Xilinx Technology Showcase & Hackathon 2017

Connect • Learn • Share

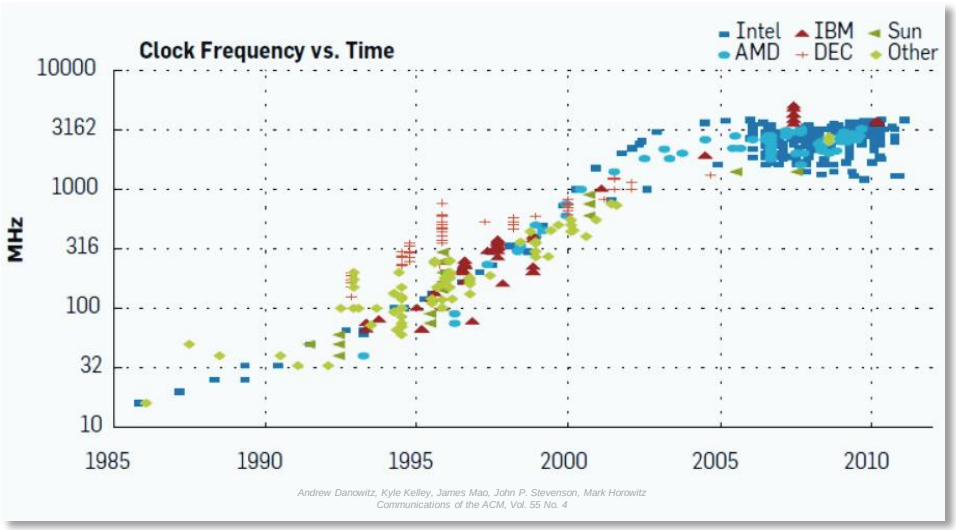
XILINX
ALL PROGRAMMABLE™

Welcome to the Xilinx Hackathon 2017

CPU Architectures not Scaling with Workloads

Continued scaling requires accelerated computing, from Edge to Cloud

- Processor frequency scaling ended in 2007
- Multicore architecture scaling has flattened

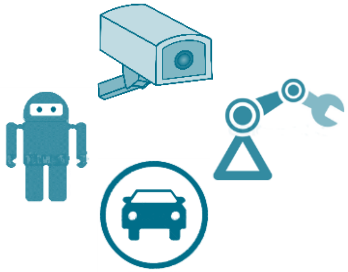


- Workloads require higher performance, lower latency
  - Cloud: video, big data, AI…
  - Edge: auto, surveillance, AI…

Workloads Move from Edge to Cloud… and Back

Reconfigurable accelerators also needed at the edge

Cloud Computing

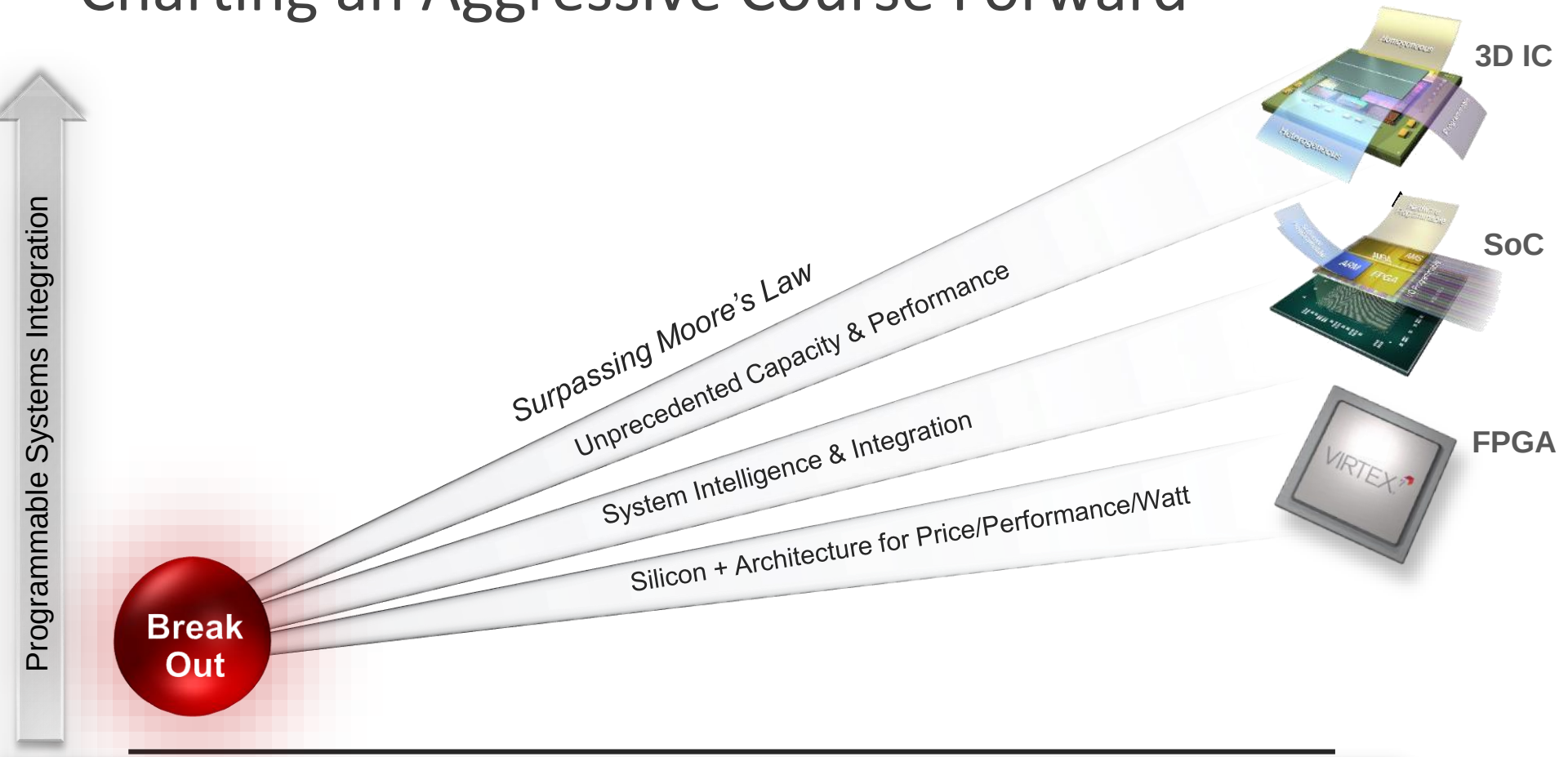


Edge Computing

Applications are moving to the Cloud:
centralized, elastic resources…

Applications are moving to the Edge:
preprocessing, lower network transfer

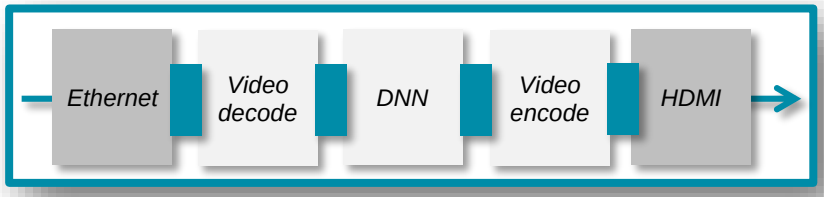
Charting an Aggressive Course Forward



Tools	VIVADO	UltraFAST Design Methodology	SDx Environments	
Architecture	7 Series	UltraScale™ Architecture		
Process	28HPL	20SoC	16FinFET+	7nm

Emerging Solution: All Programmable Accelerators

- Custom, accelerated processing
  - Reconfigurable to handle diverse workloads
- Custom memory hierarchy
  - Avoid excessive DDR transfers
- Custom IOs
  - Sensor Fusion at the edge



