

# 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、溫偉程

# Outline

- Introduction
- Method
- Architecture
- Timing Analysis



# Outline

- Introduction
- Method
- Architecture
- Timing Analysis



# Introduction

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



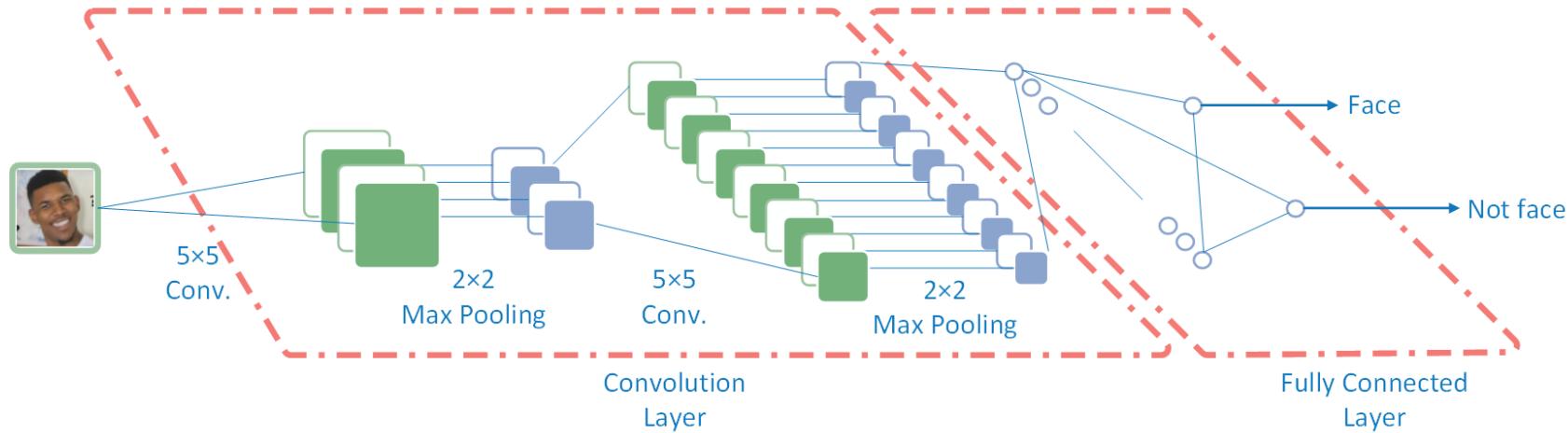
# Introduction

- How to detect faces ?



