdk_workspace/face_detection/Debug/face_detection.elf`

Call to Action

Download the new DPU TRD from Xilinx.com

- Available today at the Xilinx Edge AI developer hub:
- https://www.xilinx.com/products/design-tools/ai-inference/ai-developer-hub.html

Integrate the DPU into your custom design

- Build your own Edge AI inference application and deploy it on custom hardware

Adaptable.
Intelligent.

XILINX