VLSI System Design Project

Face Detection Accelerator

Taiwan No.1
沈育同、施鈞陽、謝炎璋、林佑恩、王得祐、温偉程

Outline

- Introduction
- Method
- Architecture
- Timing Analysis

Outline

- Introduction
- Method
- Architecture
- Timing Analysis

Introduction

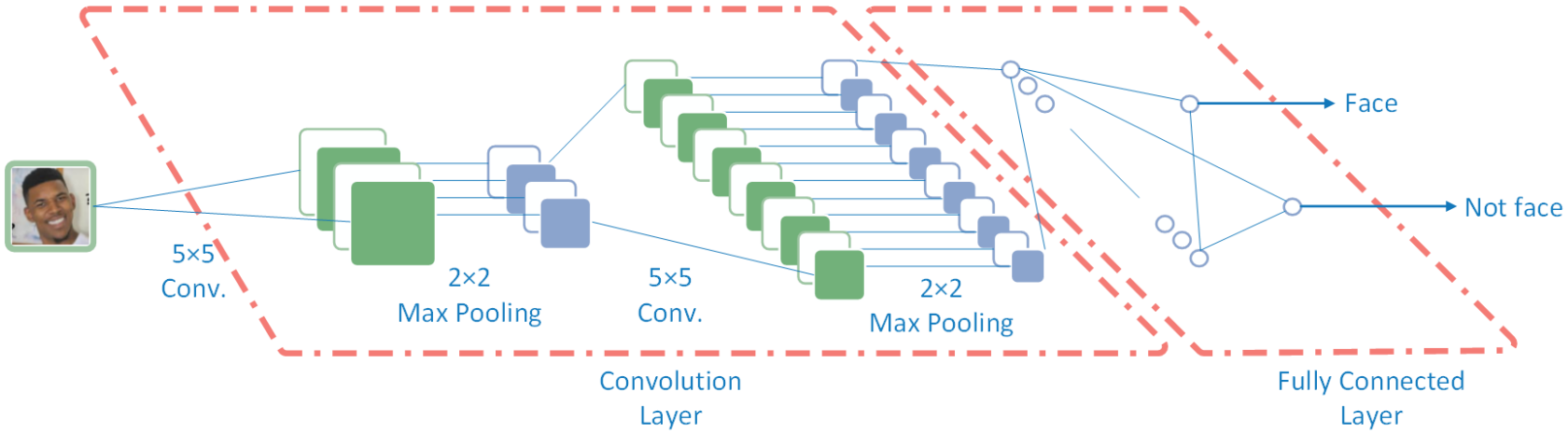
- Face recognition application
  - Photo Auto-Tagging (Facebook)
  - 天網 (China)

Introduction

- How to detect faces ?

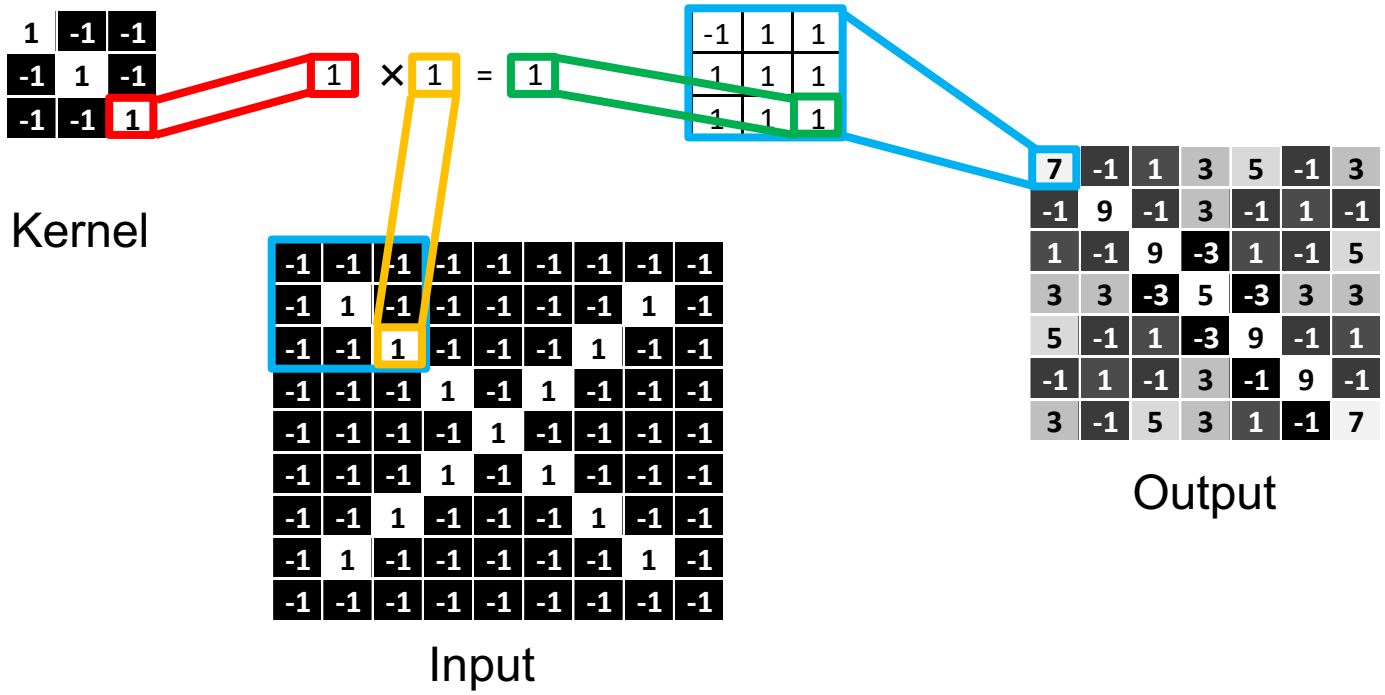
Introduction

- How to detect faces ?



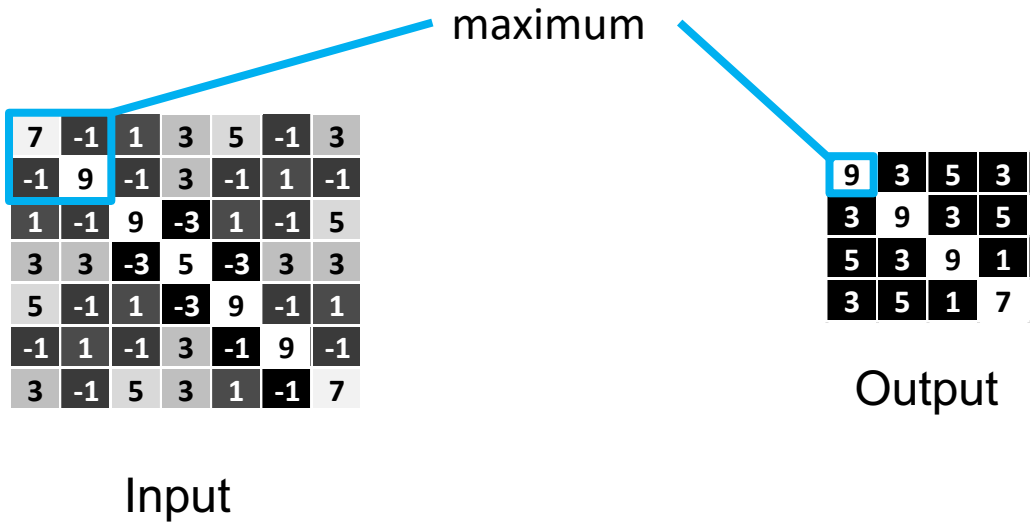
Introduction

- Convolution



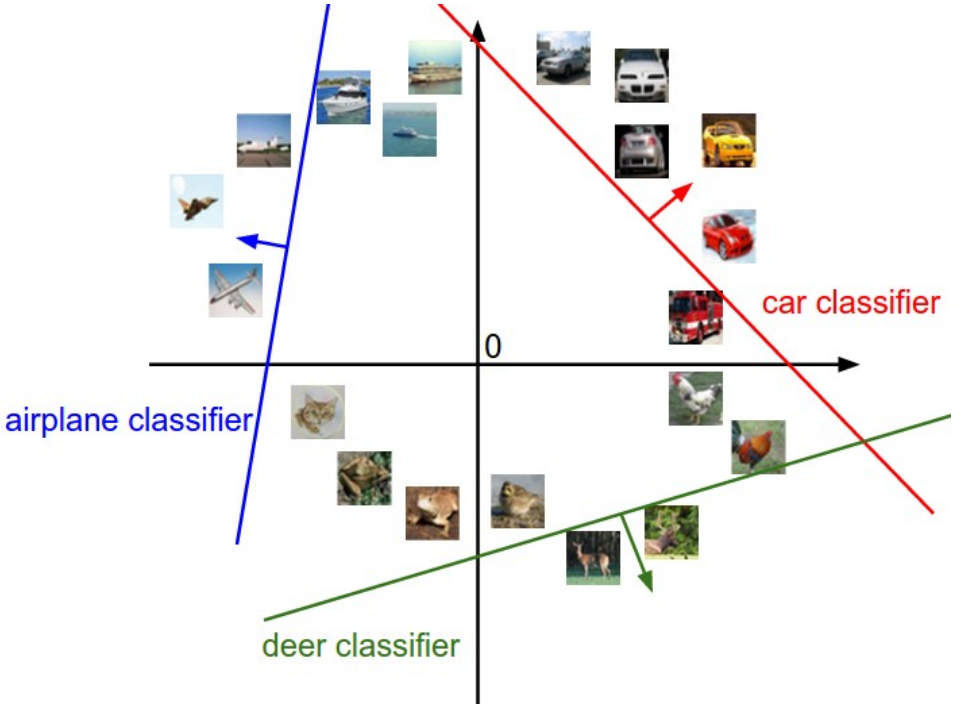
Introduction

- Pooling



Introduction

- Classification



Outline

- Introduction
- Method
- Architecture
- Timing Analysis

Method

- Data Set : LFW

Method

- Using Keras to train module

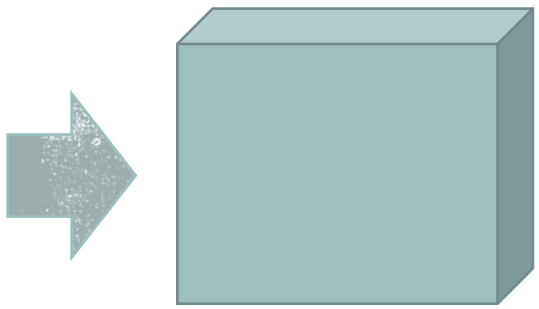
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 60, 60, 2)         52
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 30, 30, 2)         0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 26, 26, 4)         204
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 13, 13, 4)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 676)               0
_________________________________________________________________
dense_1 (Dense)              (None, 2)                 1354
=================================================================
Total params: 1,610
Trainable params: 1,610
Non-trainable params: 0
_________________________________________________________________
```