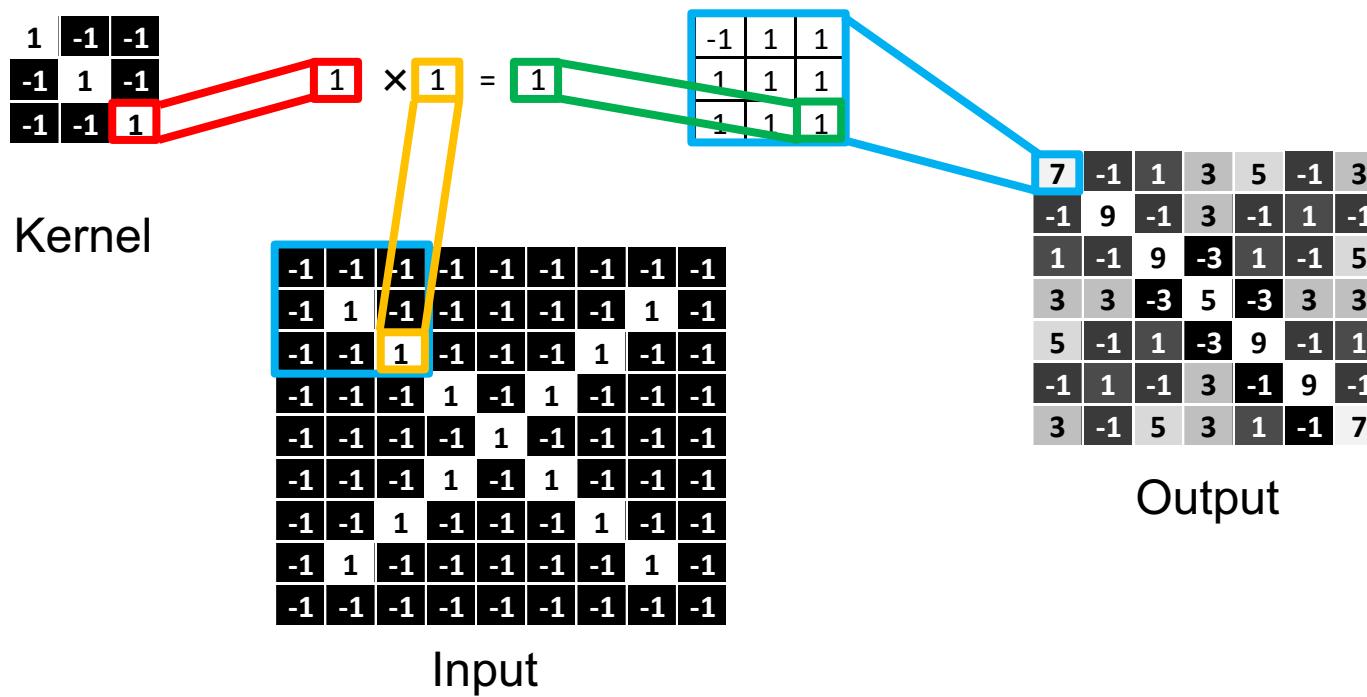
# Introduction

- How to detect faces ?



