InnovateFPGA

PipeCNN: An OpenCL-Based FPGA Accelerator for Convolution Neural Network

Student : Jianjing An and Diankun Jiang

Teacher : Dong Wang

Team Num: PR022

Institute of Information Science Beijing Jiaotong University

Email: {wangdong, 16112065, 16125141}@bjtu.edu.cn

PipeCNN —— Introduction

- **PipeCNN**

  PipeCNN is an OpenCL-based FPGA Accelerator for Large-Scale Convolutional Neural Networks (CNNs). There is a growing trend among the FPGA community to utilize High Level Synthesis (HLS) tools to design and implement customized circuits on FPGAs.

- **Key Features**
  - A completed OpenCL kernel sets for CNN forward computations
  - A generic design, efficient and scalable in performance and cost
  - Optimization Design
    - 8-bit fixed-point Design
    - Mixed window/line-buffer caching scheme

PipeCNN —— Top-Level Architecture

- **Top-Level Architecture**
  - CNN running on deeply pipelined kernels using Channel/Pipe in OpenCL
  - Use a single hardware kernel to implement both the convolution and FC layers

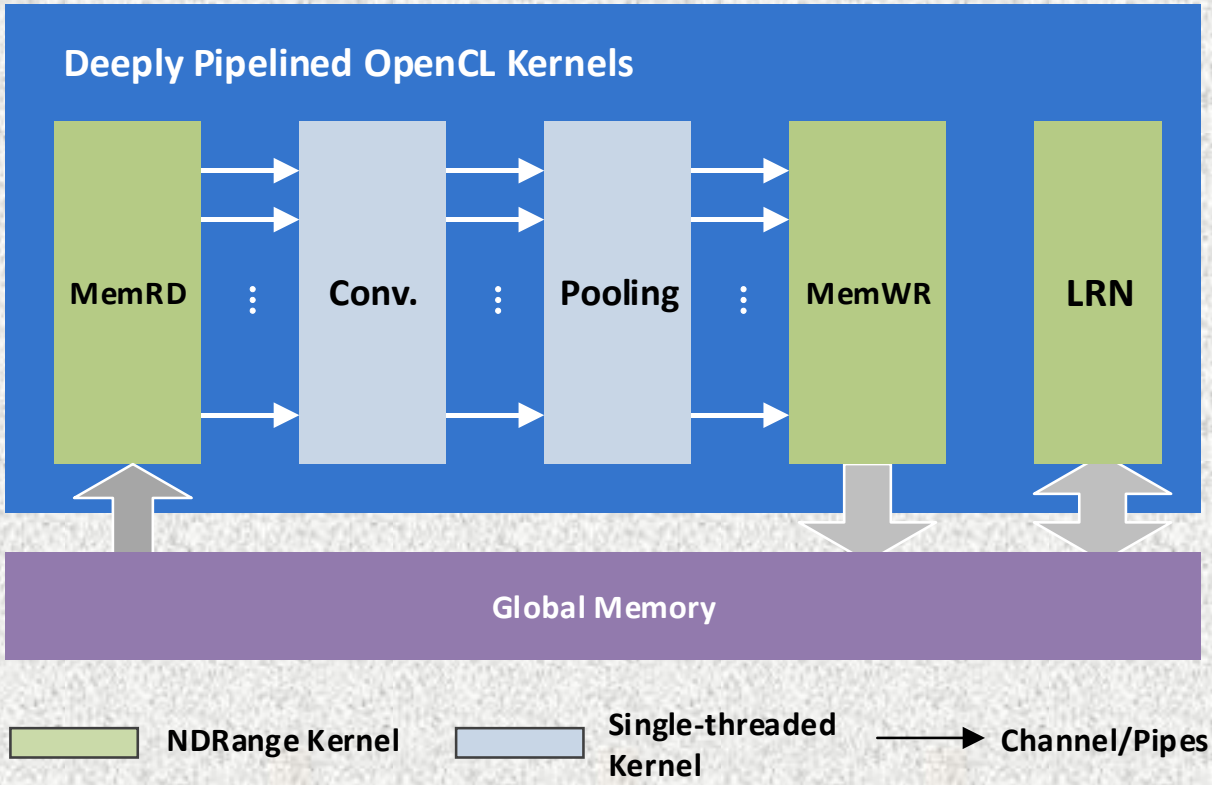


Fig1. The top-level architecture of PipeCNN.

