DAC 2018 FPGA design contest

Naveen Purushotham, Xilinx
Jingtong Hu, University of Pittsburgh
Bei Yu, Chinese University of Hong Kong
Xinyi Zhang, University of Pittsburgh
Agenda

- Welcome
- DAC Contest Committee
  - Contest Introduction
  - Webinars
  - Piazza
- PYNQ™ & Reference design discussion
  - Things to know
  - Design rules
  - Reference design
- Questions & Answers
DAC Contest Committee
Python productivity for Zynq

An open-source framework for combining SW and HW libraries on Zynq

- Use SW libraries of Python language
- Exploit programmable logic and microprocessors using HW libraries

Out of Box

Prebuilt SD card – Python, Jupyter, Ubuntu & Bitstreams
Python & Debian & Bitstream extensible

PYNQ marries data science software and capabilities of zynq and programmable hardware

PYNQ-Z1 – First PYNQ supported board
Jupyter Notebooks: browser-based development ... with rich, multi-media support

- Designed for
  - Interactive, exploratory computing
  - Reproducible results

- Ideal for
  - Teaching and learning
  - Projects and research

- Provides
  - Interactive design with Zynq
  - application-oriented perspective

Where to find more notebooks

github.com/ipython/ipython/wiki/A-gallery-of-interesting-IPython-Notebooks
Overlays aka hardware libraries – special bitstreams

Step 1:
Create an FPGA design for a class of related applications

Step 2:
Export the bitstream and a C API for programming the design

Step 3:
Wrap the C API to create a Python library

Step 4:
Import the bitstream and the library in your Python scripts and program
Neural Network Application example

Binary Neural Network on Pynq

Note: All code is entered in browser window and run

**Import the HW libraries for PL and Overlay**

```
In [ ]: from pynq import PL, Overlay
```

**Instantiate and download the classifier overlay**

```
In [ ]: classifier = Overlay("bnn.bit")
classifier.download() #programs the Zynq FPGA
```
Neural Network Application example

**Import SW python libraries and open image to be classified**

```python
In [7]: from PIL import Image
    import numpy as np

im = Image.open('deer.png')
im
```

**Launch BNN in hardware**

```python
In [5]: class_out = classifier.classify_image(im)
print("Class number: {}").format(class_out))
print("Class name: {}").format(classifier.class_name(class_out)))

Inference took 2240.00 microseconds
Classification rate: 446.43 images per second
Class number: 4
Class name: Deer
```

**Show results and analysis**
Productivity level tools for Zynq

- Python scripts & programs
- Interactive console
- Python Libraries
  - CPython Virtual Machine
  - Ubuntu core
  - IP drivers in C
- Post-bitstream-programmable overlay

Languages and Tools:
- C, C++ or SystemC
- Vivado™ HLS
- VHDL or Verilog
- System IP Integration
FINN: A Framework for Fast, Scalable Binarized Neural Network Inference

Yaman Umuroglu*, Nicholas J. Fraser*, Giulio Gambardella†, Michaela Blott‡, Philip Leong*, Magnus Jähre*, Kees Vissers§

* Xilinx Research Labs; † Norwegian University of Science and Technology; ‡ University of Sydney

ABSTRACT

Research has shown that convolutional neural networks contain significant redundancy, and high classification accuracy can be obtained even when weights and activations are reduced from floating point to binary values. In this paper, we present FINN, a framework for building fast and flexible FPGA accelerators using a flexible heterogeneous streaming architecture. By utilizing a novel set of optimizations that enable efficient mapping of binarized neural networks to hardware, we implement fully connected, convolutional and pooling layers, with per-layer compute resources being tailored to meet project-specific requirements. On a

Int. Symposium on FPGAs, Feb. 2017

- Unprecedented image classification rates
- 1,000x speed-up over Raspberry Pi3

https://github.com/Xilinx/BNN-PYNQ

SVHN, Road signs, CIFAR-10

Image pre-processing in Python

Binary Neural Network in FPGA & ARM CPU

“cat”
“A high productivity tool for experienced FPGA designers”

http://fpga.org/2017/09/05/pynq-as-a-high-productivity-platform-for-fpga-design-and-exploration

“Fellow FPGA designers, try Pynq. You’ll like it. Pynq makes exploring new FPGA ideas lightweight, fresh, fast, easy, fun again..”

Jan Gray … feedback on implementing 80 x 32-bit RISC cores on PYNQ-Z1
PYNQ is completely open-source

Where to find more information
http://www.pynq.io
PYNQ Team standing by...

Build something cool

Contribute to Open Source

Contribute Reproducible Results

If not already a member, join the PYNQ support forum
http://www.pynq.io/support.html/
DAC Contest reference design

Batch size = 500

open and resize images

inference

- Timer should start before opening the images and end after PS side receives all detected coordinates.
- Write to XML can be excluded from timer.