Custom dataflow, memory, IO

Virtex® FPGA

Zynq® MPSoC

Harness the Flexibility of Programmable HW

- HW engineers

VIVADO

- Embedded SW developers

Linux

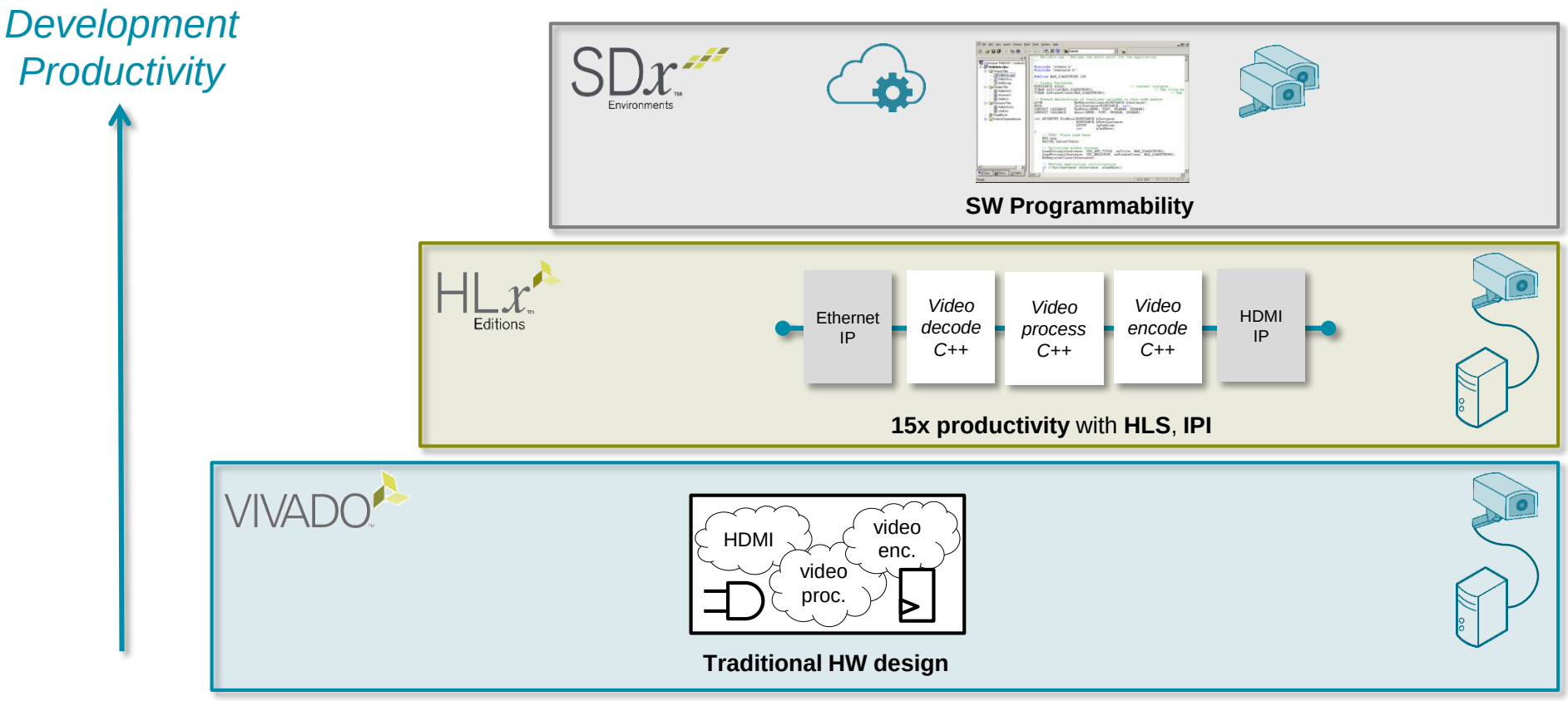
- Application SW developers

OpenCL

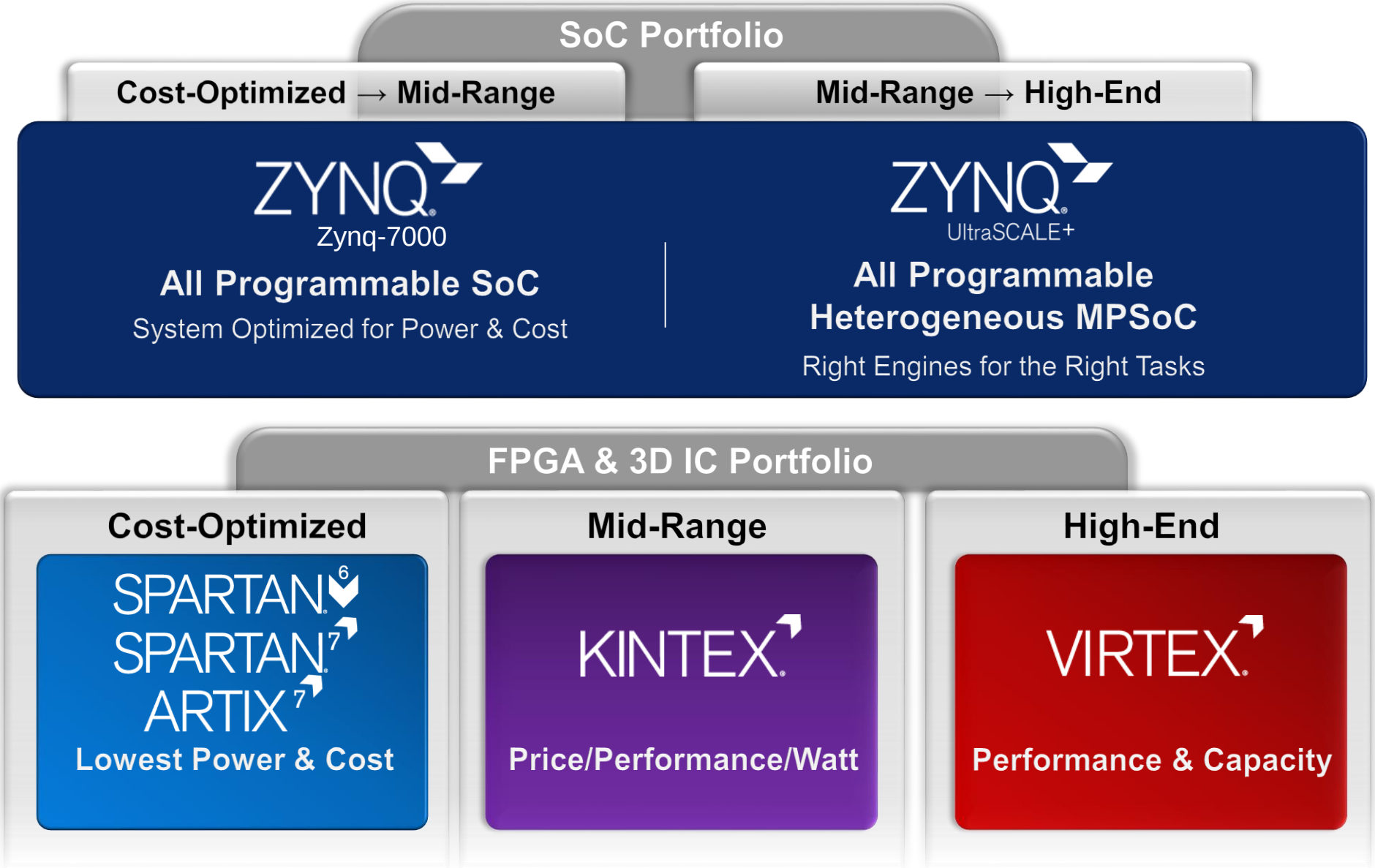
Virtex® FPGA

Zynq® MPSoC

Breakout in Programming Model



Xilinx Product Families: A Broad Portfolio

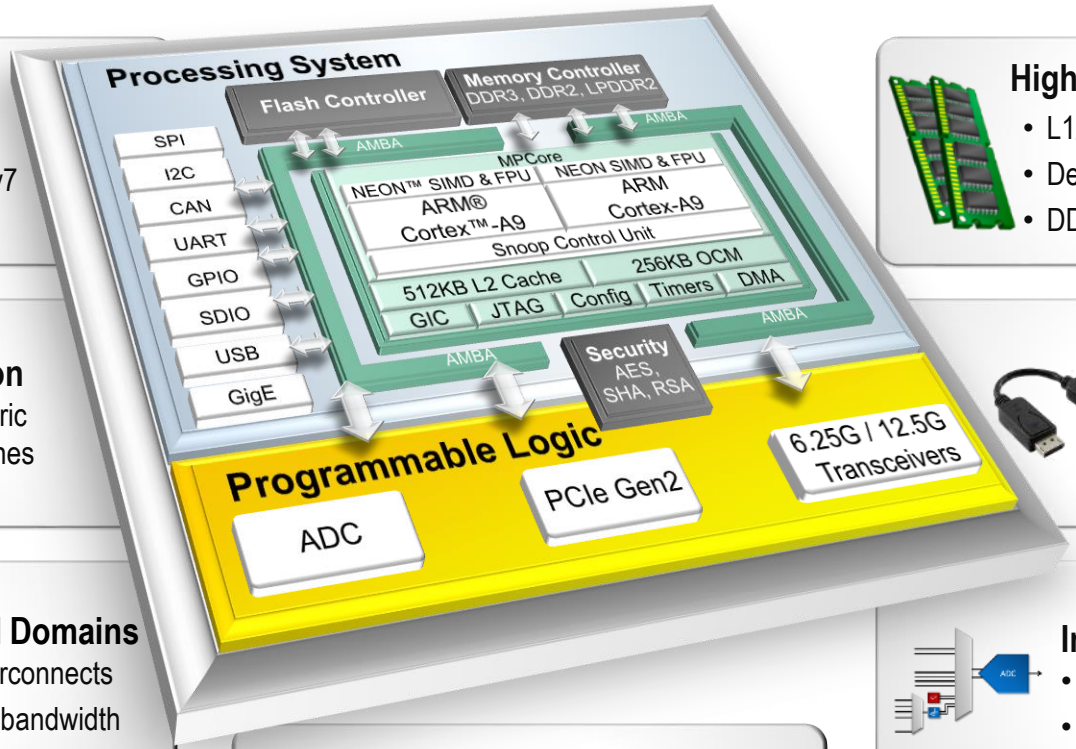


All Programmable SoC - Zynq®-7000

Zynq-7000S
Single-Core @ 766MHz
Artix-7 FPGA Fabric

Zynq-7000
Dual-Core @ 800MHz
Artix-7 FPGA Fabric

Zynq-7000
Dual-Core @ 1GHz
Kintex-7 FPGA Fabric



Application Processor
- 32-bit ARMv7
- Up to 1GHz

FPGA Acceleration
- 7 Series FPGA Fabric
- Customizable Engines

Tightly Coupled Domains
- 3000+ PS/PL interconnects
- Up to 100Gb/s of bandwidth

High Bandwidth Memory
- L1/L2 CPU Caches
- Dedicated On-Chip Memory (OCM)
- DDR3, DDR2, LPDDR2 w/ ECC

