### 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





>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**



#### **Face Detection Application Example**



#### **Face Detection Application Example**



#### **Face Detection Application Example - DNNDK**



### **Face Detection Application Example - Vivado**



**EXILINX**.

### **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

**E** XILINX

### **Live Vivado Demo**





#### **Face Detection Application Example - Petalinux**



**EXILINX**.

### **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 protobult 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

Arrow keys navigate the menu. <Enter> selects submenus ---> (or Highlighted letters are hotkeys. Pressing <Y> includes, <N> excl Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: <M> module <> module capable Primary stdin/stdout (psu\_uart\_1) ---> System stdin/stdout baudrate (115200) --->

> 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 = \langle 0x0 \ 0x8f000000 \ 0x0 \ 0x700 \rangle;
       memory = <0x6000000 0x8000000>;
       core-num = \langle 0x1 \rangle;
```

```
3 DPU Core Example:
interrupts = <0x0 \ 0x59 \ 0x1 \ 0x0 \ 0x5a \ 0x1 \ 0x0 \ 0x5b \ 0x1 >;
core-num = \langle 0x3 \rangle;
```

**E** XILINX.

};

### **Face Detection Application Example - sysroot**



# **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

### **Face Detection Application Example - XSDK**



### **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

			_
🔍 🔍 New Project	:		
pplication Projec	t	- G	
Create a managed	make application project.		
Project name: fac	e_detection		
Se del autt toc			
Location: [/home/	xilinxfae/myData/proj/ml_training/test_021419/dj	Browse	
Choose	file system: default 💲		
OS Platform: lin	11X		
Transki landaran		•	
		-	
Fadiapassy		•	
Englanness.	Cittle-endian O Big-endian		
Target Software			
Language:	○ C ● C++		
Compiler			
Compiler:			
Hypervisor Guest			
Linux System	Root: 0	Browse	
🗌 Linux Toolcha	in: Ø	Browse	
?	<back next=""> Cancel</back>	Finish	

#### > Create a SYSROOT environment variable

😣 💷 Properties for fac	e_detection			
type filter text 🛛 🗷	Environment			-
<ul> <li>Resource Builders</li> <li>C/C++ Build</li> </ul>	Configuration: D	ebug [Active]	Manage Config	jurations
Environment	Environment varia	bles to set		Add
Logging	Variable	Value	Ori	Select
Sectings Tool Chain Editor	CWD	/home/xilinxfae/myData/proj/ml_training/ml_live	e/dpu_integration_lab/sdk_workspace_preb_BU	Edit
▶ C/C++ General	SYSROOT	\${workspace_loc}/petalinux_prebuilt/images/lir	nux/sdk/sysroots/aarch64-xilinx-linux	Delete
Project References				Delete
Run/Debug Settings	(4(		)))))	Undefine
	🖲 Append variab	les to native environment		
	<ul> <li>Replace native</li> </ul>	environment with specified one		
(1)))))			Restore <u>D</u> efaults	<u>A</u> pply
?			Cancel	ОК

#### > Point the Compiler and Linker to SYSROOT

😣 💿 Properties for face_detection			
type filter text 🛛 Settings		$\Leftrightarrow \bullet \bullet \bullet \bullet \bullet \bullet$	
<ul> <li>Resource         Builders         VC/C++ Build         Build Variables         Environment         Logging         Xext Action         Xext Action</li></ul>	Other flags -c -fmessage-length= Verbose (-v) Support ANSI programs (-ansi)	=0-MT"\$@"-sysroot=\${SYSROOT}	
Settings	type filter text 🗷 ► Resource	Settings	
Image: Second	Miscellaneous           Builders             Miscellaneous           Build Variables             Build Variables           Environment             Logging           Settings             Tool Chain Editor           C/C++ General             Project References         Run/Debug Settings	Configuration: Debug [Active]	Image Configurations   Binary Parsers Binary Parsers S{SYSROOT} [option])
	?		Cancel OK

#### > Add needed libraries

😣 🗉 Properties for fac				
type filter text 🛛 🗷	Settings		⟨= = =⟩ = .	-
<ul> <li>Resource Builders</li> <li>C/C++ Build Build Variables</li> </ul>	Tool Settings Devices Puild S	Error Parsers	ĥ	
Environment Logging Settings Tool Chain Editor C/C++ General Project References Run/Debug Settings	<ul> <li>ARM v8 Linux gcc assembler</li> <li>General</li> <li>ARM v8 Linux g++ compiler</li> <li>ARM v8 Linux g++ linker</li> <li>General</li> <li>Libraries</li> <li>Miscellaneous</li> <li>Linker Script</li> <li>Inferred Options</li> <li>Software Platform</li> <li>Processor Options</li> </ul>	Libraries (-l) n2cube pthread dputils opencv_core opencv_imgcodecs opencv_highgui opencv_imgproc opencv_videoio		
?			Cancel OK	



#### > Link in the Model .elf from DNNDK

8  Properties for face_detection						
type filter text 🛛 🕱	Settings		<> ▼ <> ▼ ▼			
<ul> <li>Resource</li> <li>Builders</li> </ul>	🛞 Tool Settings 📕 Devices 🎤 Build St	eps 🚇 Build Artifact 🗟 Binary Parsers 🛛 😣 Error Parsers				
▼ C/C++ Build	🔻 🛞 ARM v8 Linux gcc assembler	Linker Flags -sysroot=\${SYSROOT}		_		
Build Variables	🖉 General	Other options (-XLinker [option])				
Environment	ARM v8 Linux g++ compiler		we the Kei			
Logging	ARM v8 Linux g++ linker					
Settings	🖉 General					
Tool Chain Editor	Libraries			-		
C/C++ General	Miscellaneous					
Project References	🖉 Linker Script					
Run/Debug Sectings	V 🖉 Inferred Options					
	Software Platform					
	Processor Options					
	W ARM V8 LINUX Print Size					
		Other Objects	1 2 2 2			
		"\${workspace_loc:/\${ProjName}/src/dpu_densebox.elf}"				
(4()))))				-		
				1		
(?)		Cancel	ОК			



### **Face Detection Application Example – Package Images**



### **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/



# 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.