```
21172/21172 [==============================] - 2s 105us/step - loss: 0.3900 - acc: 0.8169 - val_loss: 0.4706 - val_acc: 0.7864
Epoch 2/10
21172/21172 [==============================] - 1s 45us/step - loss: 0.1669 - acc: 0.9410 - val_loss: 0.2745 - val_acc: 0.8929
Epoch 3/10
21172/21172 [==============================] - 1s 51us/step - loss: 0.1080 - acc: 0.9633 - val_loss: 0.1250 - val_acc: 0.9573
Epoch 4/10
21172/21172 [==============================] - 1s 59us/step - loss: 0.0806 - acc: 0.9739 - val_loss: 0.1328 - val_acc: 0.9547
Epoch 5/10
21172/21172 [==============================] - 1s 52us/step - loss: 0.0616 - acc: 0.9812 - val_loss: 0.1300 - val_acc: 0.9547
Epoch 6/10
21172/21172 [==============================] - 1s 40us/step - loss: 0.0555 - acc: 0.9820 - val_loss: 0.1219 - val_acc: 0.9566
Epoch 7/10
21172/21172 [==============================] - 1s 52us/step - loss: 0.0490 - acc: 0.9839 - val_loss: 0.0348 - val_acc: 0.9889
Epoch 8/10
21172/21172 [==============================] - 1s 44us/step - loss: 0.0393 - acc: 0.9881 - val_loss: 0.0844 - val_acc: 0.9683
Epoch 9/10
21172/21172 [==============================] - 1s 62us/step - loss: 0.0348 - acc: 0.9896 - val_loss: 0.0527 - val_acc: 0.9815
Epoch 10/10
21172/21172 [==============================] - 1s 43us/step - loss: 0.0283 - acc: 0.9923 - val_loss: 0.0371 - val_acc: 0.9883
```

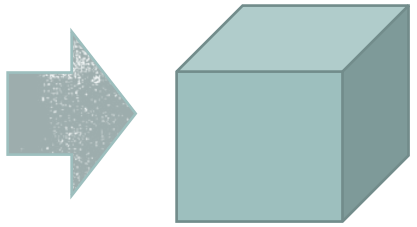
Method

- Inference

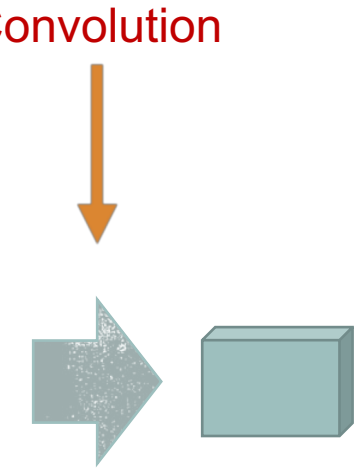
Shape = (600, 800, 1)



Shape = (298, 398, 2)



Shape = (147, 197, 4)

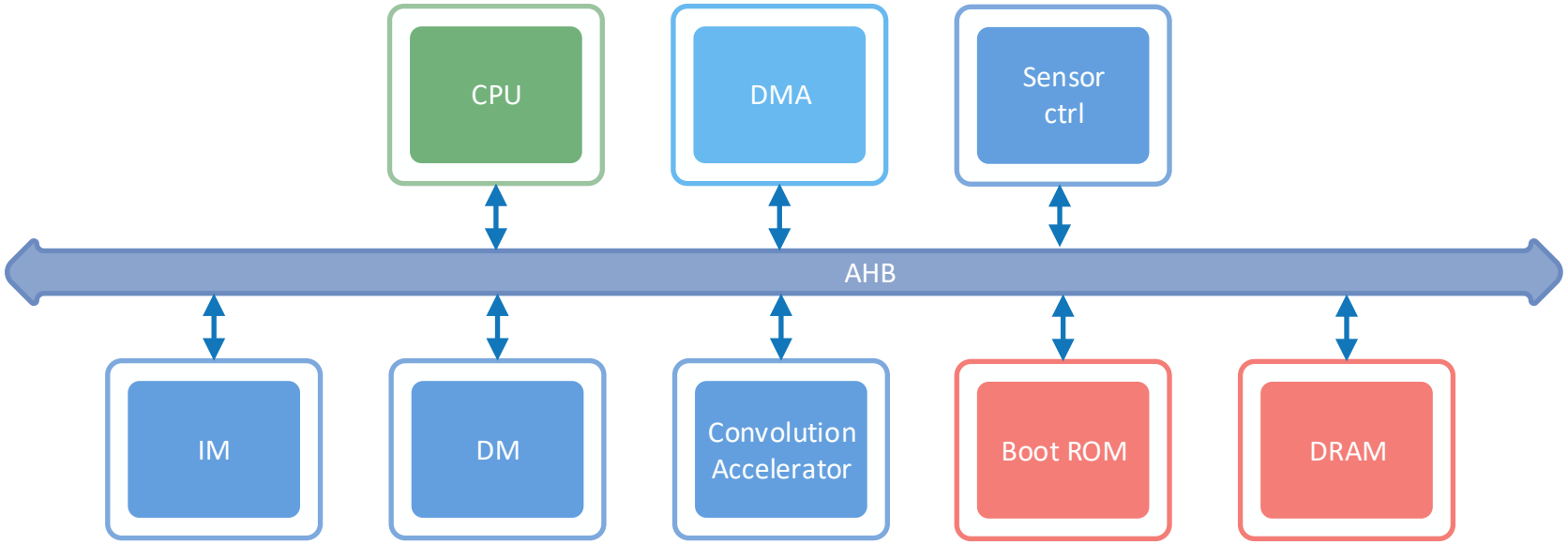


Shape = (17, 24, 2)

Outline

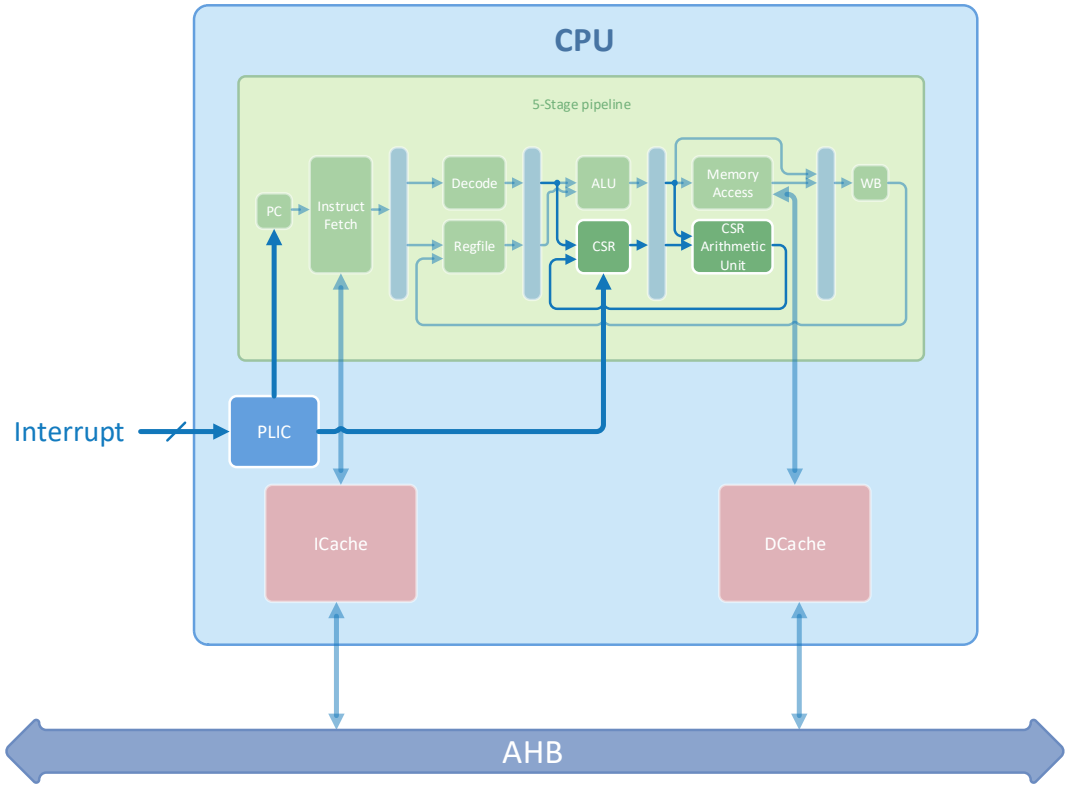
- Introduction
- Method
- Architecture
- Timing Analysis

Architecture



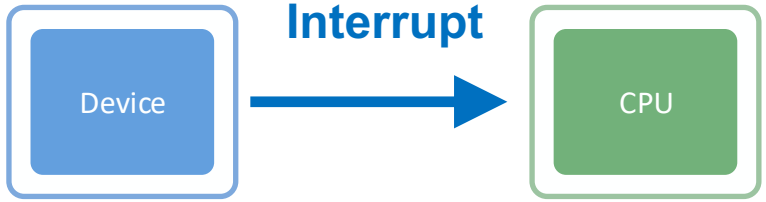
Base address	Size	Cacheable	Device
0x0000 0000	8 kbytes	Yes	ROM
0x0001 0000	32 kbytes	Yes	IM
0x0002 0000	32 kbytes	Yes	DM
0x4000 0000	32 bytes	No	DMA configuration register
0x2000 0000	1 Mbytes	Yes	DRAM
0x1000 0000	256 bytes	No	Sensor_ctrl
0x0800 0000	160 kbytes	No	Conv_wrapper (contains memory and configuration register)

Architecture - CPU



Architecture - CPU

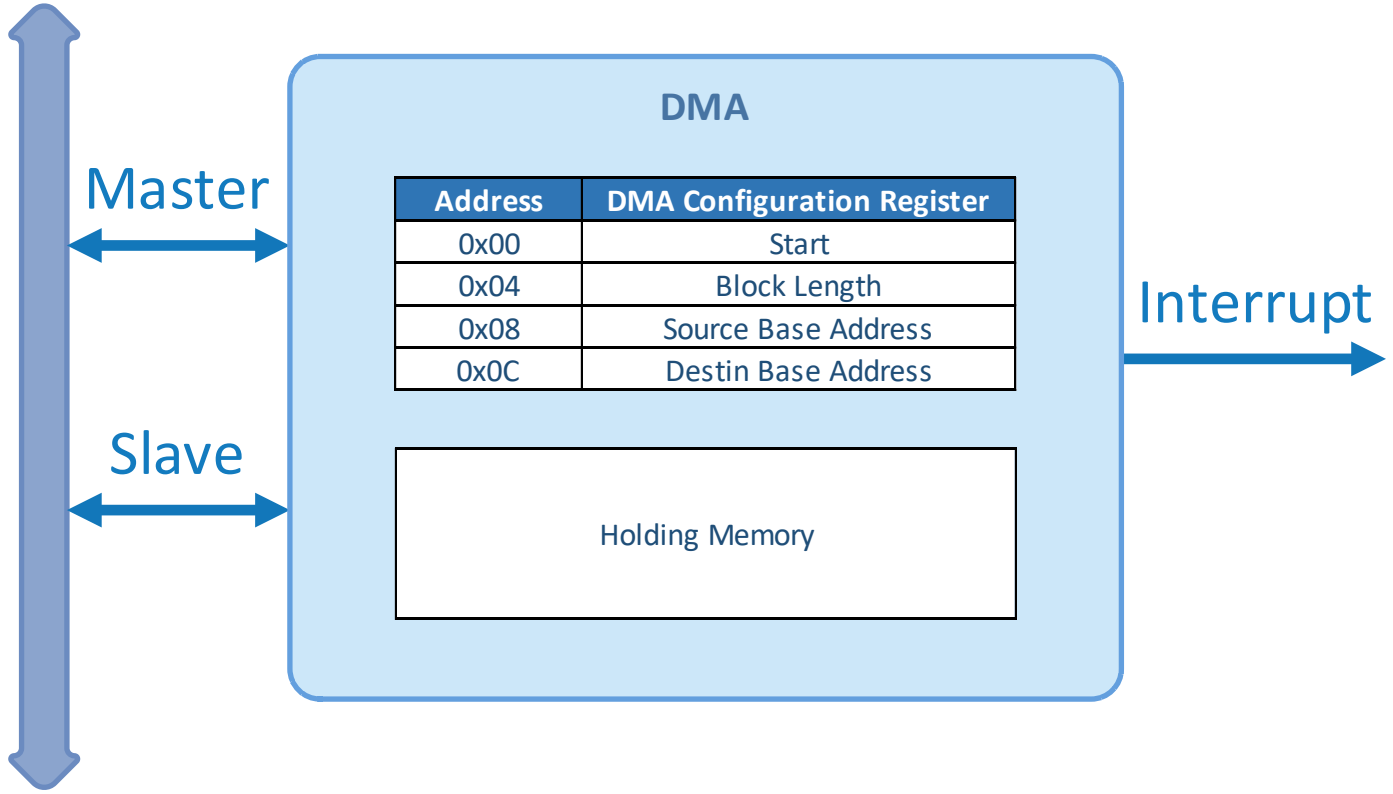
- Interrupt



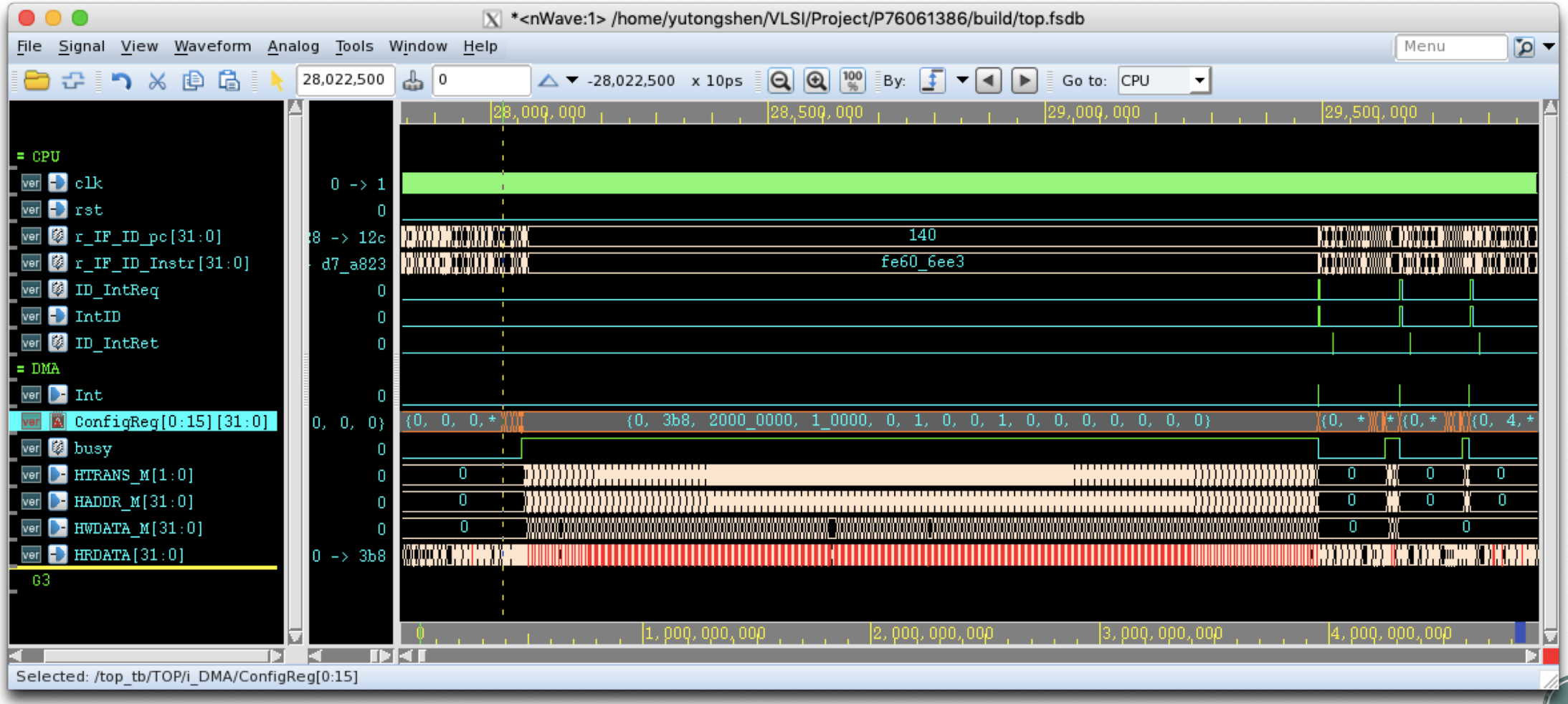
Address	Interrupt Vector Table
0x0001 0000	Sensor_ctrl
0x0001 0004	DMA