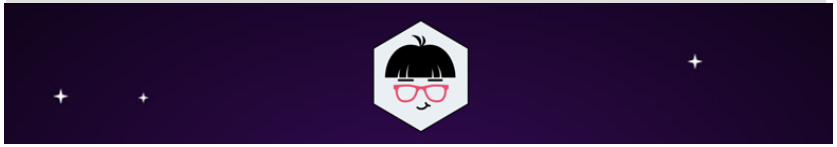
Integrated Memory Mapped Peripherals
- USB2.0, GigE, CAN
- UART, SDIO, I2C, SPI

Integrated Analog
- Dual multi-channel 12-bit ADC
- Temp & Voltage sensors

Extensive IP Portfolio
- Standardized AXI4 interfaces
- Enables peripheral expansion

XILINX
ALL PROGRAMMABLE™

Welcome and Orientation



- Logistics
- Hackathon Theme & Goals
- Community Projects
- Head back up to Retreat!

Organizers

Craig Abramson

Mindy Brooks

Derek Curd

Greg Daughtry

Brad Fross

Adrian Hernandez

Graham Schelle

Helpers

- We will be with you the full 30 hours (in shifts)!

Kv Thanjavur Bhaaskar

Anurag Dubey

Vikhyat Goyal

David Gronlund

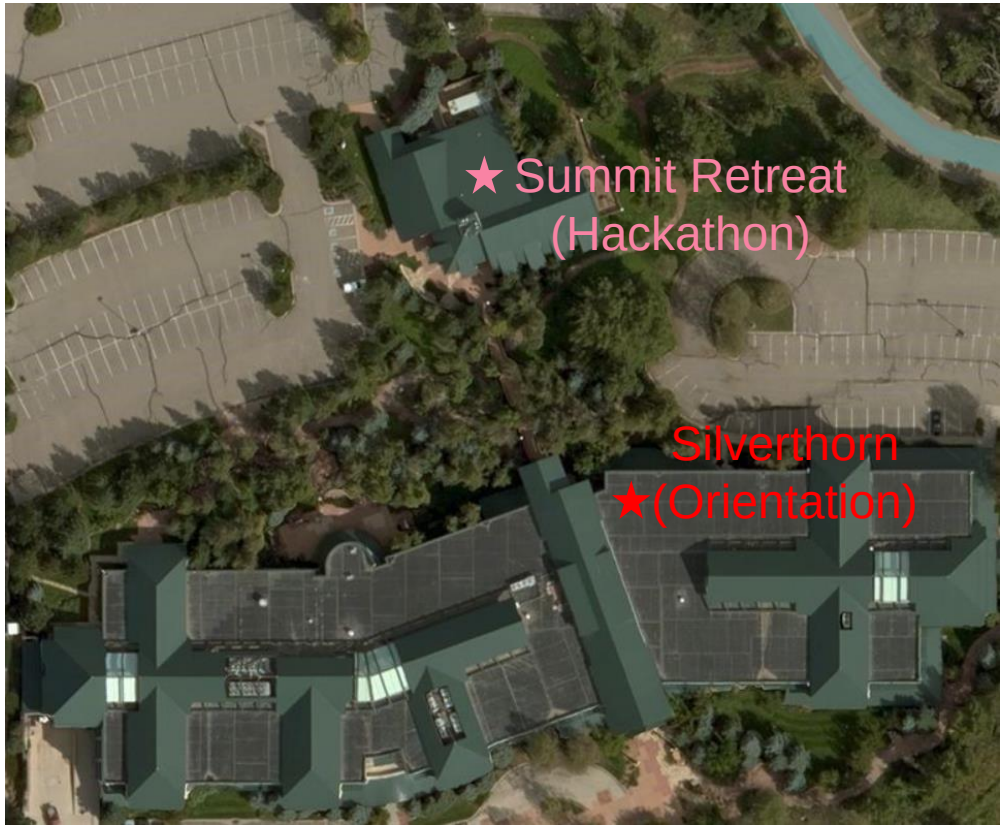
Dustin Richmond

Naveen Purushotham

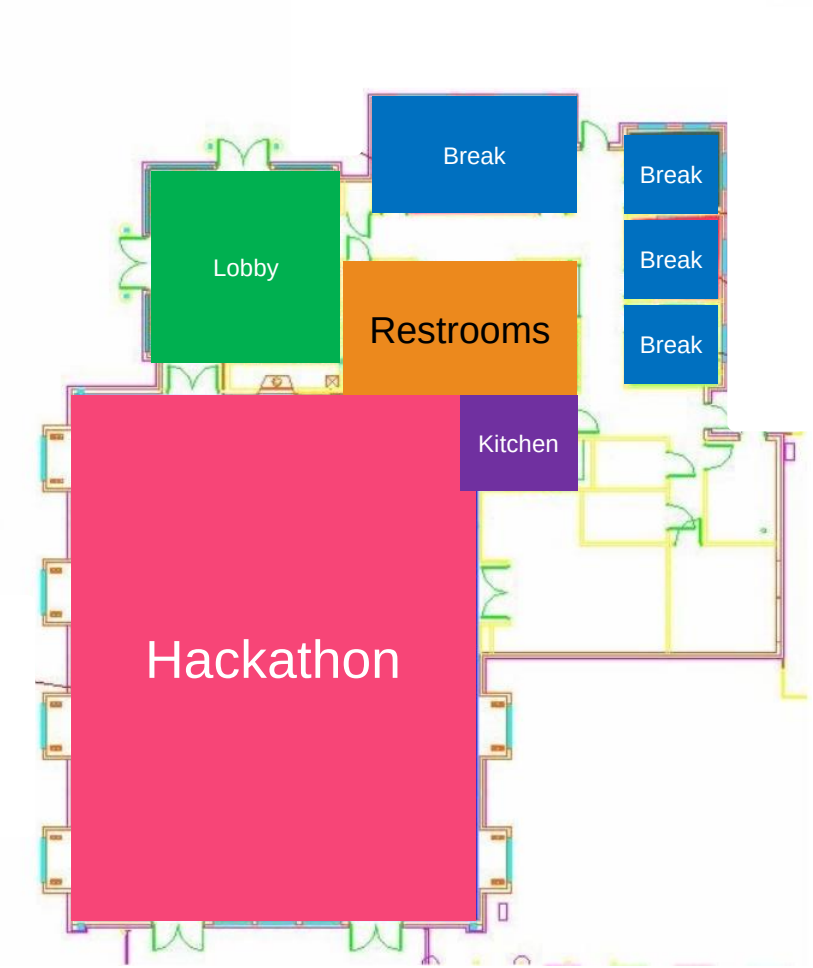
Rock Qu

Graham Schelle

Logistics: Site Map



Logistics: Summit Retreat



Logistics: Schedule

- Friday:

5 PM – 7 PM	**Dinner & Orientation** (Silverthorne)
7 PM – 8:30 PM	**PYNQ-Z1 Hand Out and Tutorials** (Move to Summit Retreat)
11 PM - 12 AM	**(Recommended) Project Pitch and Feedback**

- Saturday:

12 AM – 4 AM	**Mini Challenge 1**
8 AM – 9 AM	**Breakfast**
9 AM – 1 PM	**Mini Challenge 2**
12 PM – 1 PM	**Lunch**
8 PM – 10 PM	**Dinner**
9:30 PM	**Judging Starts**
11:00 PM	**Awards**

Drinks and Snacks will be available throughout the event

Hackathon Prizes

Category	Value
Best Design	$800
Runner Up	$400
Surprise Category	$100
Best Programmable Logic	$100
Mini Challenge 1	Surprise Team Prize ($150 value)
Mini Challenge 2	Surprise Team Prize ($150 value)
Saturday Night Non-Tech Challenge	$100

Everyone can keep their PYNQ-Z1 board!