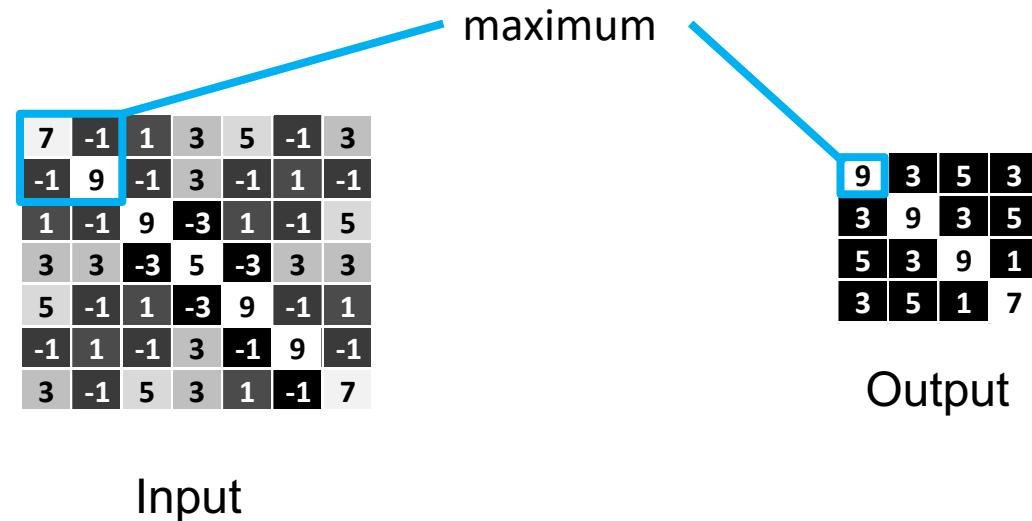
# Introduction

- Convolution



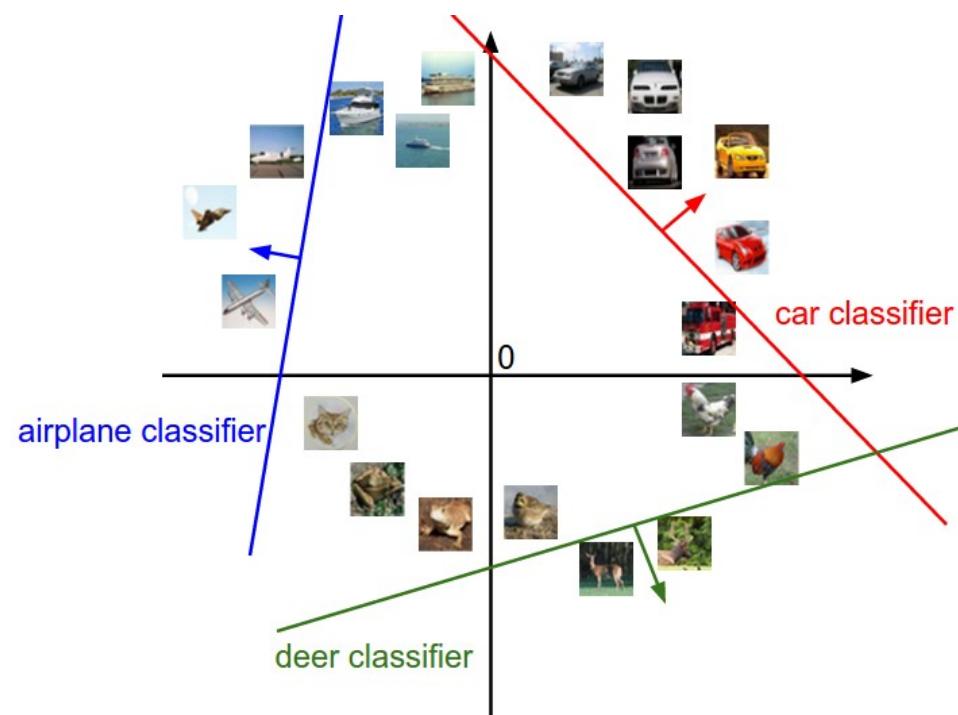
# Introduction

- Pooling



# Introduction

- Classification



# Outline

- Introduction
- Method
- Architecture
- Timing Analysis



# Method

- Data Set : LFW

## Labeled Faces in the Wild



**Menu**

- LFW Home
  - Mailing
  - Explore
  - Download
  - Train/Test

**Labeled Faces in the Wild Home**





# Method

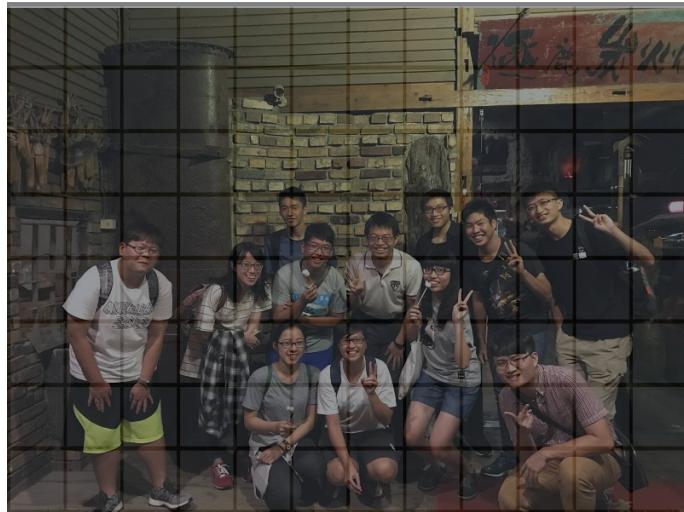
- Using Keras to train module

```
Layer (type)          Output Shape         Param # 
=====            ======           =====
conv2d_1 (Conv2D)    (None, 60, 60, 2)      52
max_pooling2d_1 (MaxPooling2D) (None, 30, 30, 2) 0
conv2d_2 (Conv2D)    (None, 26, 26, 4)      204
max_pooling2d_2 (MaxPooling2D) (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
```