```
# Define constants
.section .text
.align 2
.globl trap_entry
trap_entry:
  j sensor_ctrl
  j dma_isr
sensor_ctrl:
    addi sp, sp, -4
    sw x1, 0(sp) # save ra
    jal savereg
    jal copy
    jal loadreg
    lw x1, 0(sp) # load ra
    addi sp, sp, 4
    mret
dma_isr:
    addi sp, sp, -4
    sw x1, 0(sp) # save ra
    jal savereg
    jal _dma_isr
    jal loadreg
    lw x1, 0(sp) # load ra
    addi sp, sp, 4
    mret
```

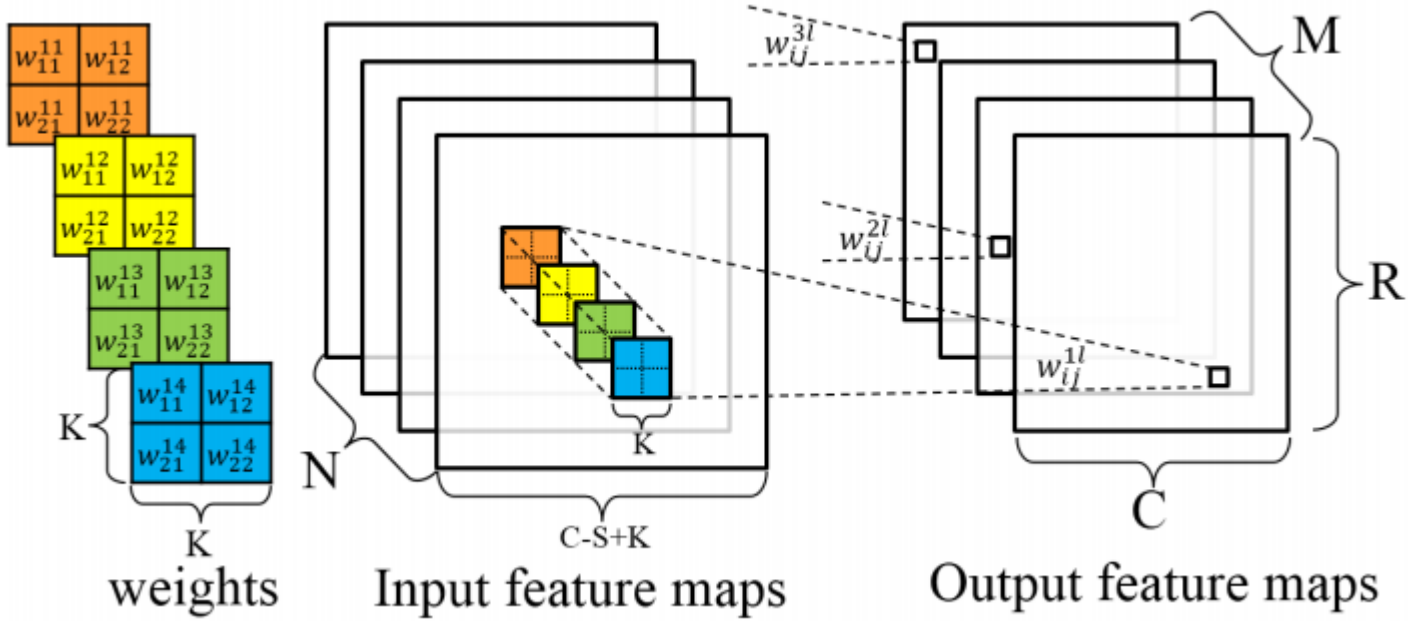
Architecture - DMA



Architecture - DMA



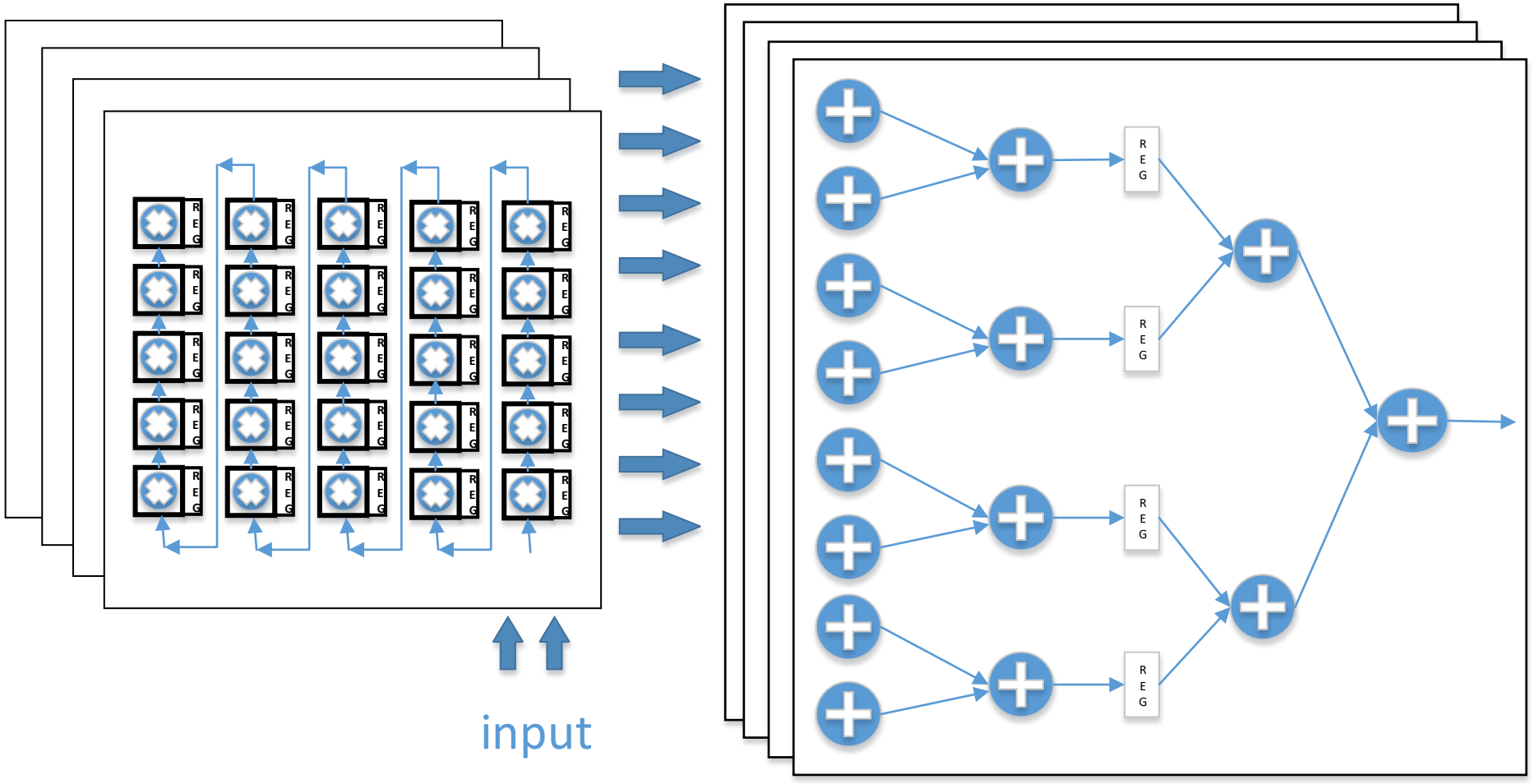
Architecture – Convolution Accelerator



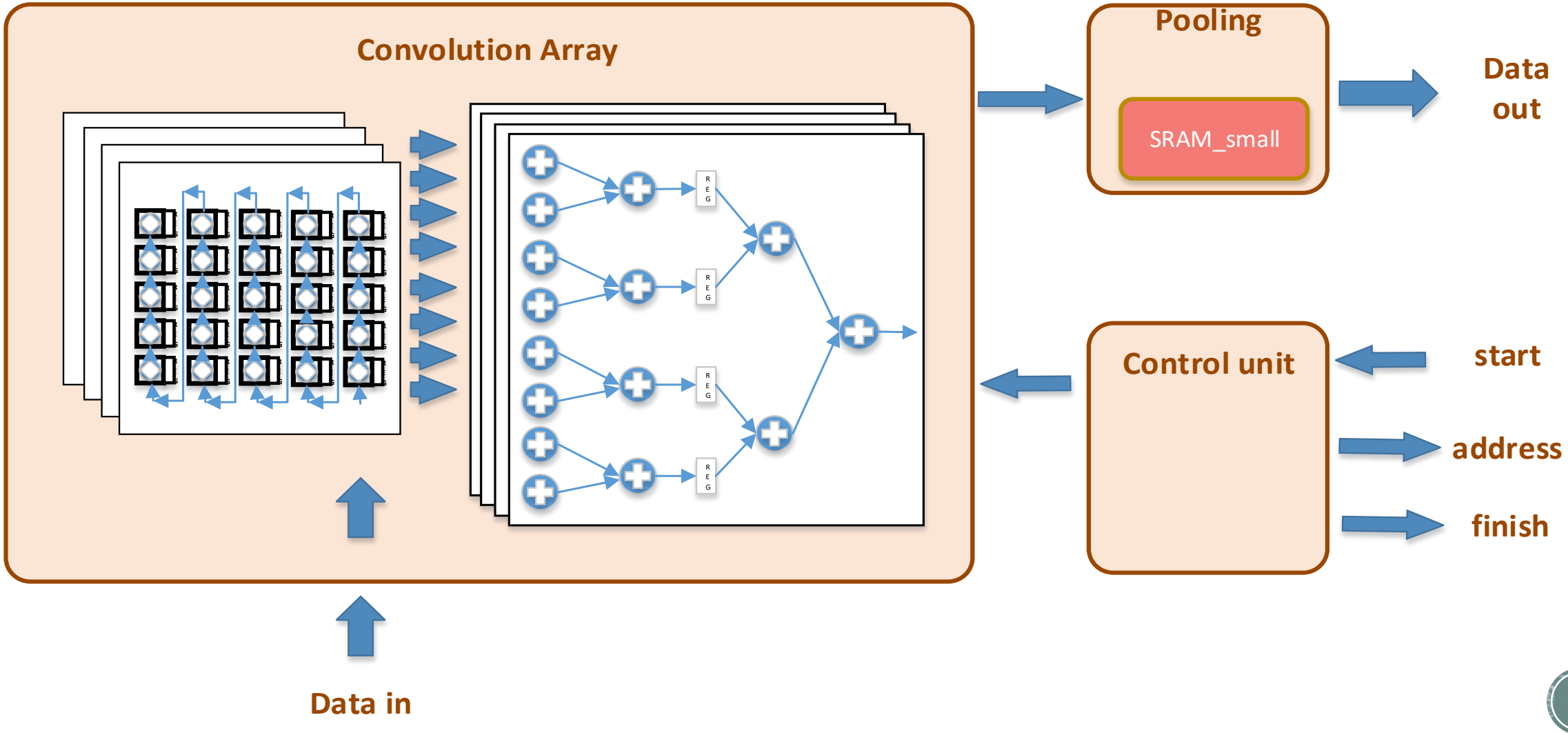
- Convolution computing

ref: C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao, and J. Cong, “Optimizing fpga-based accelerator design for deep convolutional neural networks,” in FPGA, 2015.