PipeCNN —— Convolution and FC Layers

- **Convolution:**

$$D_0(f_o, y, x) = \sum_{f_i=0}^{C_l} \sum_{k_y=0}^{K} \sum_{k_x=0}^{K} W_l(f_0, f_i, k_y, k_x) \cdot D_i(f_i, y+k_y, x+k_x)$$

- **Inner-product:**

$$D_o(f_0) = \sum_{f_i=0}^{C_l} W_l(f_0, f_i) \cdot D_i(f_i)$$

- **Unified formula:**

$$D_0(f_o, y, x) = \sum_{f_i=0}^{C_l} \sum_{n=0}^{N} W_l(f_0, f_i, n) \cdot D_i(f_i, n) \quad (\mathrm{N} = \mathrm{K} \times K \text{ or } 1)$$

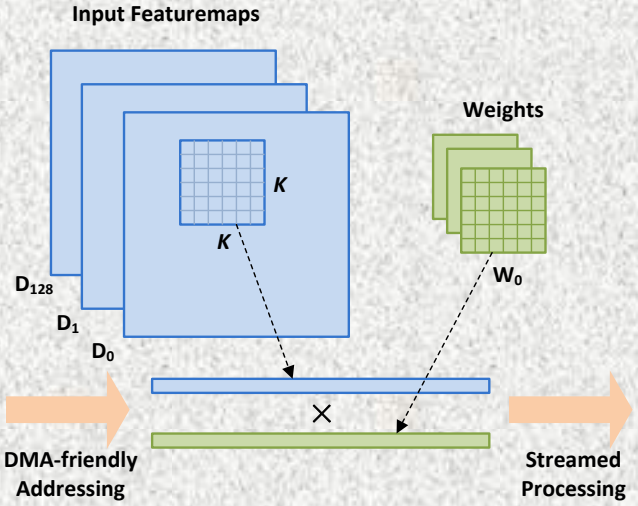


Fig 2. Transform 3D Conv. into 1D accumulation

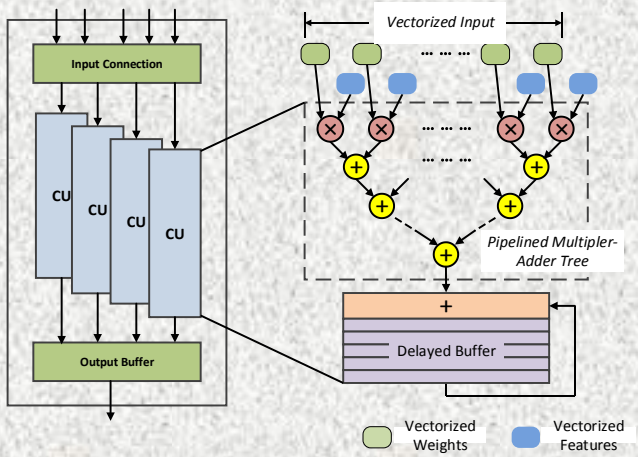


Fig 3. OpenCL-Modeled Accumulation Circuit

PipeCNN —— Data Mover Kernels

- **Improving Throughput and Minimizing BW requirements**
  - Vectorizing feature map and weight
  - Utilizing on-chip cache and reusing data in multiple CUs