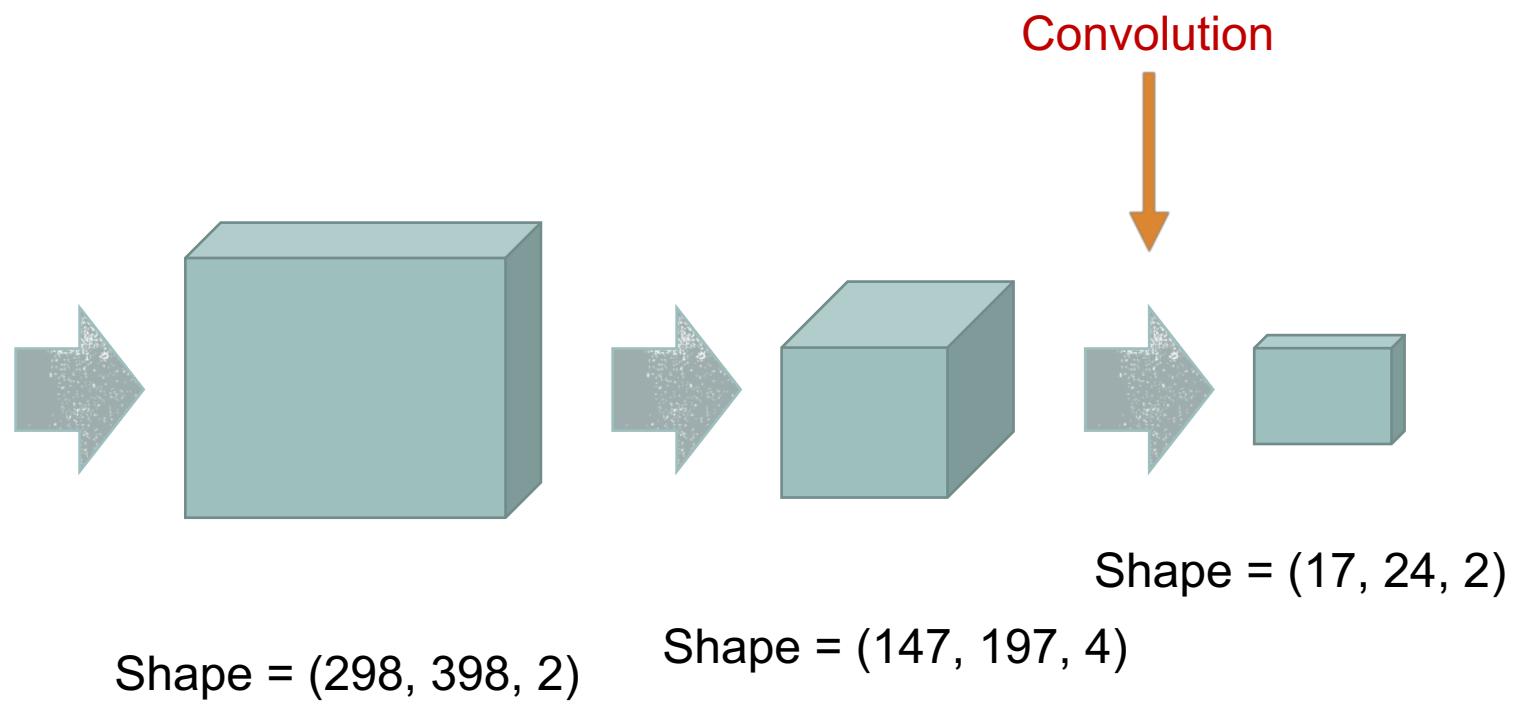
```
21172/21172 [=====] - 2s 105us/step - loss: 0.3900 - acc: 0.8169 - val_loss: 0.4706 - val_acc: 0.7864
Epoch 2/10
21172/21172 [=====] - 1s 45us/step - loss: 0.1669 - acc: 0.9410 - val_loss: 0.2745 - val_acc: 0.8929
Epoch 3/10