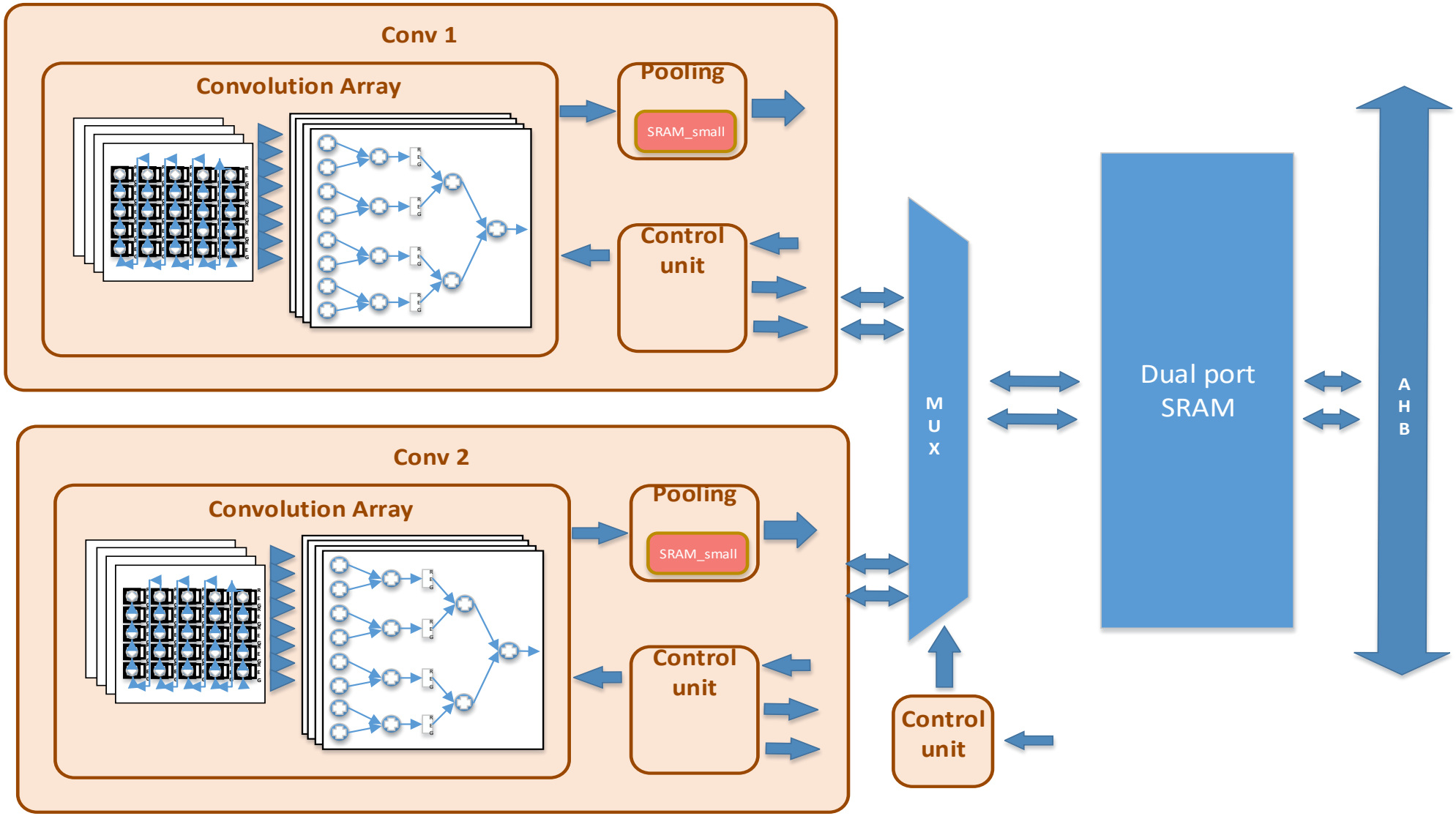
Architecture – Convolution Accelerator



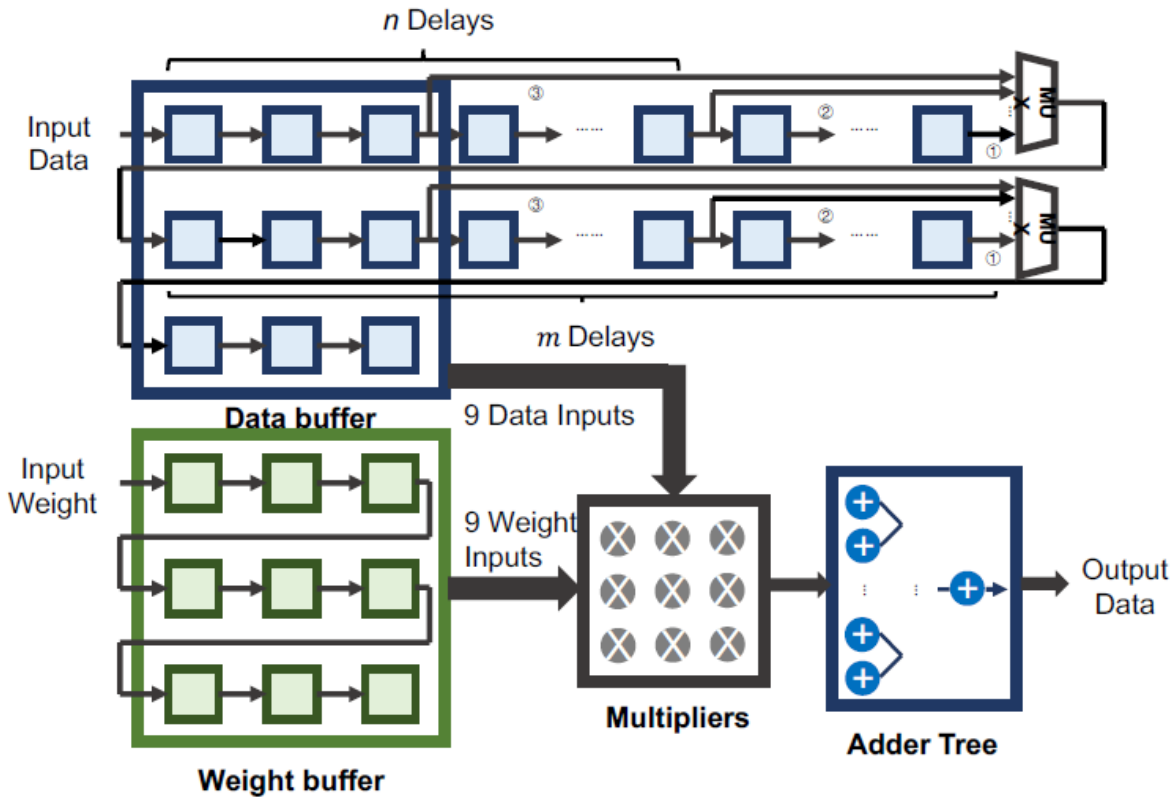
Architecture – Convolution Accelerator



Architecture – Convolution Accelerator



Architecture – Compare with others



ref: Jiantao Qiu, Jie Wang, Song Yao, Kaiyuan Guo, Boxun Li, Erjin Zhou, Jincheng Yu, Tianqi Tang, Ningyi Xu, Sen Song, Yu Wang, and Huazhong Yang. Going deeper with embedded fpga platform for convolutional neural network. In ACM International Symposium on FPGA, 2016

Architecture – Compare with others

- Line buffer
  - Faster
    1 data in 1 data out
  - More registers
    800*4(kernel size -1) + 5

- Ours
  - Less registers
    5*5
  - Take more time
    5 data in 1 data out