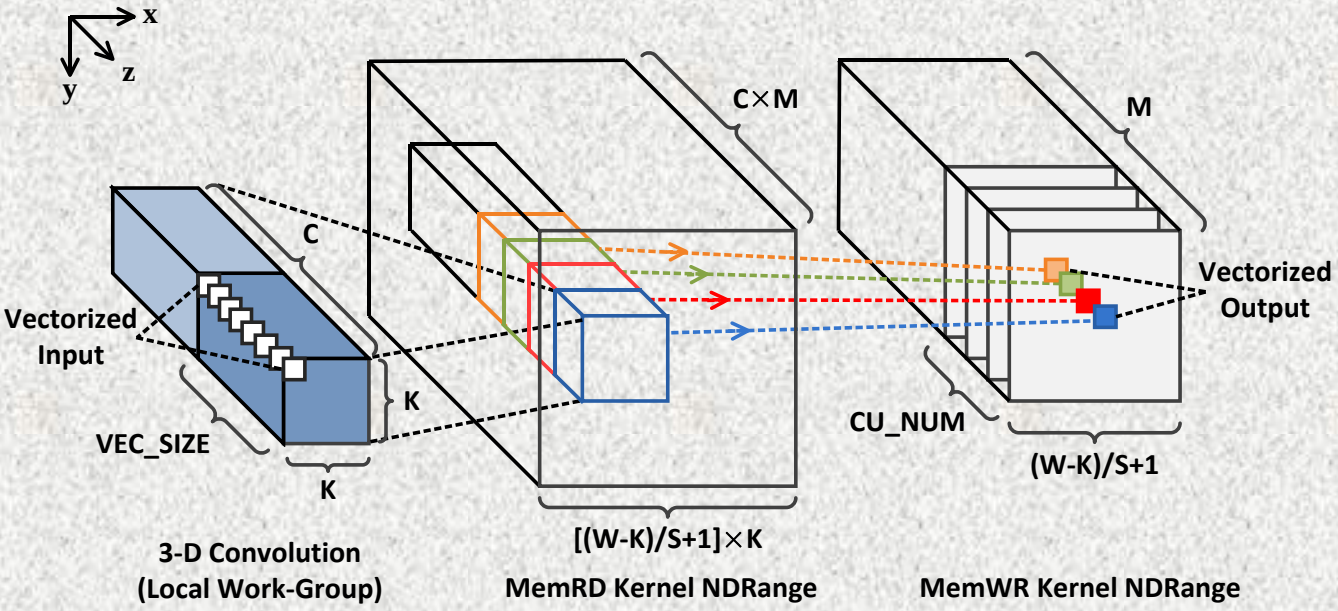


Fig 4. Data vectorization and reuse in the NDRange

PipeCNN —— Current Status

- **Current Status**
  - **A kernel set supporting state-of-the-art CNN**
    - Convolution/FC/pooling/LRN/BN/Relu/Sigmoid/Softmax
    - AlexNet/VGG/NIN/SqueezNet/GoogleNet/ResNet verified
  - **Tested on main stream FPGA boards**
    - Arria-10 (high-end), Stratix-V, Cyclone-V(low-cost)

PipeCNN —— Demonstration1

Imagenet Classification

- Accuracy: Table1
- Speed: 110ms on DE10-NANO platform

Table1 The comparison of AlexNet model classification accuracy

Accuracy	Top-1	Top-5
Full precision(32 bit)	56.8%	79.8%
This work(8 bit)	56.2%	79.5%