21172/21172 [=====] - 1s 51us/step - loss: 0.1080 - acc: 0.9633 - val_loss: 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)

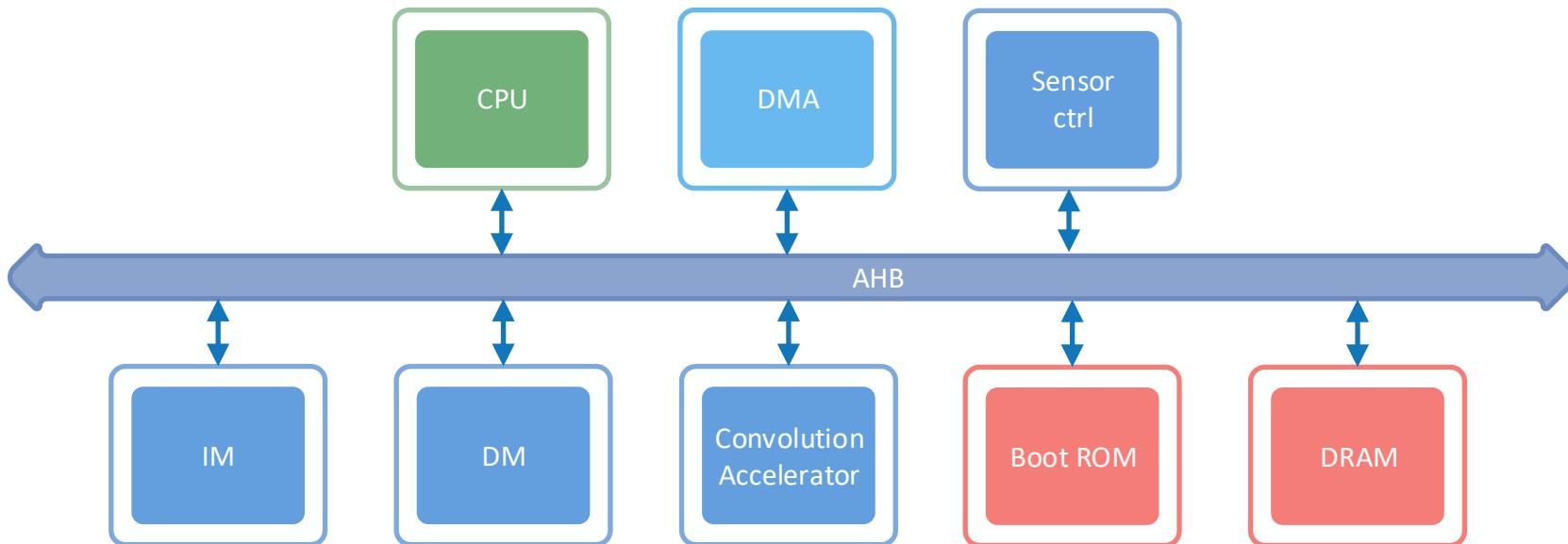


# Outline

- Introduction
- Method
- Architecture
- Timing Analysis



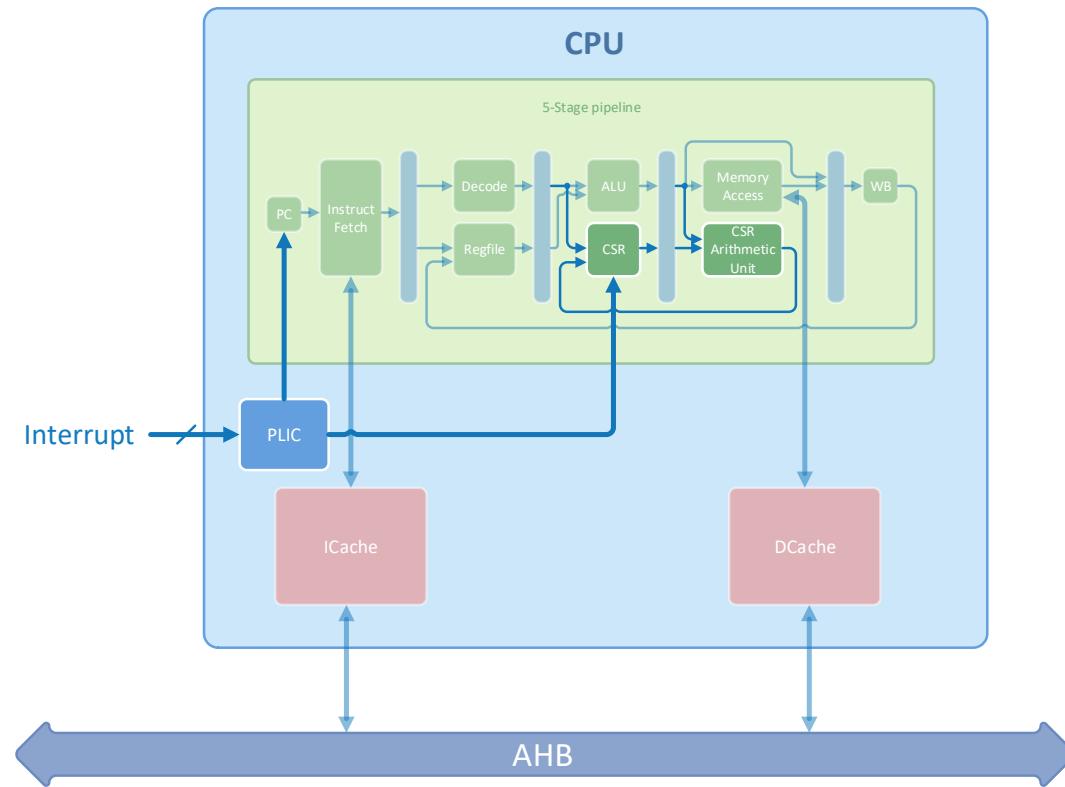
# Architecture



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)