Architecture – Convolution Accelerator

- 2 layers convolution
  - Input : 800*600*1
  - Layer 1 output : 398*298*2 (16 bits)
  - Output : 197*147*4 (16 bits)

- Cycles estimation
  - with output 3rd dimension parallel, Parallelism = 2

    L1 : 800*600 (all input) + computing and pooling + 398*298*2 (all output) ≒ 1700000

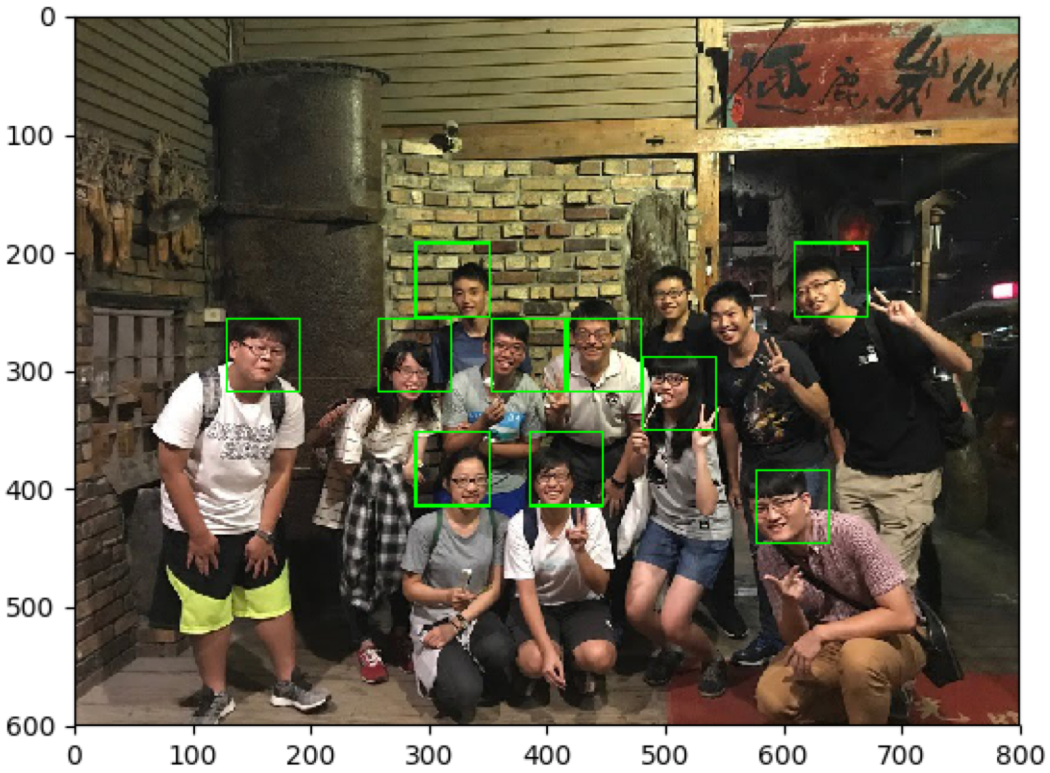
    L2 : 398*298*2(last input) + computing and pooling + 197*147*4 (all output) ≒ 1250000

    Overall : 1700000 + 1250000 ≒ 3000000(cycles)

    Overall with control signal : 3000000 ~ 3700000(cycles)

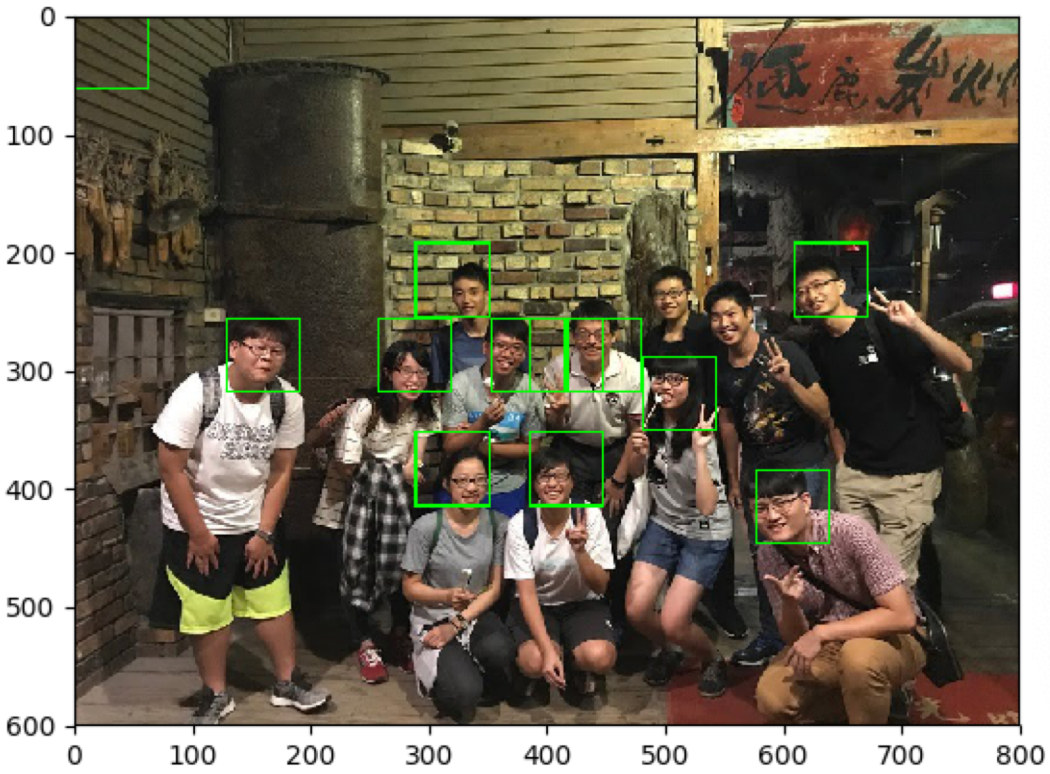
Architecture - Verification

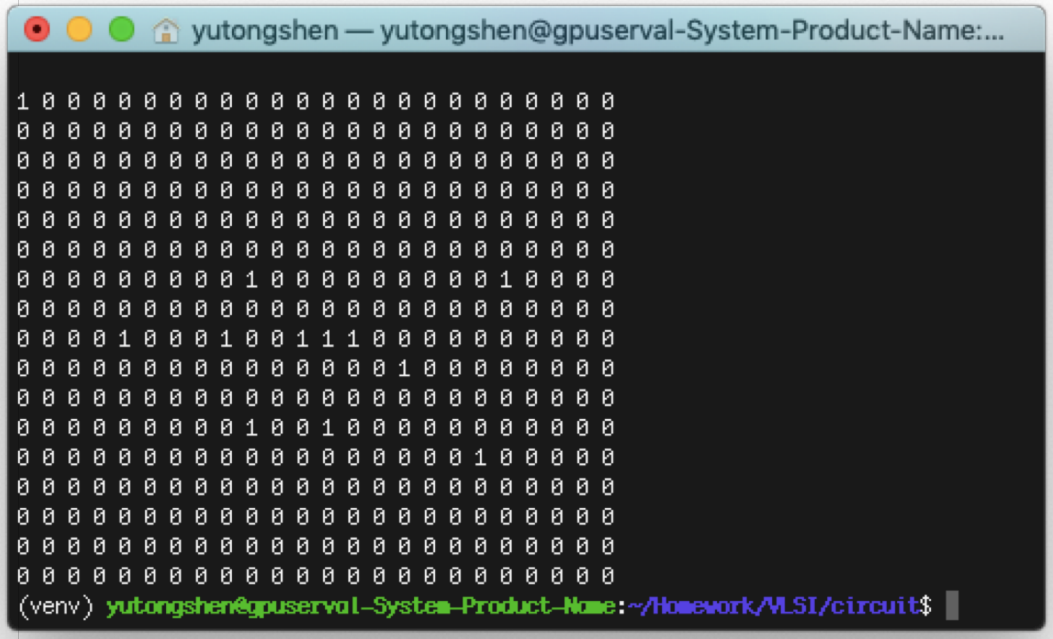
- Model training by Keras (floating point 32 bits)



Architecture - Verification

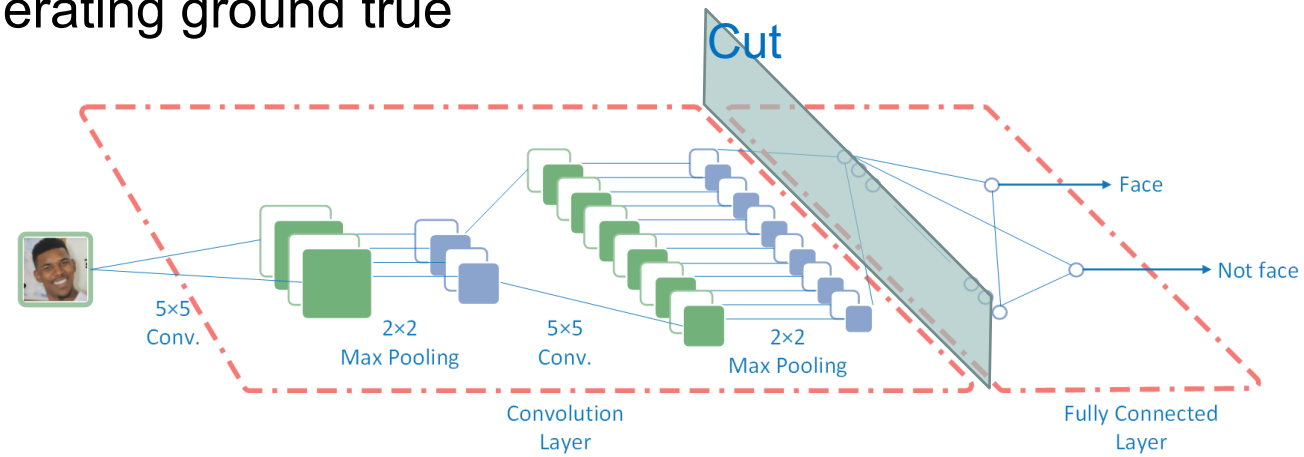