Hackathon Judging Criteria

	Score (0-10)	Criteria	Description
Idea	10	Creativity	How creative or innovative is the idea behind the design?
	10	Viability	How relevant is the design's use case to a real problem?
Open Source and Reproducible	10	open-source	Are build steps provided, python packaged, and available on github as notebook and source code.
	10	reproducible	Could someone reproduce this work starting from a pynq v2.0 image and going through their notebook
Implementation	10	Feasibility	Does it actually work?
	10	Uniqueness	How well does the app leverage a variety of different open-source APIs and programmable logic

Code of Conduct

- Be Respectful
- Harassment and abuse are never tolerated
- Participants asked to stop any harassing behavior are expected to comply immediately.
- If you are being harassed, notice that someone else is being harassed, or have any other concerns, please contact Xilinx Hackathon organizers

- Full Code of Conduct in Hackathon packets

WiFi & Internet Access

- WiFi
  - Network securebrd
  - Password hackathon2017
- Ethernet
  - No wall-plate connections
  - Bridge network from laptop to PYNQ-Z1

Collaboration

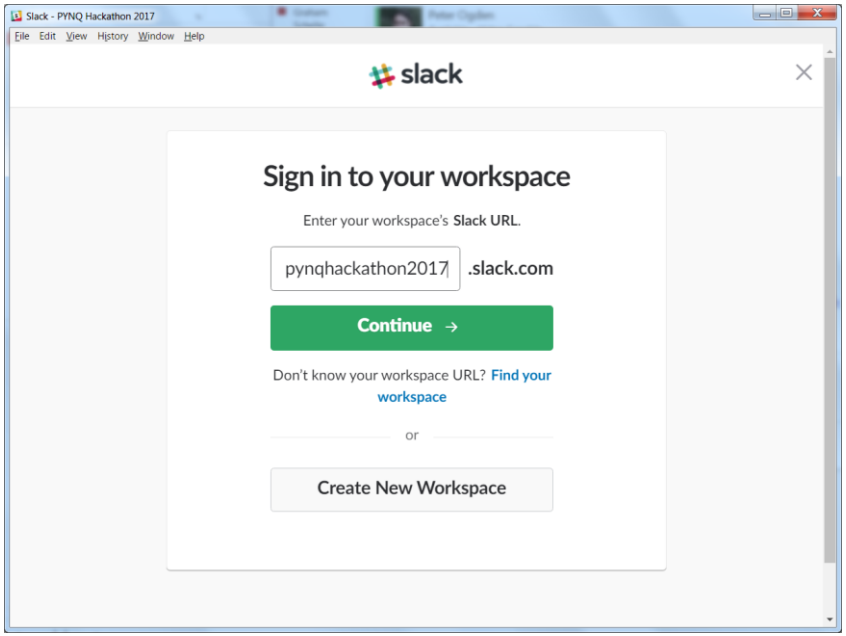
- GitHub

  PYNQ Repositories

  – http://github.com/xilinx/PYNQ

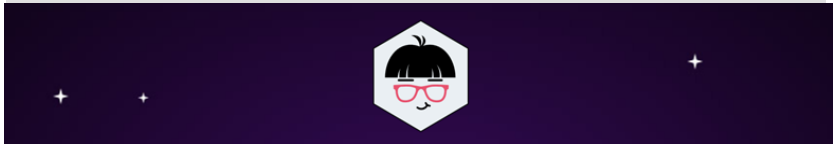
  – https://github.com/drichmond/PYNQ-Hackathon-2017

- Slack



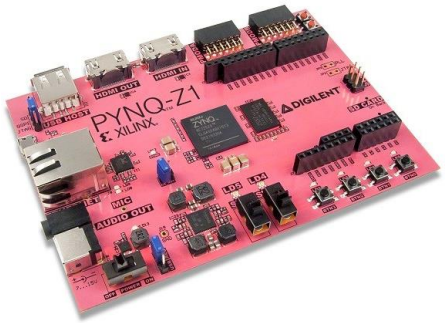
XILINX
ALL PROGRAMMABLE™

Welcome and Orientation



- Logistics
- Hackathon Theme & Goals
- Community Projects
- Head back up to Retreat!

Hackathon Themes



Build something cool

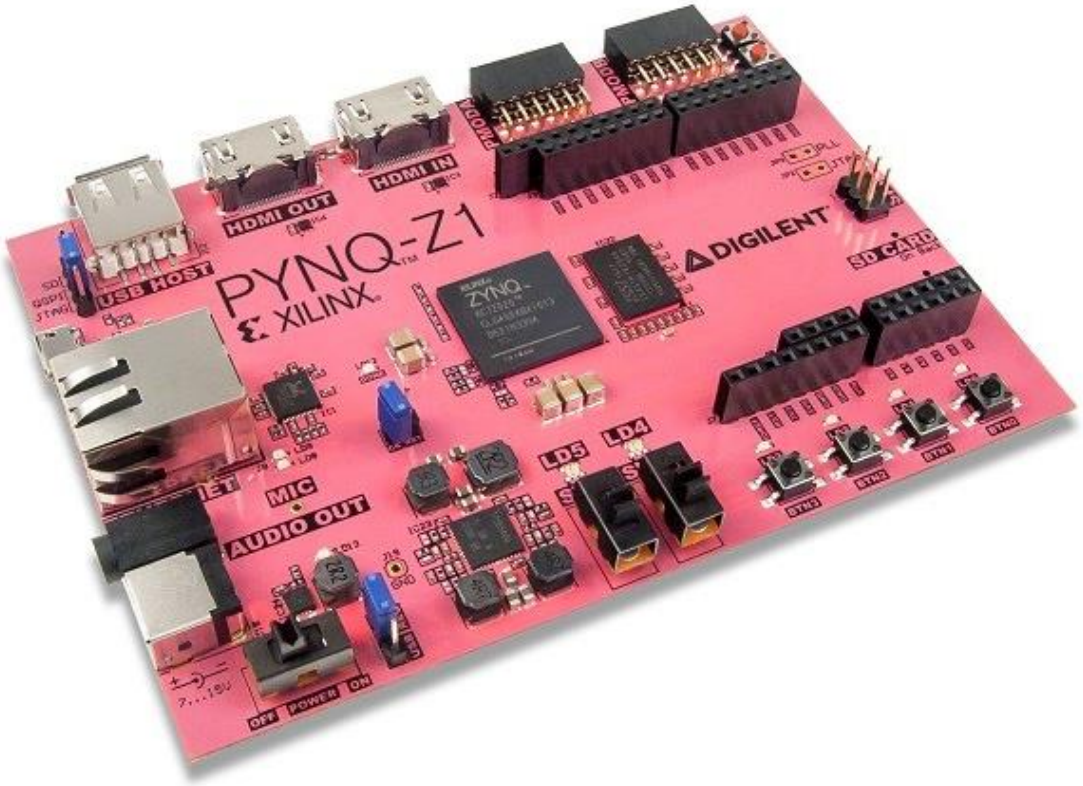
GitHub

Contribute to Open Source

jupyter

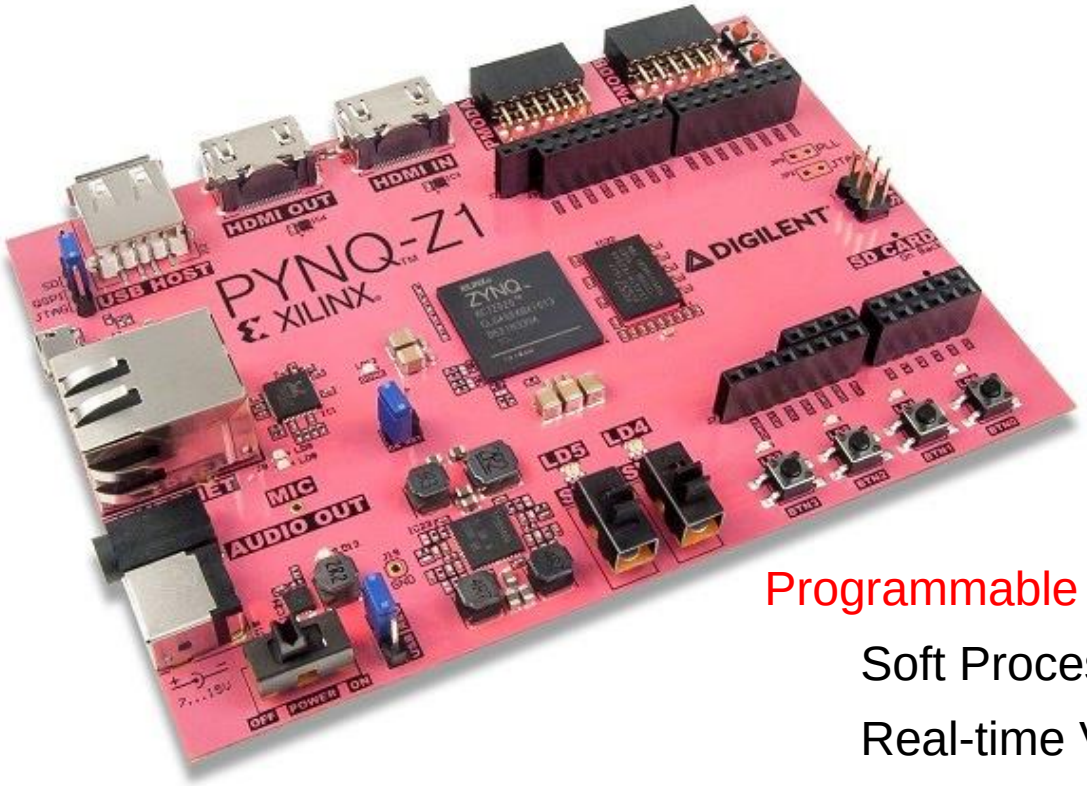
Contribute Reproducible Results

PYNQ-Z1 Board



- Zynq XC7Z020 CLG400
- 512 MB DRAM
- HDMI input, HDMI out
- Microphone input
- Audio output
- 10/100/1G Ethernet
- MicroSD slot
- USB 2.0
- 4 LEDs, 4 buttons, 2 switches
- Arduino connector
  - With additional IO
- 2 Pmod 8-pin GPIOs