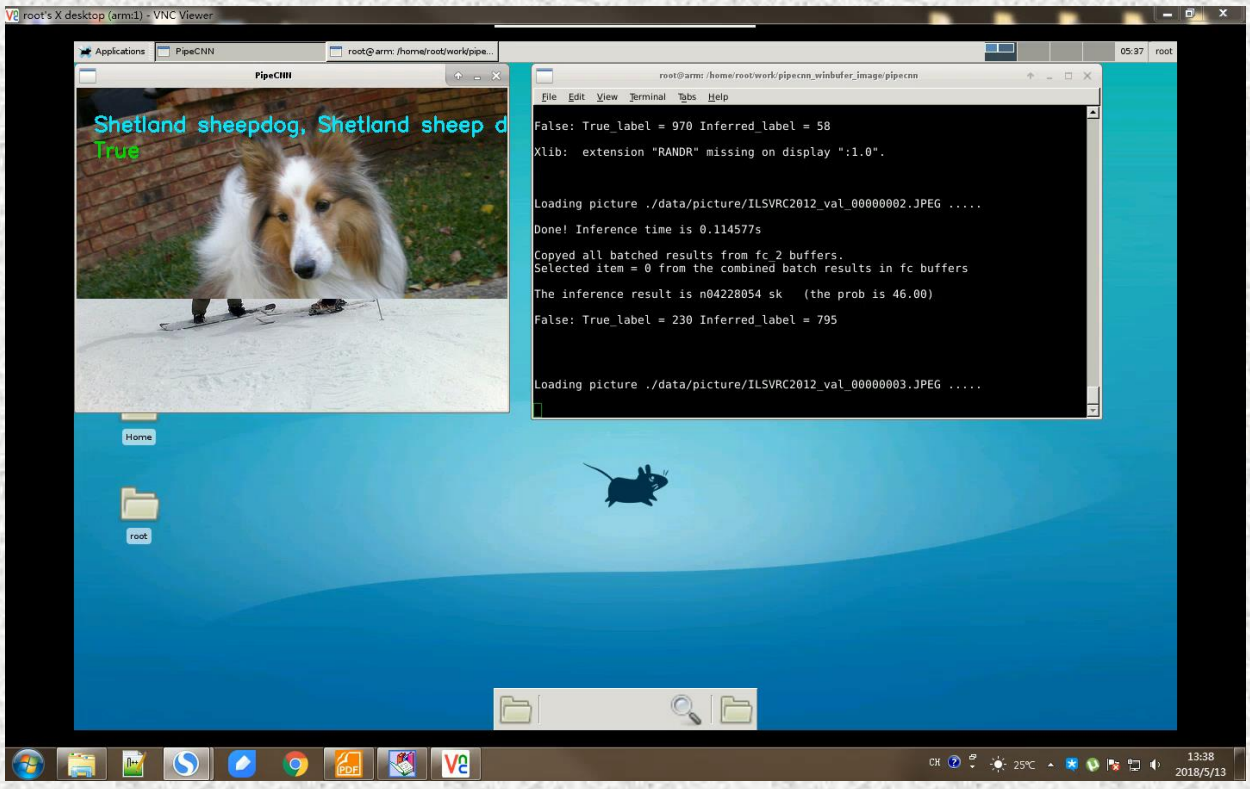


Fig 5. Imagenet Classification on Alexnet

PipeCNN —— Demonstration2

Object Recognition via Camera

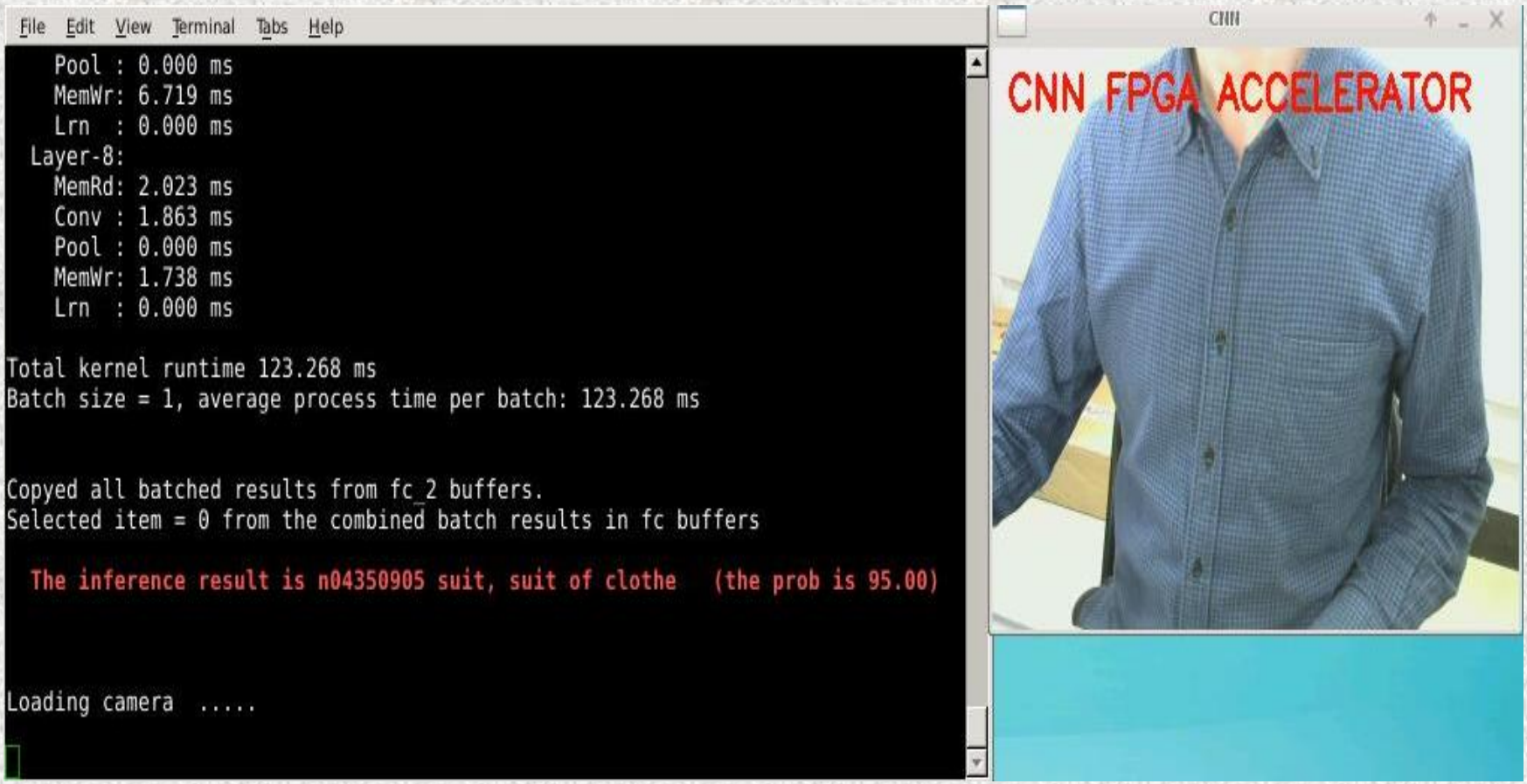