- Convert weights to 16 bits fixed point

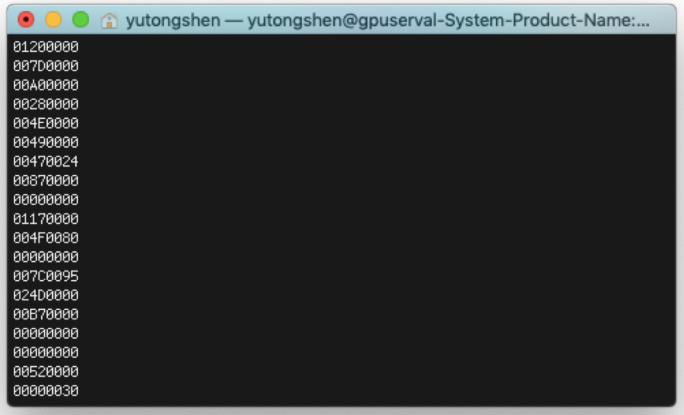




Architecture - Verification

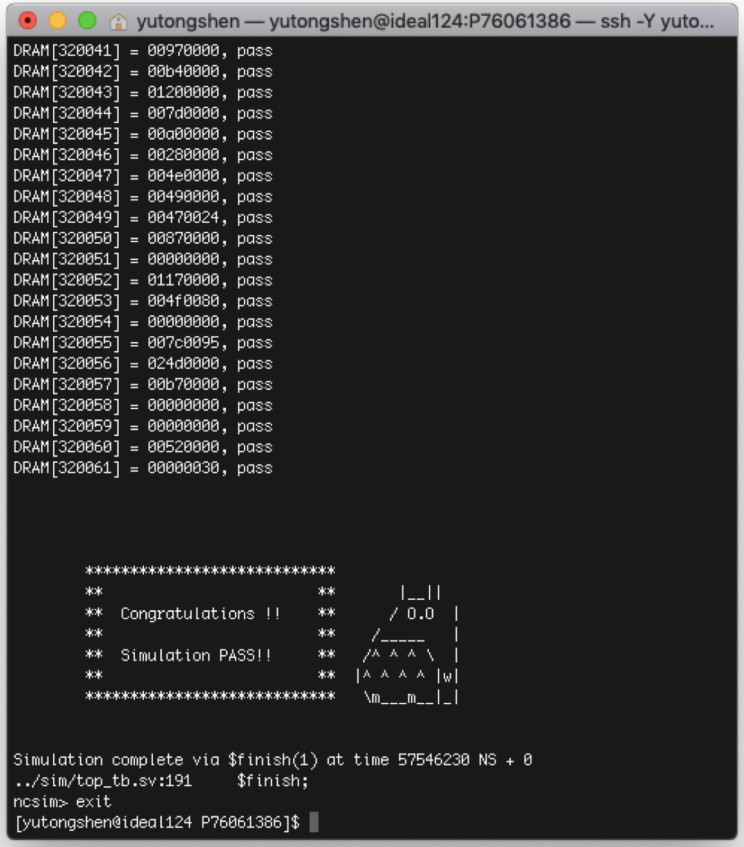
- Generating ground true





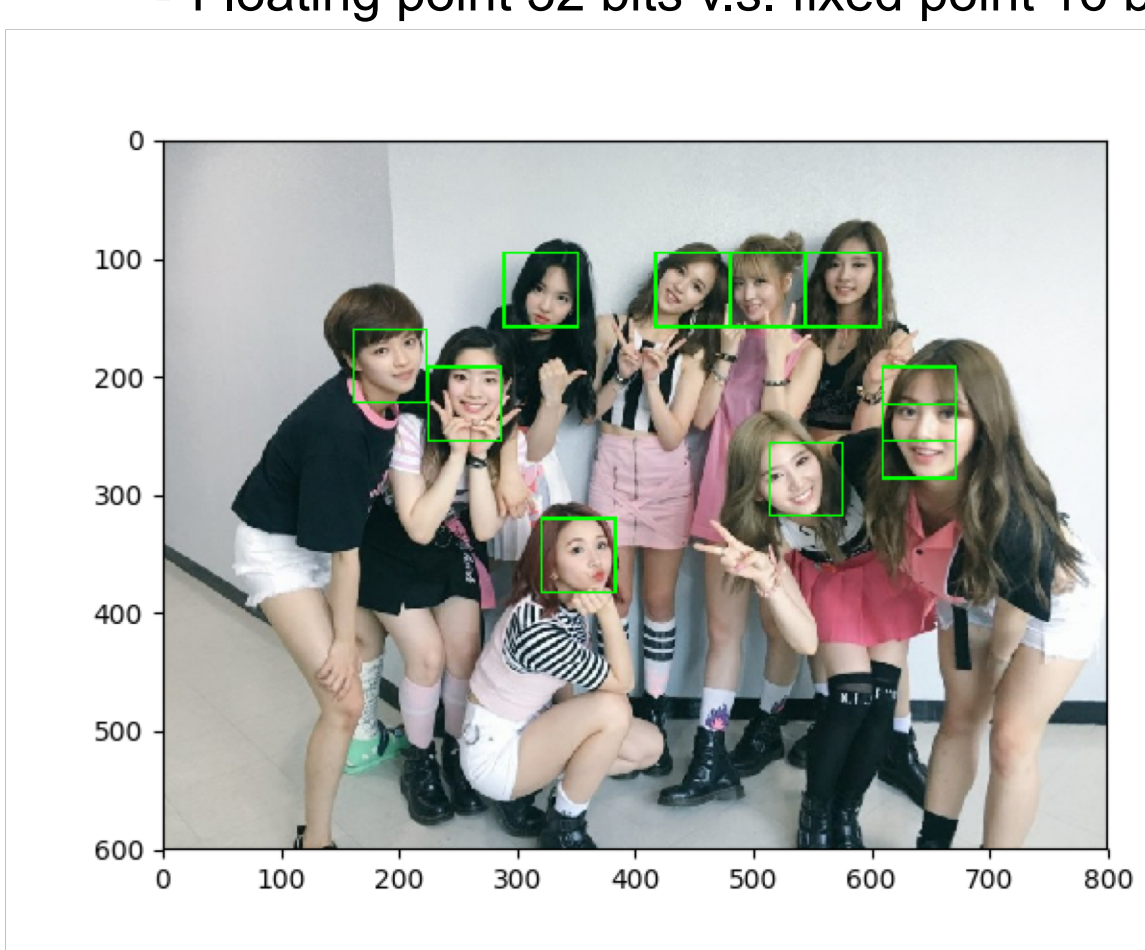
Architecture - Verification

- Result



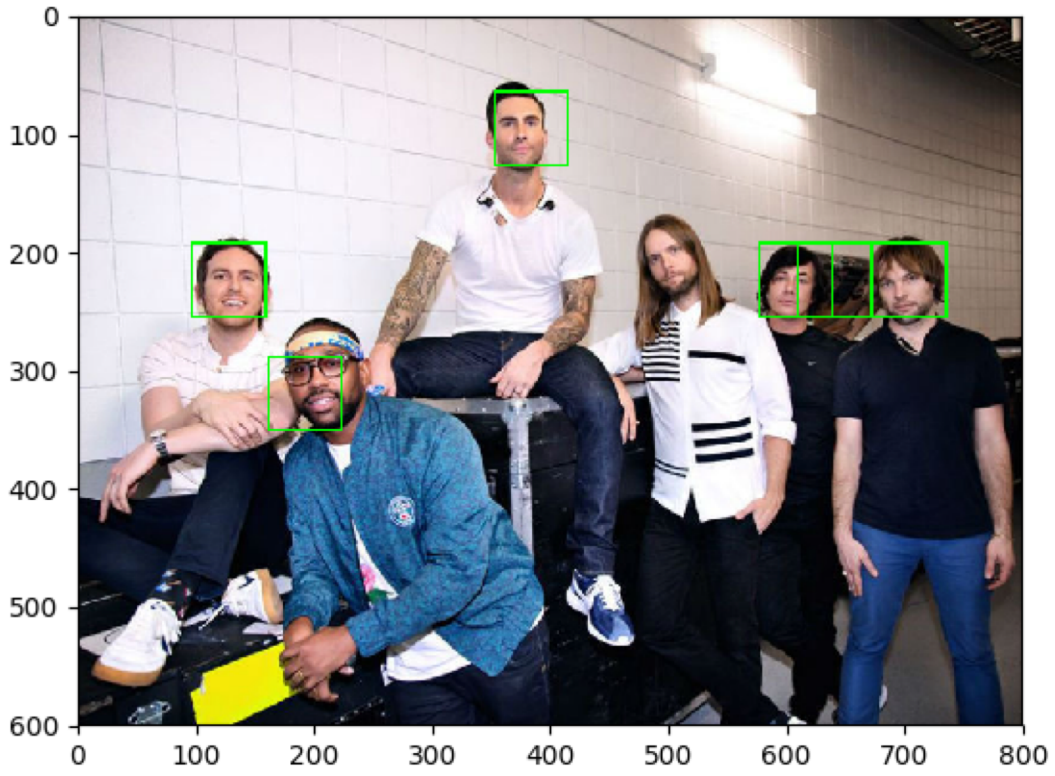
Architecture - Verification

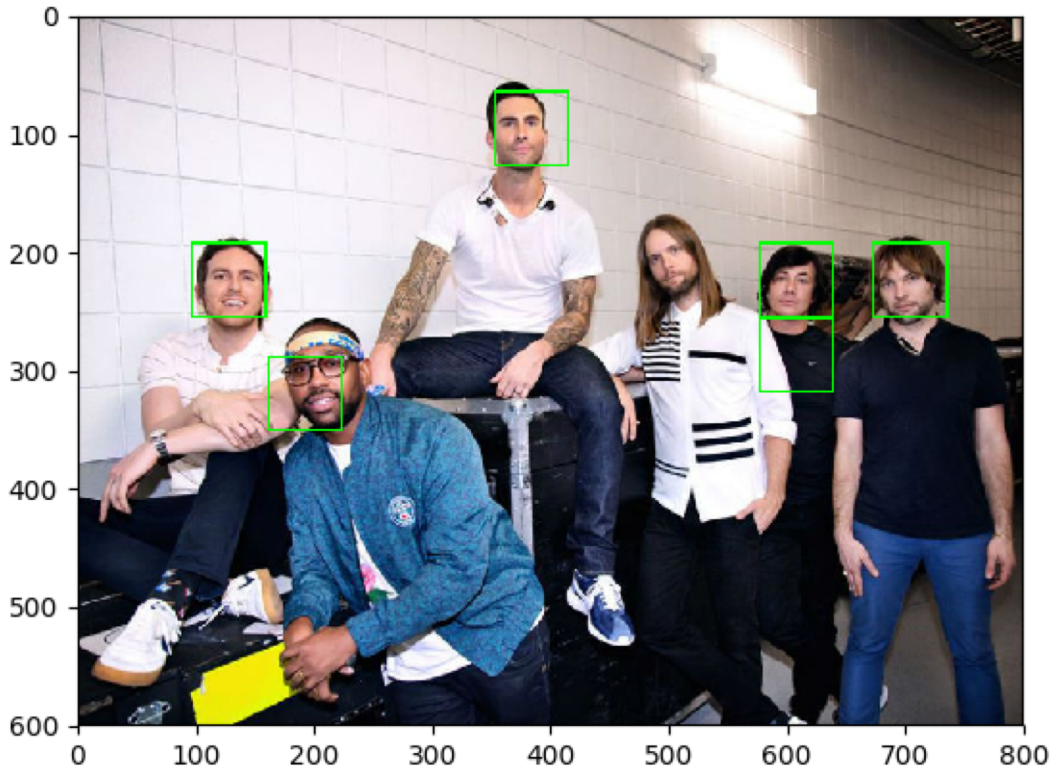
- Floating point 32 bits v.s. fixed point 16 bits



Architecture - Verification

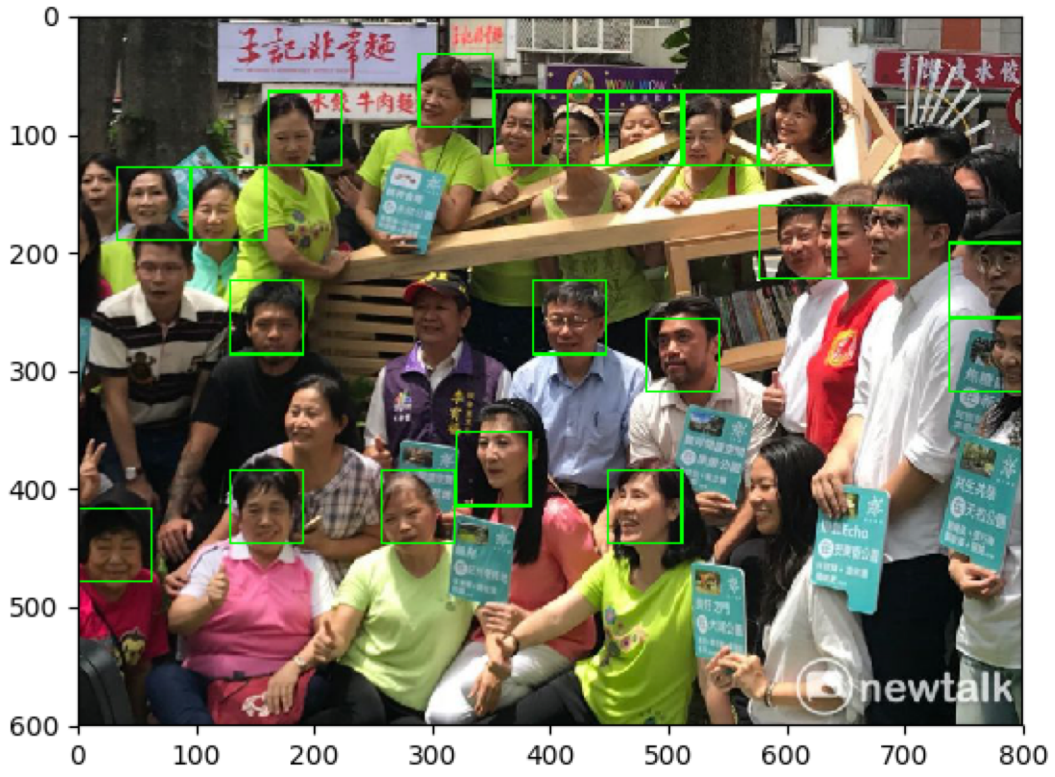
- Floating point 32 bits v.s. fixed point 16 bits

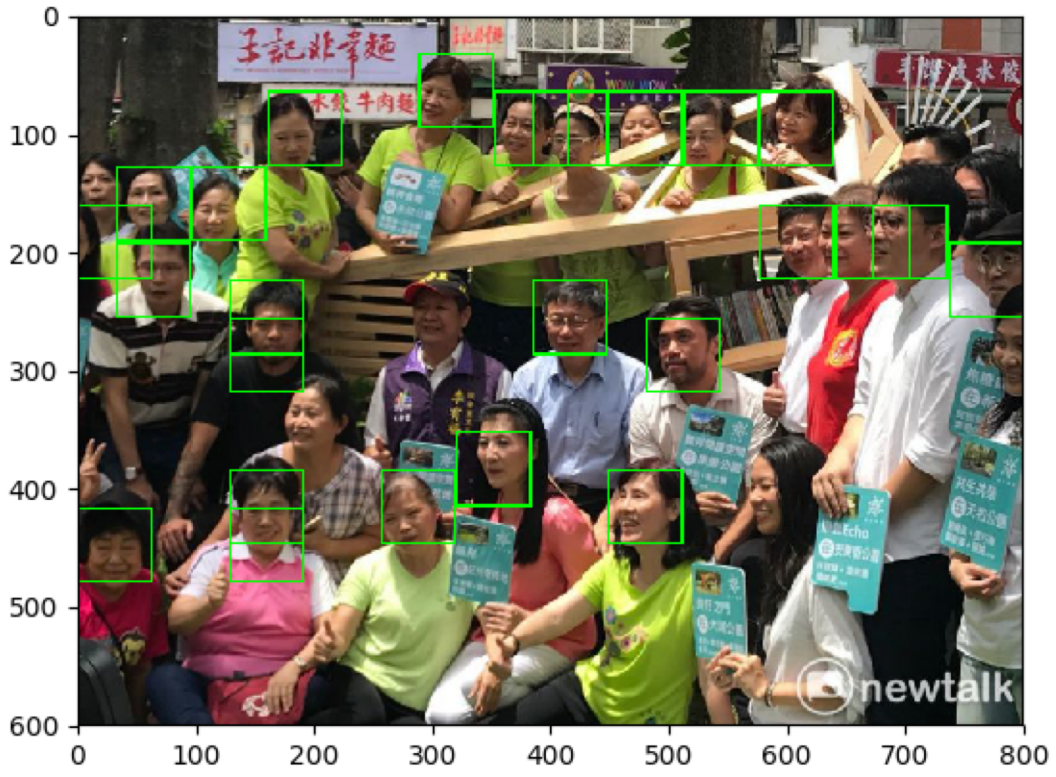




Architecture - Verification

- Floating point 32 bits v.s. fixed point 16 bits





Outline

- Introduction
- Method
- Architecture
- Timing Analysis

Timing Analysis

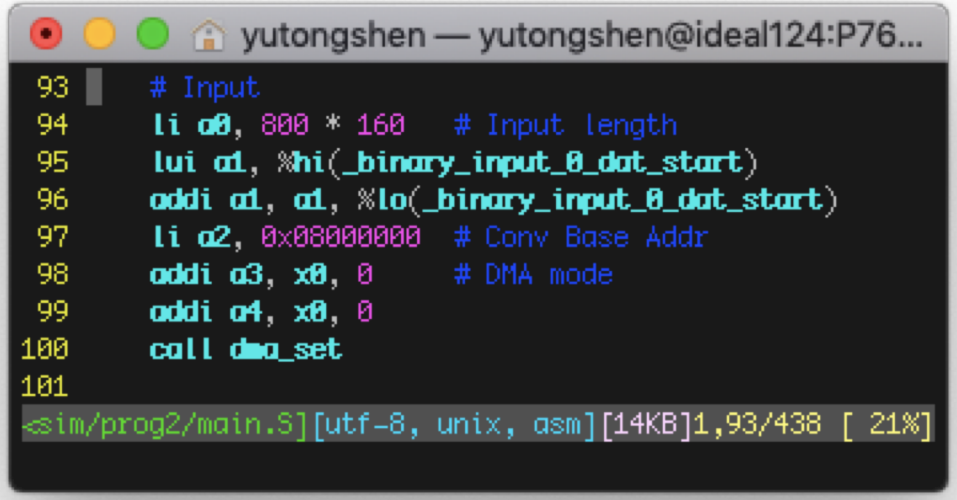