PYNQ-Z1 Board



Software Preloaded

Ubuntu

Python 3.6

WiFi Connectivity

Extend it

- Apt-get, pip install, git clone

Crash it

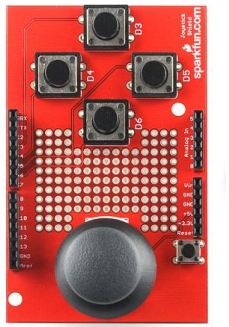
- We'll burn a new SDCard

Programmable Logic Designs Preloaded

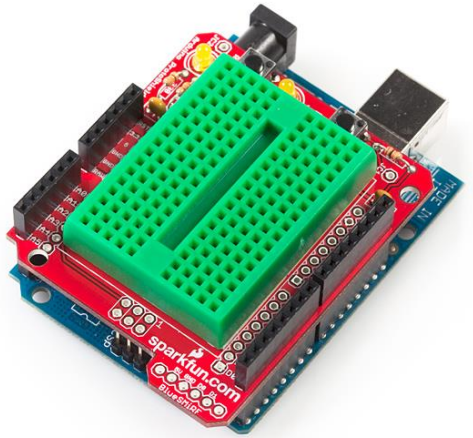
- Soft Processors on Arduino & PMOD
- Real-time Video Pipeline, Audio
- Neural Networks
- LogicTools – programmable waveforms
- 25+ example notebooks

PYNQ Peripherals (1 of 4)

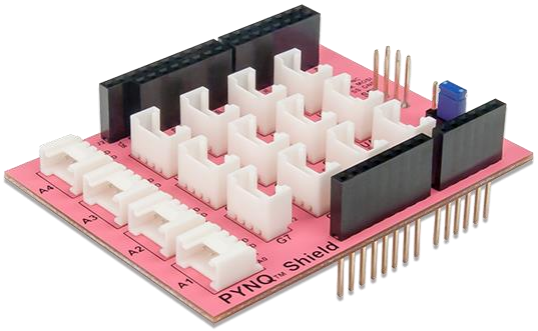
Ardumoto Shield Kit



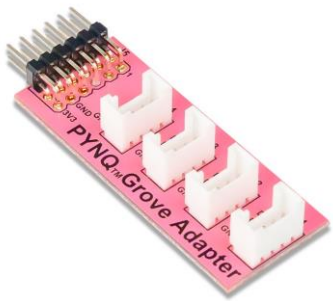
Joystick Shield Kit



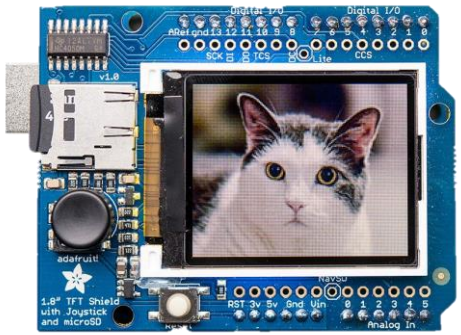
ProtoShield Kit



Arduino Grove shield

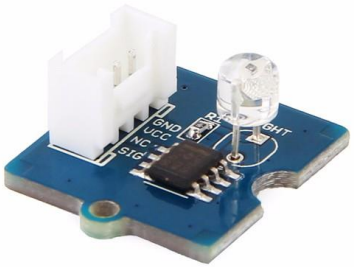


PMOD Grove Adapter

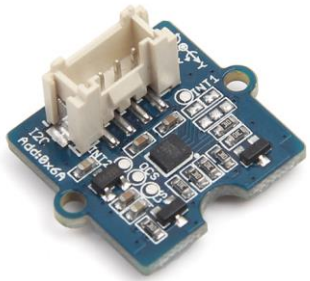


LCD18 shield

PYNQ Peripherals (2 of 4)



Light Sensor



6-Axis Accelerometer & Gyroscope

Chainable RGB LED

Ultrasonic Ranger

Temperature & Humidity Sensor

PIR Motion Sensor

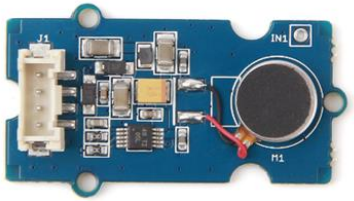
PYNQ Peripherals (3 of 4)

Buzzer

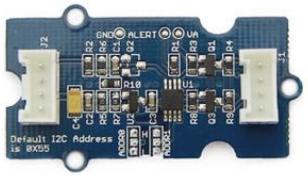
Heart Rate Monitor

OLED Display

Color Sensor



Haptic Motor



I2C Analog-Digital Converter

LED Bar

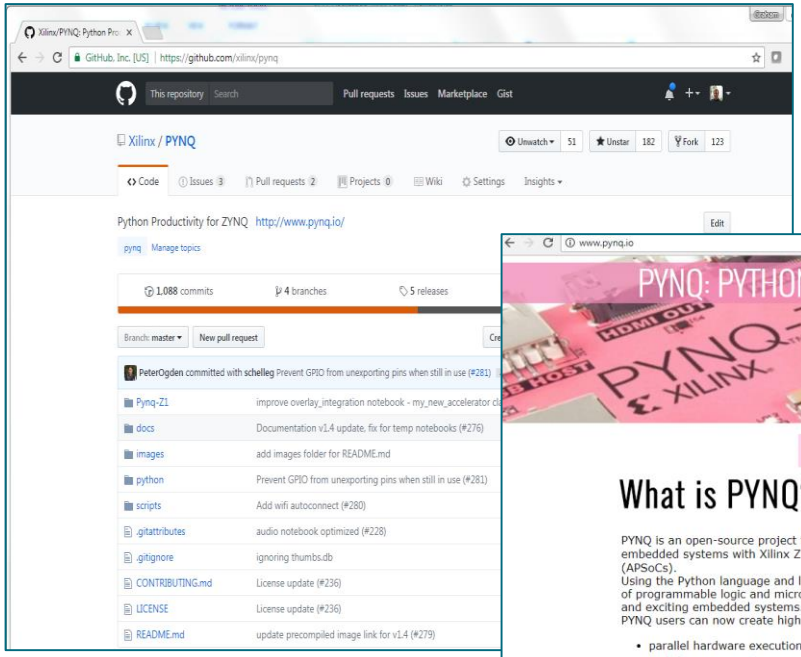
PYNQ Peripherals (4 of 4)