Fig 6. Object recognition via camera on Alexnet

PipeCNN —— Demonstration3

Face Recognition

Datasets: LFW

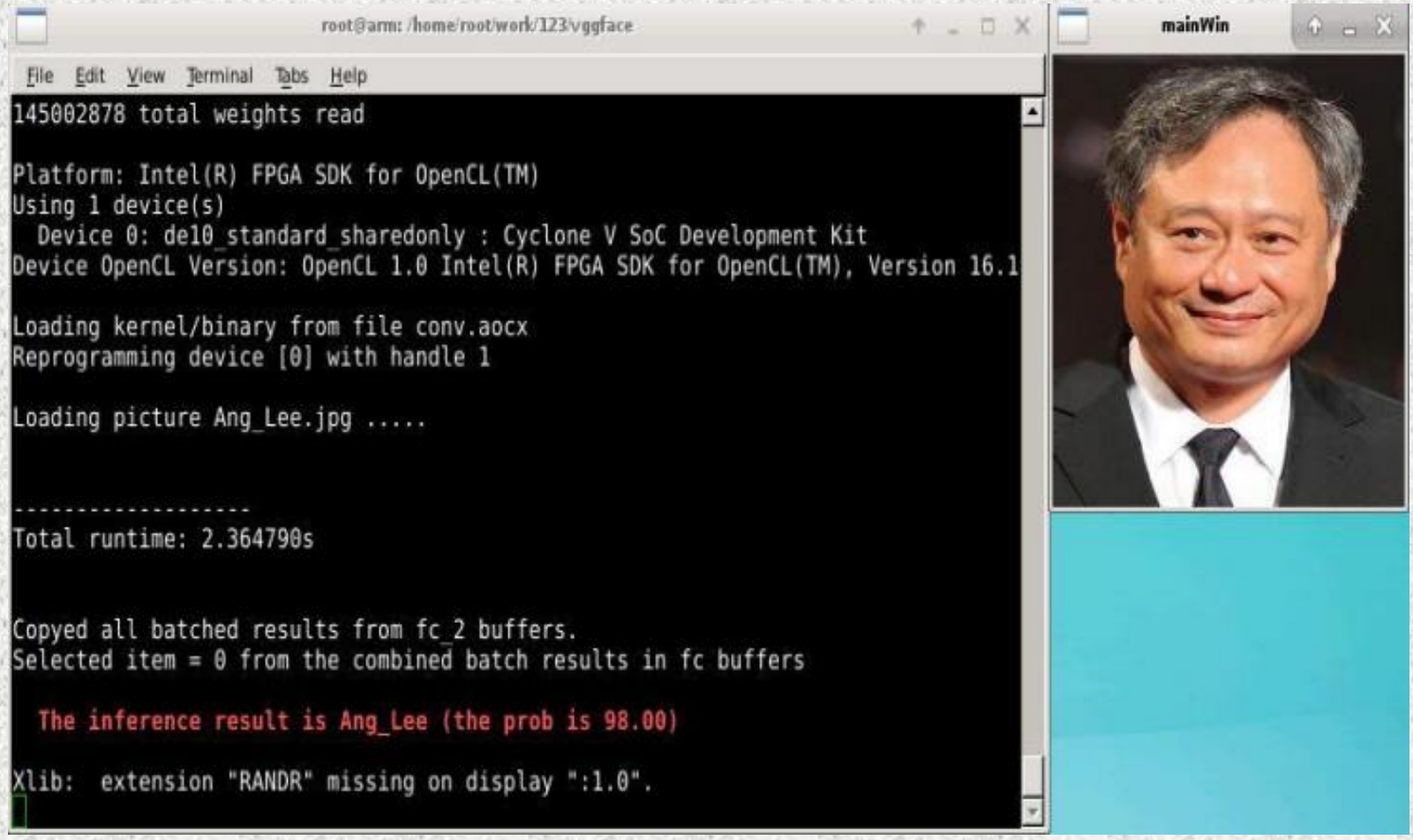
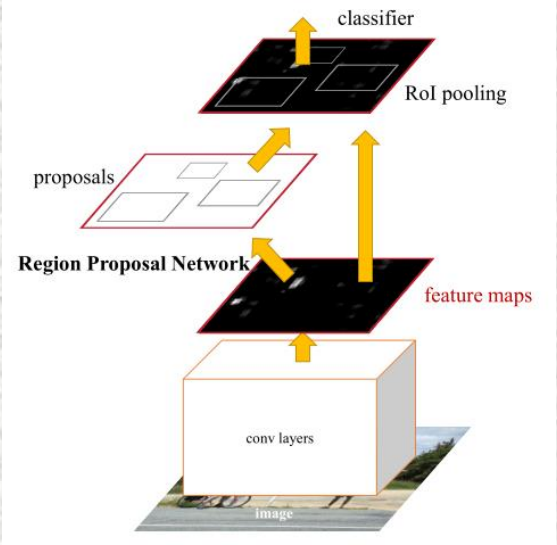