- Cutting 4 segments

Timing Analysis

- Cutting 4 segments

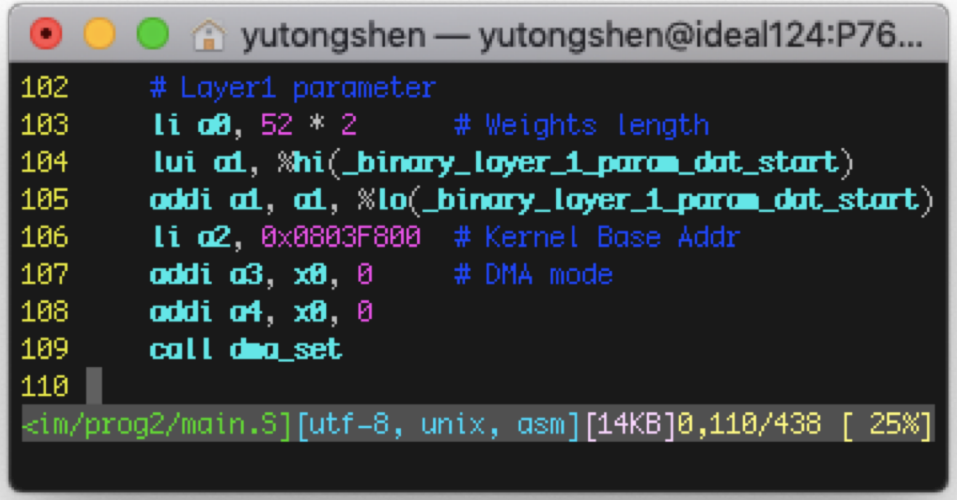
Timing Analysis

- Step 1. Move the segment of input from external DRAM to ASPU memory



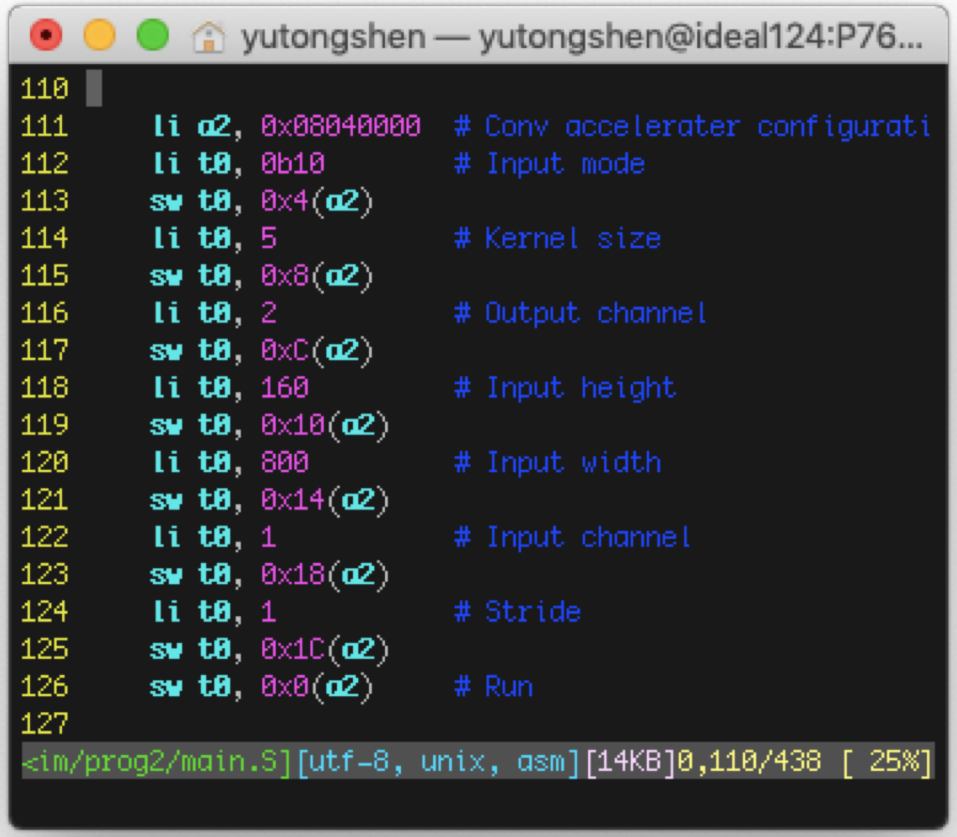
Timing Analysis

- Step 2. Move convolutional weights (layer1) from external DRAM to ASPU memory



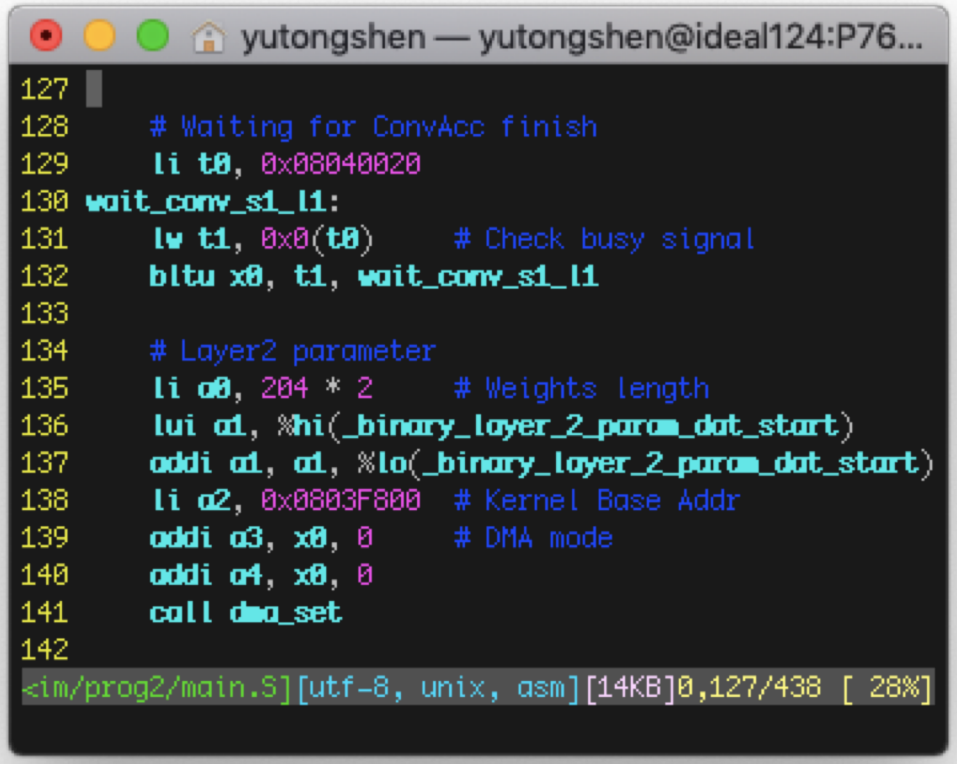
Timing Analysis

- Step 3. Setting convolutional accelerator



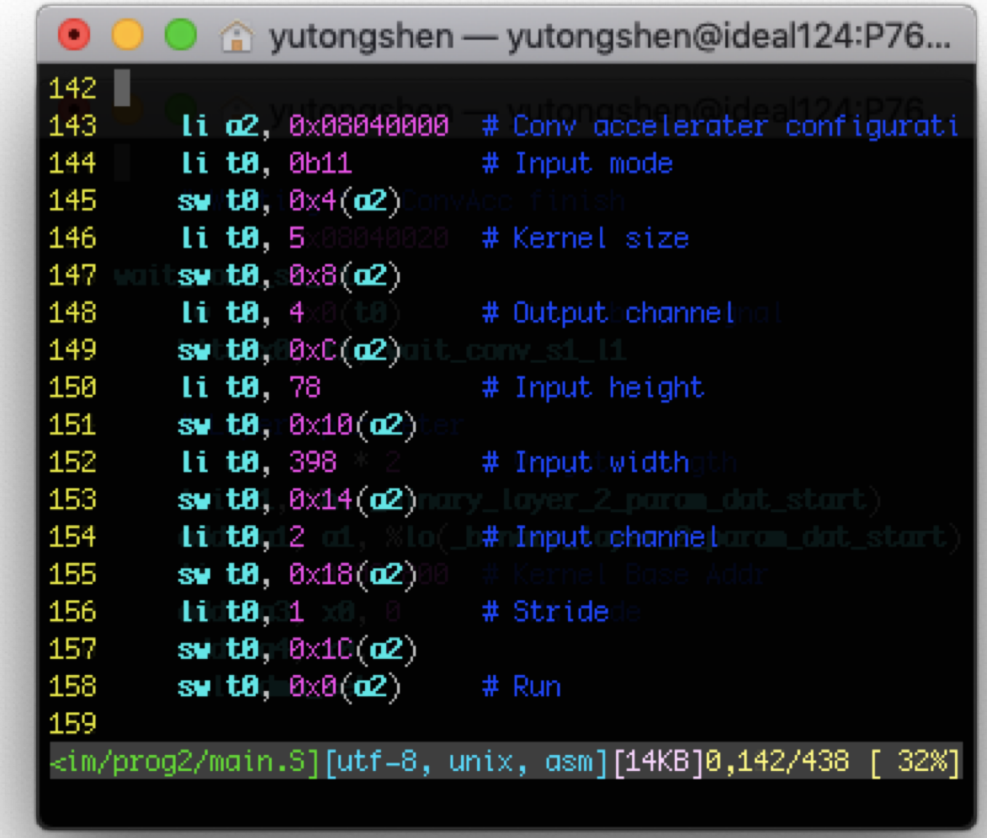
Timing Analysis

- Step 4. Waiting for Convolutional accelerator finish, then move convolutional weights (layer2) from external DRAM to ASPU memory



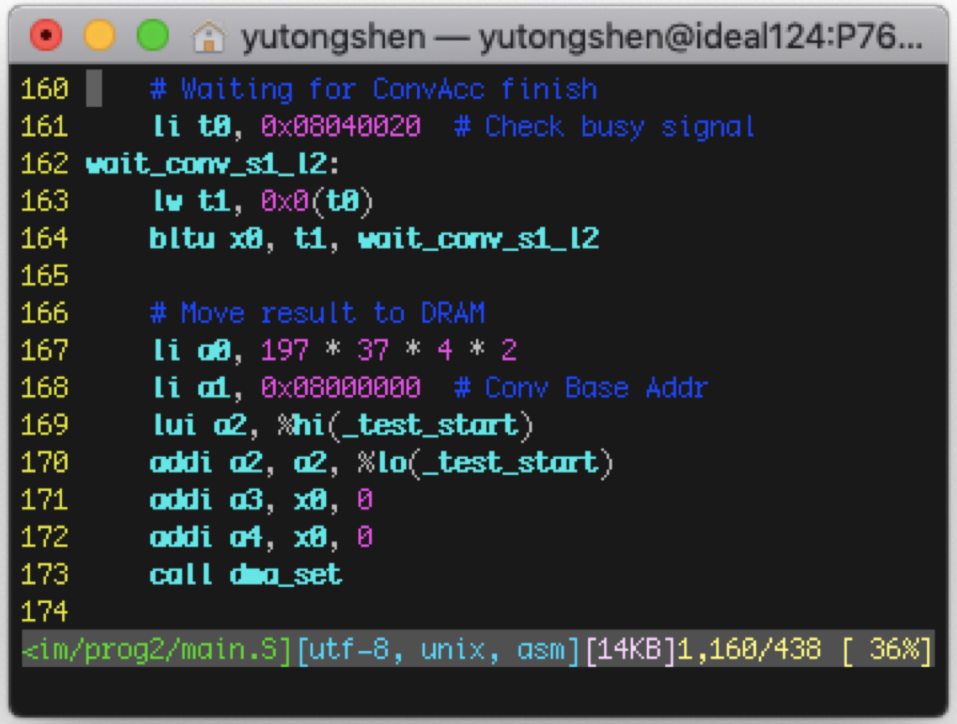
Timing Analysis

- Step 5. Setting convolutional accelerator



Timing Analysis

- Step 6. Moving result to DRAM