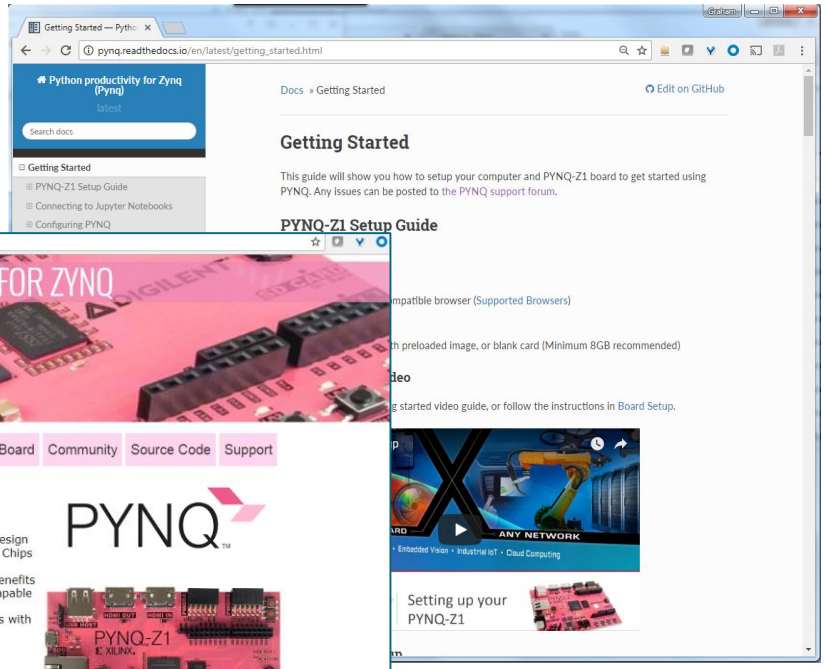
WiFi Dongle

720p USB Webcam

We'll give some project ideas when we hand out boards

PYNQ is completely open-source



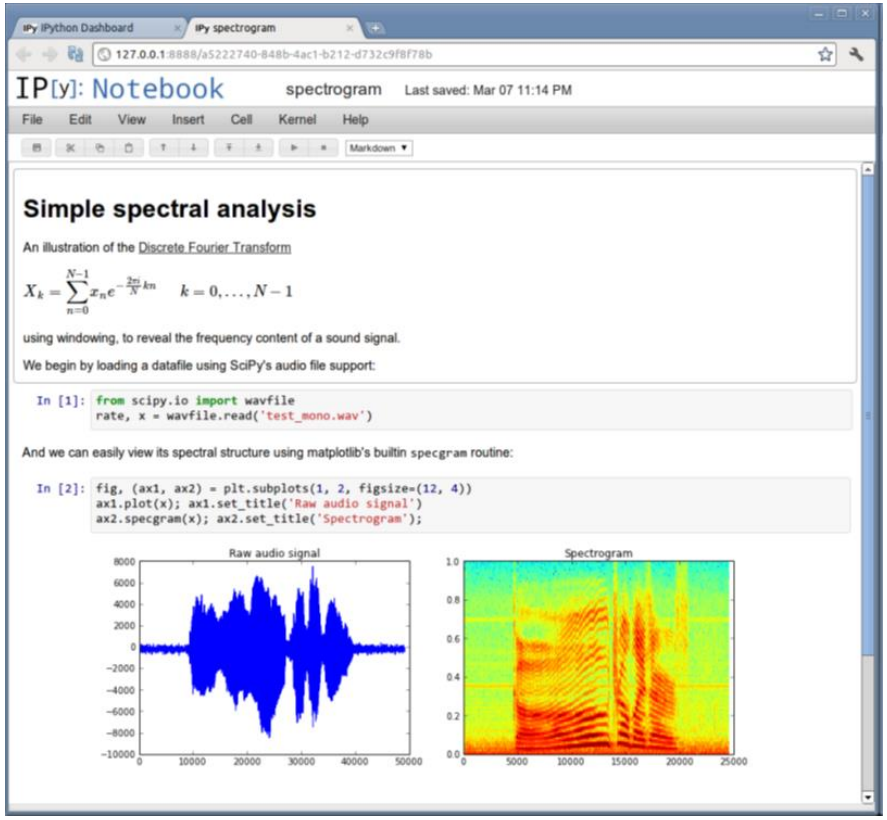


GitHub

1. No Xilinx tools required to <u>use</u> PYNQ

2. At this hackathon, we'll strive for open-*
   -source, -projects, -solutions, -collaboration

Notebooks: browser-based development … with rich, multi-media support



- Designed for
  - Interactive, exploratory computing
  - Reproducible results
- Ideal for
  - Teaching and learning
  - Projects and research
- Rapid adoption
  - Across multiple disciplines

github.com/ipython/ipython/wiki/A-gallery-of-interesting-IPython-Notebooks

Part of a broader trend towards new, web-based IDEs

2 Caltech scientists share Nobel Prize in physics for gravitational wave discoveries

In the above spectrograms, you can see lots of excess power below ~20 Hz, as well as strong spectral lines at 500, 1000, 1500 Hz (also evident in the ASDs above). The lines at multiples of 500 Hz are the harmonics of the "violin modes" of the fibers holding up the mirrors of the LIGO interferometers.

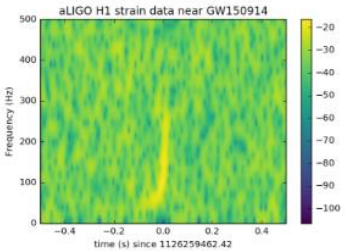
The signal is just barely visible here, at time=0 and below 500 Hz. We need to zoom in around the event time, and to the frequency range from [20, 400] Hz, and use the whitened data generated above.

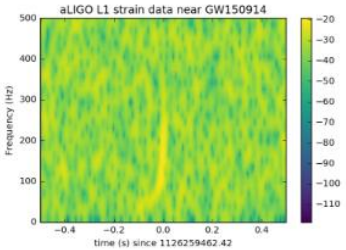
11]:
```
#  plot the whitened data, zooming in on the signal region:
tevent = 1126259462.422           # Mon Sep 14 09:50:45 GMT 2015
deltat = 10.                      # seconds around the event
# index into the strain time series for this time interval:
indxt = np.where((time_H1 >= tevent-deltat) & (time_H1 < tevent+deltat))

# pick a shorter FFT time interval, like 1/16 of a second:
NFFT = fs/16
# and with a lot of overlap, to resolve short-time features:
NOVL = NFFT*15/16
# and choose a window that minimizes "spectral leakage"
# (https://en.wikipedia.org/wiki/Spectral_leakage)
window = np.blackman(NFFT)

# Plot the H1 whitened spectrogram around the signal
plt.figure()
spec_H1, freqs, bins, im = plt.specgram(strain_H1_whiten[indxt], NFFT=NFFT, Fs=fs, window=window,
                                        noverlap=NOVL, cmap=spec_cmap, xextent=[-deltat,deltat])
plt.xlabel('time (s) since '+str(tevent))
plt.ylabel('Frequency (Hz)')
plt.colorbar()
plt.axis([-0.5, 0.5, 0, 500])
plt.title('aLIGO H1 strain data near GW150914')
plt.savefig('GW150914_H1_spectrogram_whitened.png')

# Plot the L1 whitened spectrogram around the signal
plt.figure()
spec_H1, freqs, bins, im = plt.specgram(strain_L1_whiten[indxt], NFFT=NFFT, Fs=fs, window=window,
                                        noverlap=NOVL, cmap=spec_cmap, xextent=[-deltat,deltat])
plt.xlabel('time (s) since '+str(tevent))
plt.ylabel('Frequency (Hz)')
plt.colorbar()
plt.axis([-0.5, 0.5, 0, 500])
plt.title('aLIGO L1 strain data near GW150914')
plt.savefig('GW150914_L1_spectrogram_whitened.png')
```



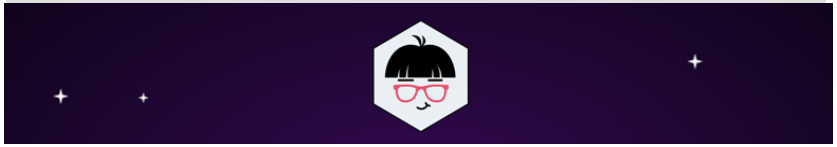


LIGO results shared as Jupyter notebooks

https://losc.ligo.org/s/events/GW150914/GW150914_tutorial.html

XILINX
ALL PROGRAMMABLE™

Welcome and Orientation



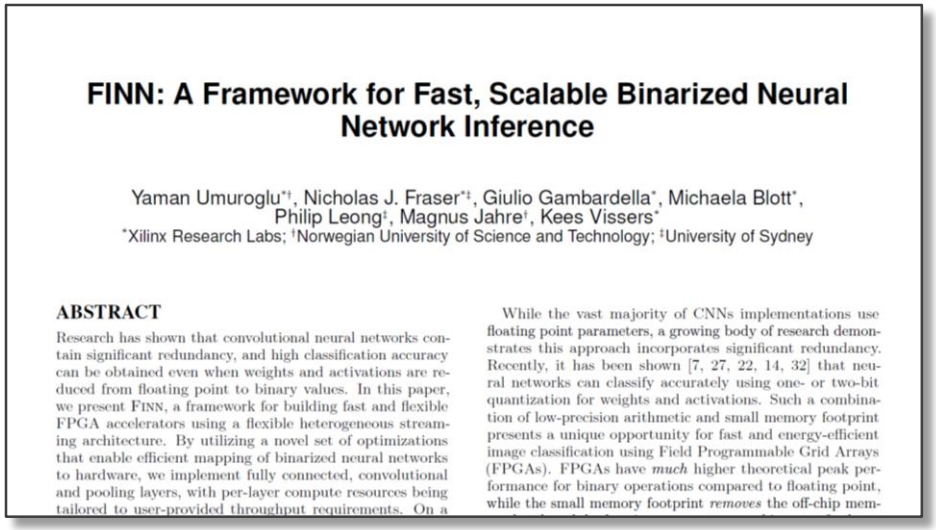
- Logistics
- Hackathon Theme & Goals
- Community Projects
- Head back up to Retreat!