Fig 7. Face recognition on Vgg16

PipeCNN —— Demonstration4

Object Detection



- Full precision mAP: 56.2
- 8-bit mAP: 54.5

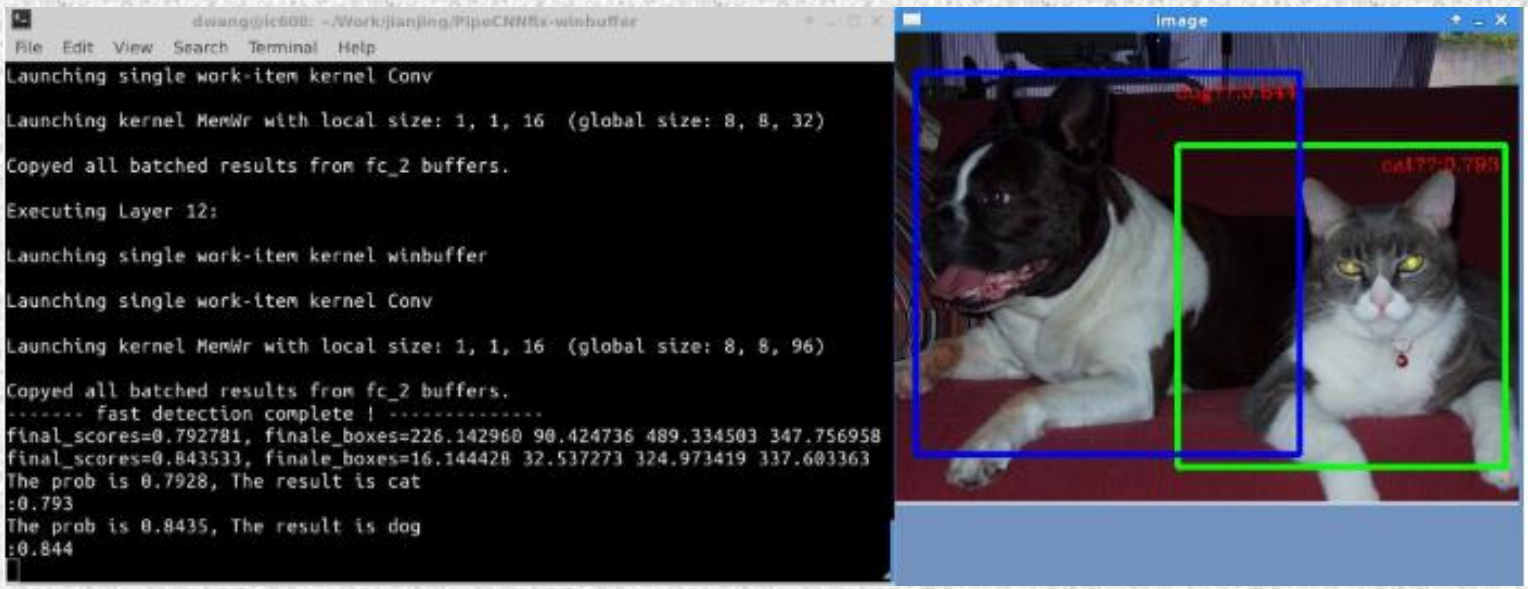
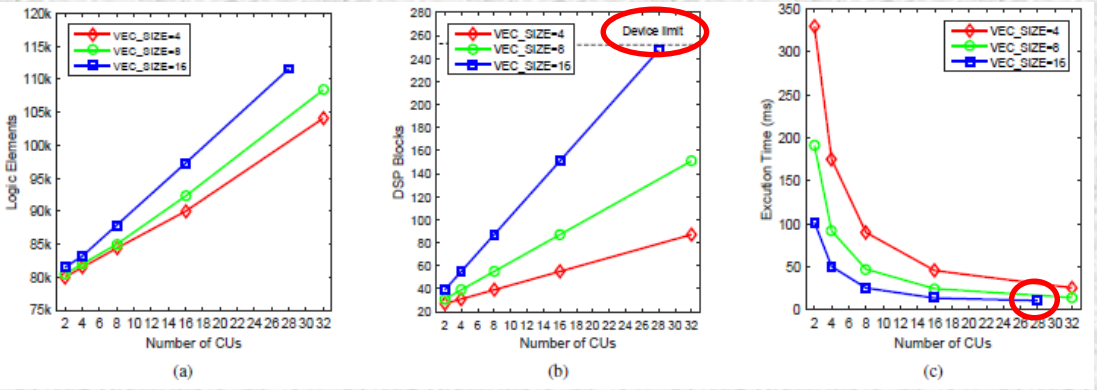