Timing Analysis

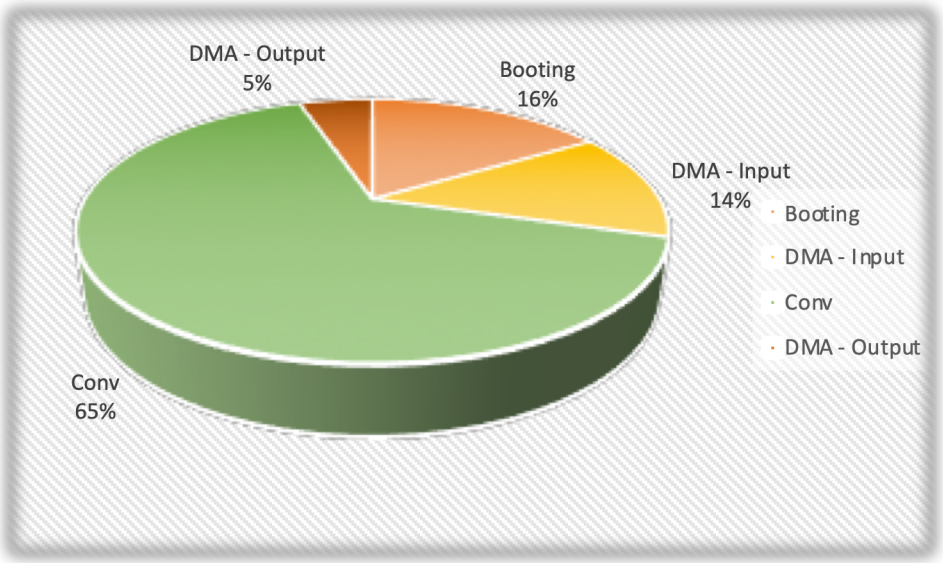
Cycle	937,000	1,130,000	2,075,000	2,148,000
AHB				
CPU	Bootloader			
DMA		Move image to Conv		Move result to DRAM
Conv			Calculate segment1	

Cycle	23,400,000	3,286,000	3,359,000
AHB			
CPU			
DMA	Move image to Conv		Move result to DRAM
Conv		Calculate segment2	

Cycle	3,551,000	4,496,000	4,570,000
AHB			
CPU			
DMA	Move image to Conv		Move result to DRAM
Conv		Calculate segment3	

Cycle	4,761,000	5,683,000	5,754,000
AHB			
CPU			
DMA	Move image to Conv		Move result to DRAM
Conv		Calculate segment4	

Timing Analysis



Job	Time (cycle)
Booting	937,000
DMA - Input	768,000
Conv	3,758,000
DMA - Output	291,000
Total cycle	5,754,000

Timming analysis	Time (ns)
Clock period	10
Total time	57,540,000

Thanks for listening