Hackathons and Usergroups

Polytechnic University of Milan
100+ student hackathon

Tokyo User Group
75+ industry participants

FINN: Binary Neural Network Overlay on PYNQ



FINN: A Framework for Fast, Scalable Binarized Neural Network Inference

Yaman Umuroglu*†, Nicholas J. Fraser*‡, Giulio Gambardella*, Michaela Blott*, Philip Leong‡, Magnus Jahre†, 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 user-provided throughput requirements. On a

While the vast majority of CNNs implementations use floating point parameters, a growing body of research demonstrates this approach incorporates significant redundancy. Recently, it has been shown [7, 27, 22, 14, 32] that neural networks can classify accurately using one- or two-bit quantization for weights and activations. Such a combination of low-precision arithmetic and small memory footprint presents a unique opportunity for fast and energy-efficient image classification using Field Programmable Grid Arrays (FPGAs). FPGAs have *much* higher theoretical peak performance for binary operations compared to floating point, while the small memory footprint *removes* the off-chip mem-

Int. Symposium on FPGAs, Feb. 2017

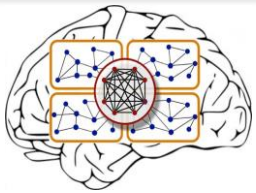
- Unprecedented image classification rates
- 1,000x speed-up over Raspberry Pi3

NTNU
Norwegian University of Science and Technology

THE UNIVERSITY OF SYDNEY

SVHN,
Road signs,
CIFAR-10

Image pre-processing in Python



Binary Neural Network in FPGA & ARM CPU

“cat”

Machine Learning on PYNQ cluster

Spark acceleration on FPGAs: A use case on machine learning in Pynq

Elias Koromilas, Ioannis Stamelos
Department of Electrical and Computer Engineering,
National Technical University of Athens
Athens, Greece

Christoforos Kachris
Institute of Communication and Computer Systems (ICCS/NTUA)
Athens, Greece

Dimitrios Soudris
Department of Electrical and Computer Engineering,
National Technical University of Athens
Athens, Greece

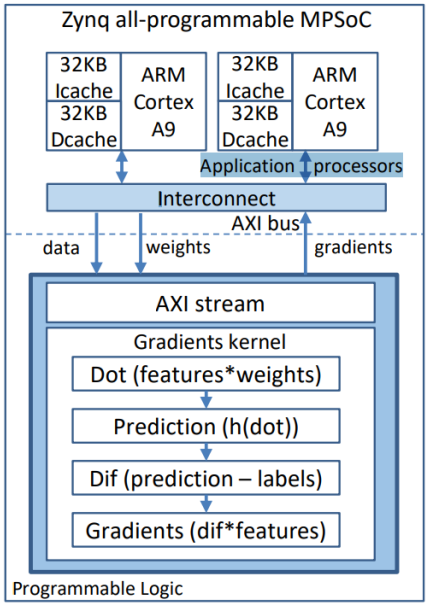
***Abstract*—Spark is one of the most widely used frameworks for data analytics. Spark allows fast development for several applications like machine learning, graph computations, etc. In this paper, we present Spynq: A framework for the efficient deployment of data analytics on embedded systems that are based on the heterogeneous MPSoC FPGA called Pynq. The mapping of Spark on Pynq allows that fast deployment of embedded and cyber-physical systems that are used in edge and fog computing. The proposed platform is evaluated in a typical machine learning application based on logistic regression. The performance evaluation shows that the heterogeneous FPGA-based MPSoC can achieve up to 11x speedup compared to the execution time in the ARM cores and can reduce significantly the development time of embedded and cyber-physical systems on Spark applications.**

powered down in order to comply with thermal constraints [3]. One way to address this problem is through the utilization of hardware accelerators. Hardware accelerators can be used to offload the processor, increase the total throughput and reduce the energy consumption.

In this paper we present a framework for the seamlessly utilization of hardware accelerators in heterogeneous SoCs that are used to speedup the processing of Spark data analytics applications.

The main contributions of this paper are the followings:

- An efficient framework for the seamlessly utilization of hardware accelerators for Spark applications



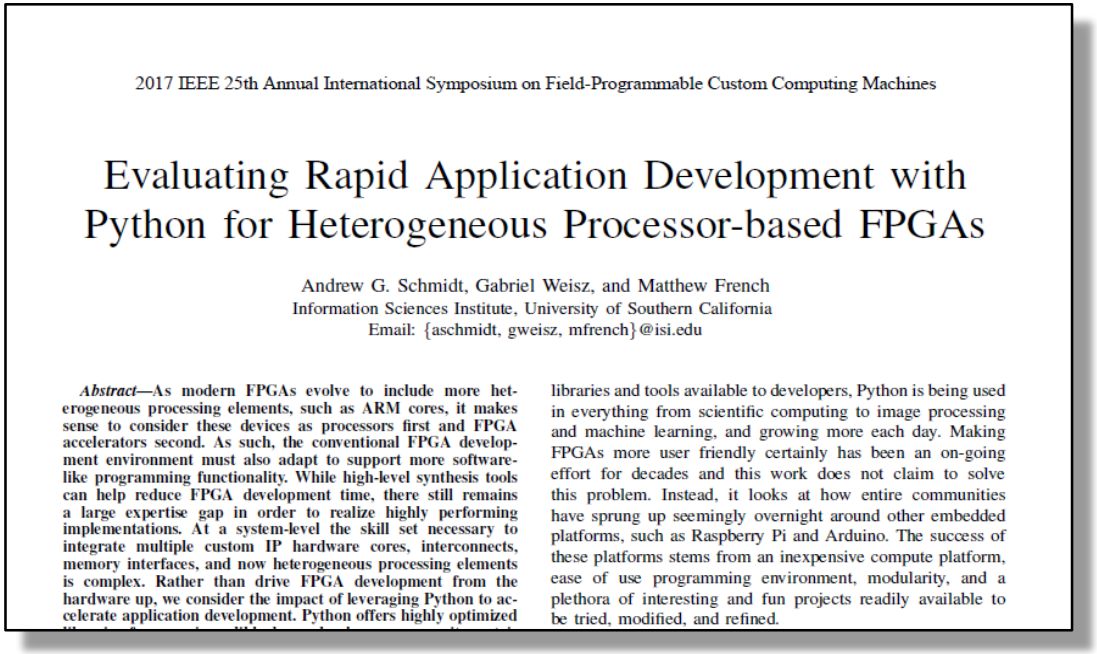
Best student paper award, MODCAST 2017

National Technical University of Athens

SPYNQ: Spark PYNQ cluster

https://www.youtube.com/watch?v=ix1NI7yq8O4

Evaluating PYNQ for image processing

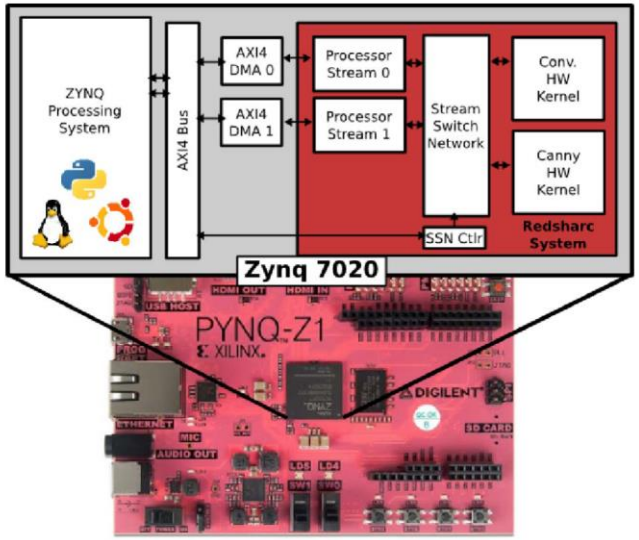


2017 IEEE 25th Annual International Symposium on Field-Programmable Custom Computing Machines

Evaluating Rapid Application Development with Python for Heterogeneous Processor-based FPGAs

Andrew G. Schmidt, Gabriel Weisz, and Matthew French
Information Sciences Institute, University of Southern California
Email: {aschmidt, gweisz, mfrench}@isi.edu

***Abstract*—As modern FPGAs evolve to include more heterogeneous processing elements, such as ARM cores, it makes sense to consider these devices as processors first and FPGA accelerators second. As such, the conventional FPGA development environment must also adapt to support more software-like programming functionality. While high-level synthesis tools can help reduce FPGA development time, there still remains a large expertise gap in order to realize highly performing implementations. At a system-level the skill set necessary to integrate multiple custom IP hardware cores, interconnects, memory interfaces, and now heterogeneous processing elements is complex. Rather than drive FPGA development from the hardware up, we consider the impact of leveraging Python to accelerate application development. Python offers highly optimized libraries and tools available to developers, Python is being used in everything from scientific computing to image processing and machine learning, and growing more each day. Making FPGAs more user friendly certainly has been an on-going effort for decades and this work does not claim to solve this problem. Instead, it looks at how entire communities have sprung up seemingly overnight around other embedded platforms, such as Raspberry Pi and Arduino. The success of these platforms stems from an inexpensive compute platform, ease of use programming environment, modularity, and a plethora of interesting and fun projects readily available to be tried, modified, and refined.**