Fig 8. Object Detection based on Faster R-CNN(Alexnet)

PipeCNN —— Results

Design Space Exploration

Optimal Resource Utilization



Highest Throughput

Fig. 9 Design space exploration for AlexNet model on Stratix-V A7 FPGA board. CU denotes compute units, and VEC_SIZE represents the degree of data parallelism utilized. (a) Logic elements utilization; (b) DSP blocks utilization; (c) Inference time.

Resource utilization

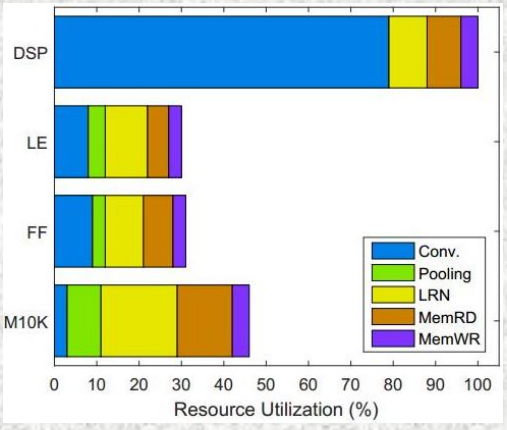


Fig .10 Resource utilization of each kernel for AlexNet model

PipeCNN —— Results

Comparison with software accelerators on mobile CPU/GPU*

Table. 2 Summary of the measured performance and power consumption on different platforms

Platform	Frequency	Inference Time [b]	Effective Power [c]	System Power [d]
ARM Cortex [a] A57/A53 CPU	1.9 Ghz (A57) 1.3 Ghz (A53)	20,767 ms	2.4 W	4.1 W
Mali-T760 GPU	700 Mhz	482 ms	0.52 W	2.3 W
Cyclone A5 SoC-FPGA	800 Mhz (CPU) 140 Mhz (FPGA)	110 ms	0.5 W	2.1 W

[a] Samsung Galaxy Note 4 (Exynos 5433)
[b] AlexNet benchmark was used.
[c] Effective power = total power - standby power
[d] Measured by using external power meter with screen turned off

* Oskouei S S, Golestani H B, Hashemi M. CNNdroid: GPU-Accelerated Execution of Trained Deep Convolutional Neural Networks on Android, ACM Conference on Multimedia 2016.

PipeCNN —— Results

- **Comparison with HLS/OpenCL-based designs**

Table. 2 Summary of the measured performance and power consumption on different platforms

	FPGA2015	FPGA2016	FPGA2017	Our Work
Device	Virtex-7 VX485T(28nm)	Stratix-V GXA7(28nm)	Arria-10 AX1150(20nm)	Stratix-V GXA7(28nm)
FPGA Capacity	485K LUTs 2,800 DSPs	622K LEs 256 DSPs	1,150K LEs 2,800 DSPs	622K LEs 256 DSPs
Frequency	100MHz	120MHz	303MHz	200MHz
Precision	Float(32b)	fixed(8b-16b)	float(16b)	Fixed(8b)
Inference Time [a]	21.6 ms[b]	45.7 ms	0.98 ms	10.5 ms
Throughput	61.6 GOPS[b]	31.8 GOPS	1,382 GOPS	133.2 GOPS
DSP Consumed	2,240	246	1,476	247
Perf. Density (GOPS/DSP/GHz/W)	0.015	0.042	0.068	0.103
Power	18.6 W	25.8 W	45 W	26.2 W

4 × Improvement

1.5x

[a] alexNet model is used.
[b] Convolution operation only.

PipeCNN —— Open-Source

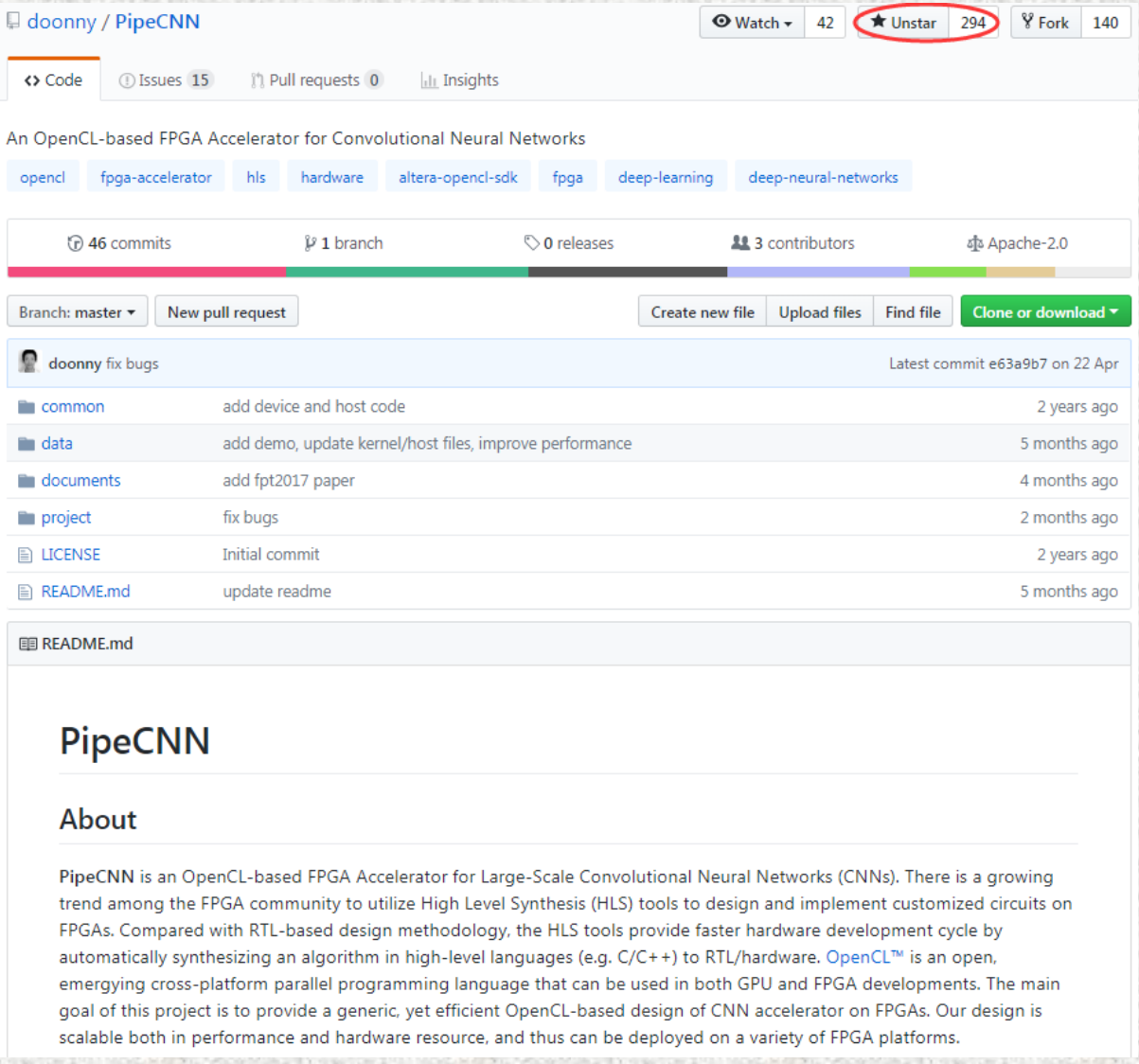


Fig. 11 open-source PipeCNN github

PipeCNN

https://github.com/doonny/PipeCNN

Thanks ~

Backgrounds

- **The Advantages of OpenCL**
  - Cross Vendor/Architecture/Device Support
    - Xilinx, Altera, Intel, AMD, Nvidia, ARM, TI
    - FPGA, CPU, GPU, DSP, Many-Core
  - High-Level Programming Language/Interface
    - C, C++, Python, Java
  - Fast Design, Verification, Test
    - From months to hours
  - Ecosystem
    - clBlas, clFFT, clSPARSE, TensorFlow, Caffe
  - Integrated with RTL-based Flow for FPGA
    - Wrap RTL modules as kernel functions