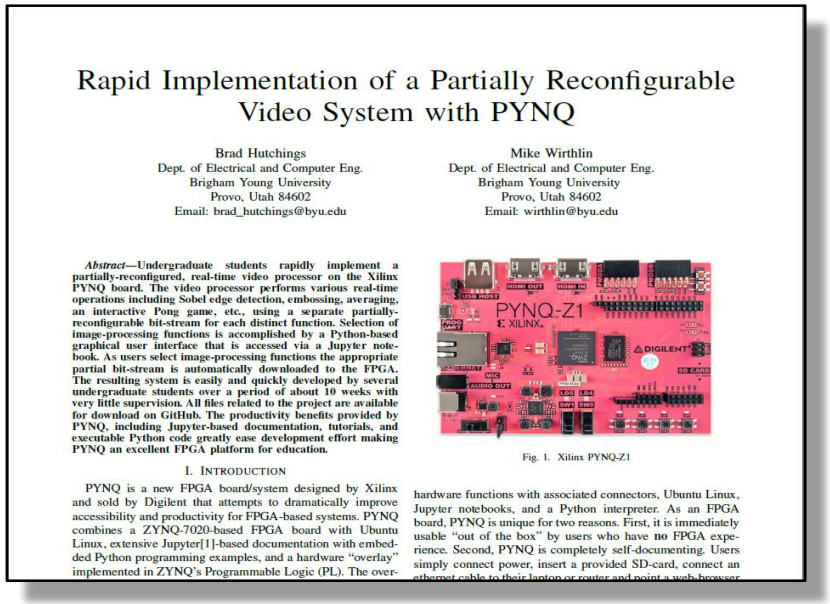


Best short paper award, FCCM 2017

USC Viterbi School of Engineering *Information Sciences Institute*

"The results are highly promising , with the ability to match and exceed performances from C implementations, up to 30x speedup."

“PYNQ is a game-changer”



Rapid Implementation of a Partially Reconfigurable Video System with PYNQ

Brad Hutchings
Dept. of Electrical and Computer Eng.
Brigham Young University
Provo, Utah 84602
Email: brad_hutchings@byu.edu

Mike Wirthlin
Dept. of Electrical and Computer Eng.
Brigham Young University
Provo, Utah 84602
Email: wirthlin@byu.edu

***Abstract*—Undergraduate students rapidly implement a partially-reconfigured, real-time video processor on the Xilinx PYNQ board. The video processor performs various real-time operations including Sobel edge detection, embossing, averaging, an interactive Pong game, etc., using a separate partially-reconfigurable bit-stream for each distinct function. Selection of image-processing functions is accomplished by a Python-based graphical user interface that is accessed via a Jupyter notebook. As users select image-processing functions the appropriate partial bit-stream is automatically downloaded to the FPGA. The resulting system is easily and quickly developed by several undergraduate students over a period of about 10 weeks with very little supervision. All files related to the project are available for download on GitHub. The productivity benefits provided by PYNQ, including Jupyter-based documentation, tutorials, and executable Python code greatly ease development effort making PYNQ an excellent FPGA platform for education.**

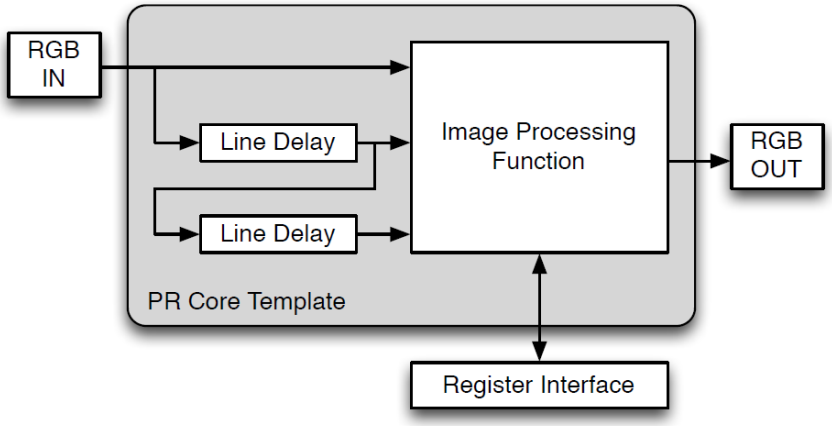
Fig. 1. Xilinx PYNQ-Z1

I. Introduction

PYNQ is a new FPGA board/system designed by Xilinx and sold by Digilent that attempts to dramatically improve accessibility and productivity for FPGA-based systems. PYNQ combines a ZYNQ-7020-based FPGA board with Ubuntu Linux, extensive Jupyter[1]-based documentation with embedded Python programming examples, and a hardware “overlay” implemented in ZYNQ’s Programmable Logic (PL). The overhardware functions with associated connectors, Ubuntu Linux, Jupyter notebooks, and a Python interpreter. As an FPGA board, PYNQ is unique for two reasons. First, it is immediately usable “out of the box” by users who have **no** FPGA experience. Second, PYNQ is completely self-documenting. Users simply connect power, insert a provided SD-card, connect an ethernet cable to their laptop or router and point a web-browser

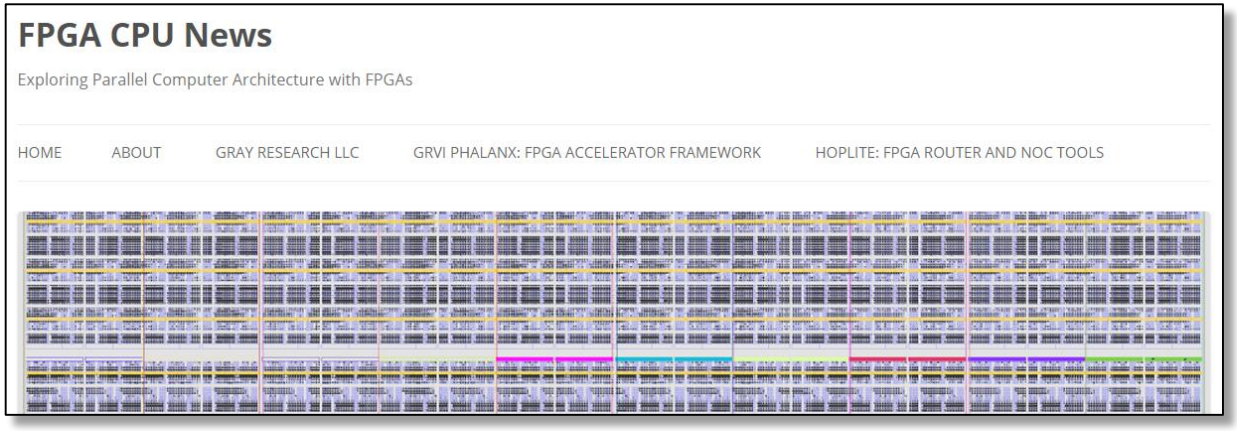
FPL2017

Partially reconfigurable video overlay



“After teaching students how to use FPGA-based systems for over 20 years, this is the first time we simply handed the boards to the students and **all** of them were able to use and/or add functionality to the system with no supervision.”

“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: PYTHON PRODUCTIVITY ON ZYNQ

Home | Get Started | PYNQ-Z1 Bo

Community Projects

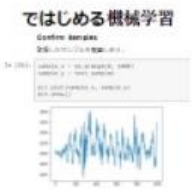
Selection of projects and notebooks

A selection of projects from the PYNQ community is shown below. Note that some examples are built on different versions of the PYNQ image.

Binary Neural Network
Xilinx Labs, NTNU, University Sydney

SPynq:
NTUA Greece

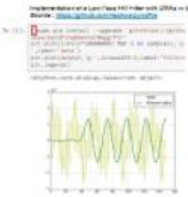
Processing noisy filters
PYNQ Japan user group



Soft GPU for ZC706
Ruhr University Bochum

Video Processing
Vectorblox

FIR filter example
CU Boulder



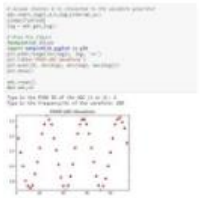
DMA and stream Tutorial

CNN Example
Imperial College London

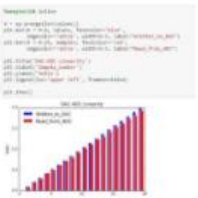
Example Notebooks

A selection of notebook examples are shown below that are included in the PYNQ image. The notebooks contain live code, and generated output from the code can be saved in the notebook. Notebooks can be viewed as webpages, or opened on a Pynq enabled board where the code cells in a notebook can be executed.

ADC waveforms



DAC ADC example



Downloading overlays

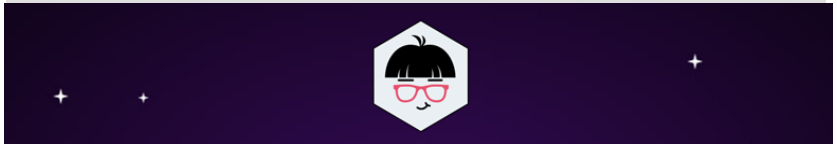
Grove ADC

Group Participation!

- Please introduce yourself
  - Name
  - School / Company
  - Interests
  - Do you need a team?

XILINX
ALL PROGRAMMABLE™

Welcome and Orientation



- Logistics
- Hackathon Theme & Goals
- Community Projects
- Head back up to Retreat!