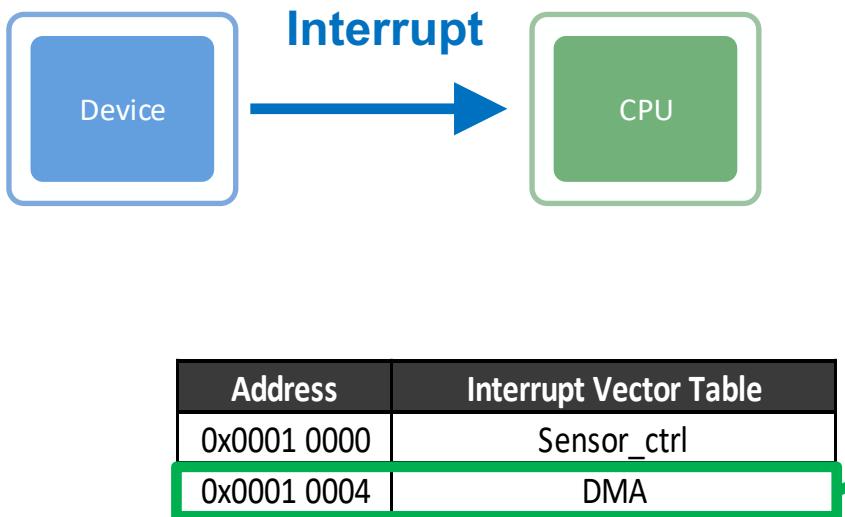


# Architecture - CPU



# Architecture - CPU

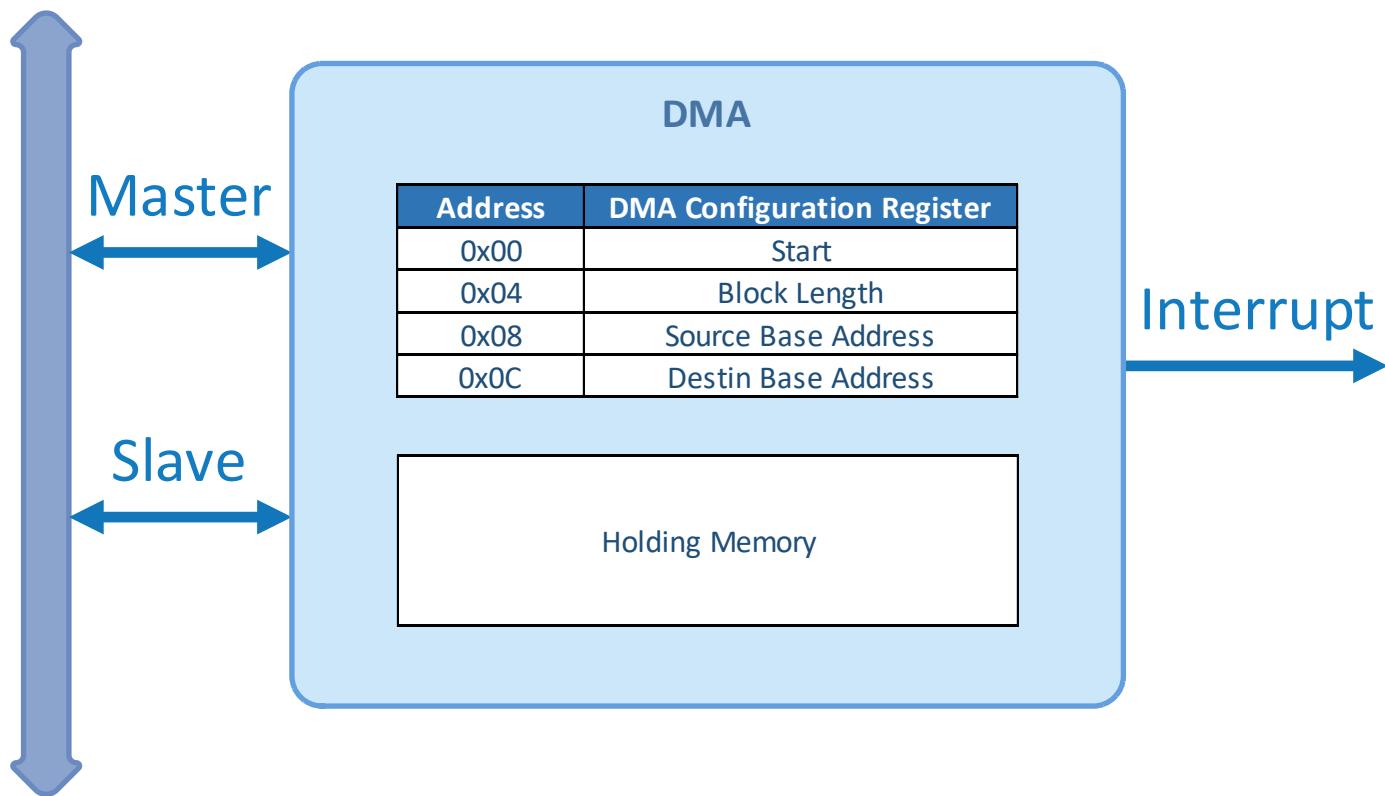
- Interrupt



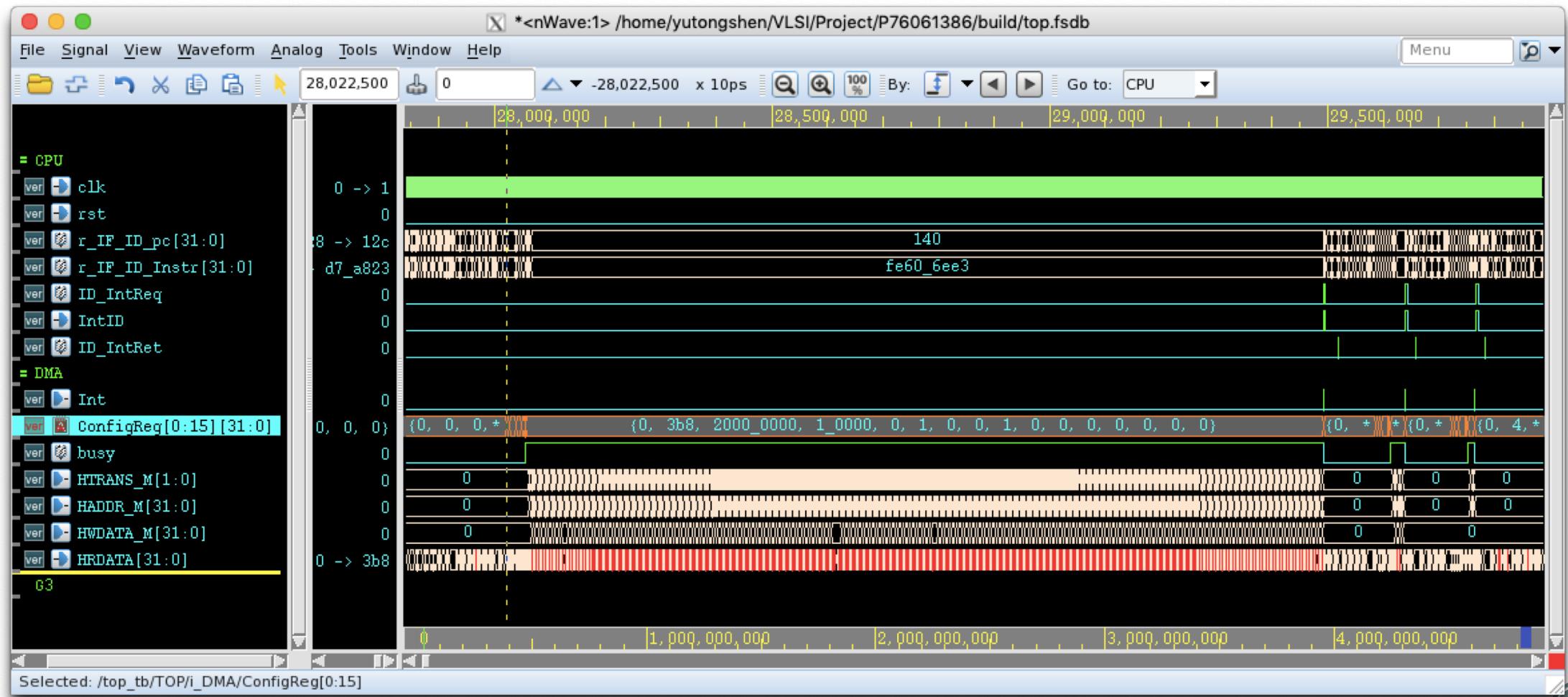
```
# Define constants
.section .text
.align 2
.globl trap_entry
trap_entry:
    j sensor_ctrl
    j dma_isr
sensor_ctrl:
    addi sp, sp, -4
    sw x1, 0(sp) # save ra
    jal saovereg
    jal copy
    jal loadreg
    lw x1, 0(sp) # load ra
    addi sp, sp, 4
    nret
dma_isr: total 3848
    addi sp, sp, -4 15 yuto
    sw x1, 0(sp) # save ra
    jal saovereg
    jal _dma_isr
    jal loadreg
    lw x1, 0(sp) # load ra
    addi sp, sp, 4 3 yuto
    nret
44000 -rw-r--r-- 1 yuto
```



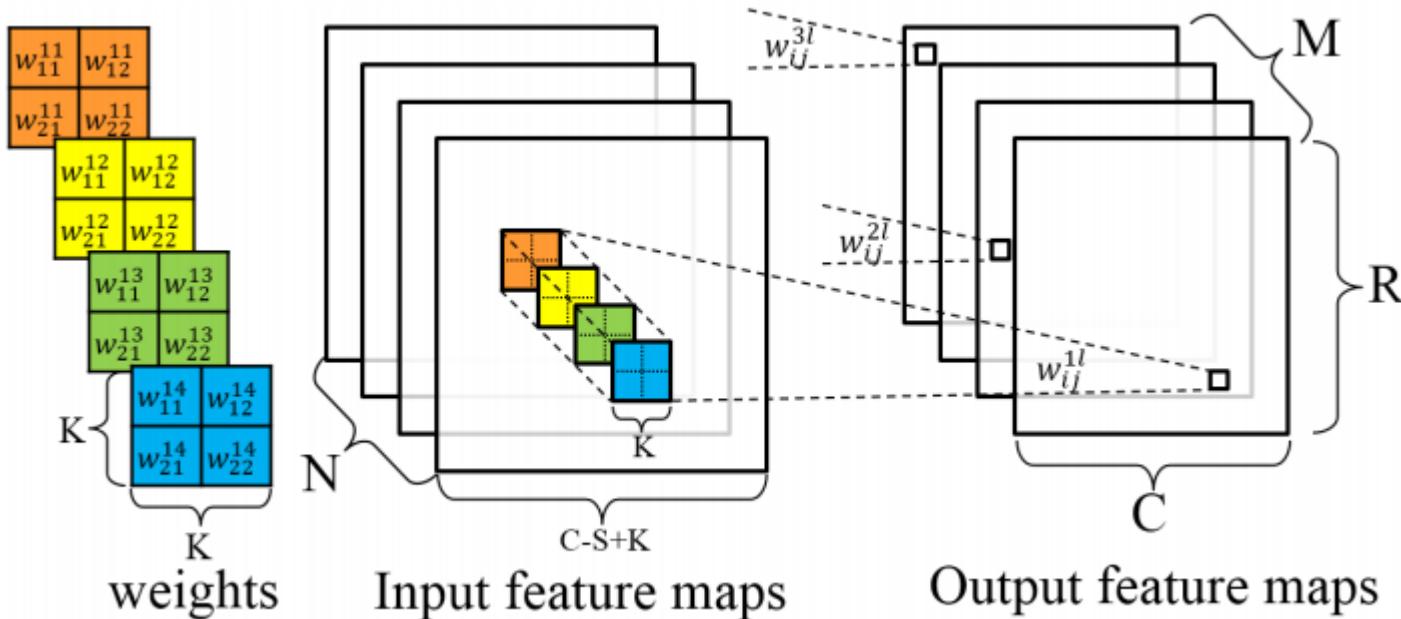
# Architecture - DMA



# Architecture - DMA



# Architecture – Convolution Accelerator

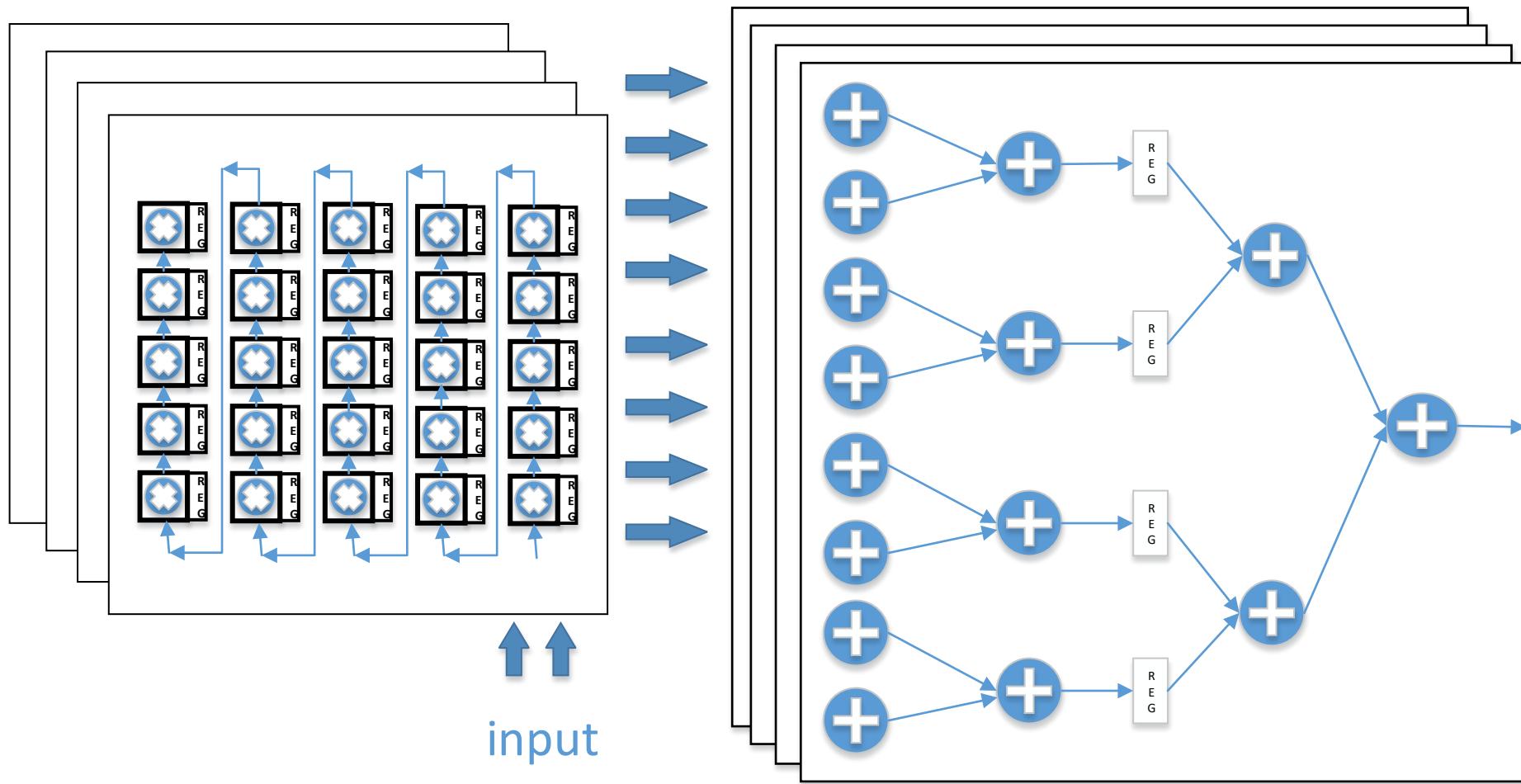


- Convolution computing

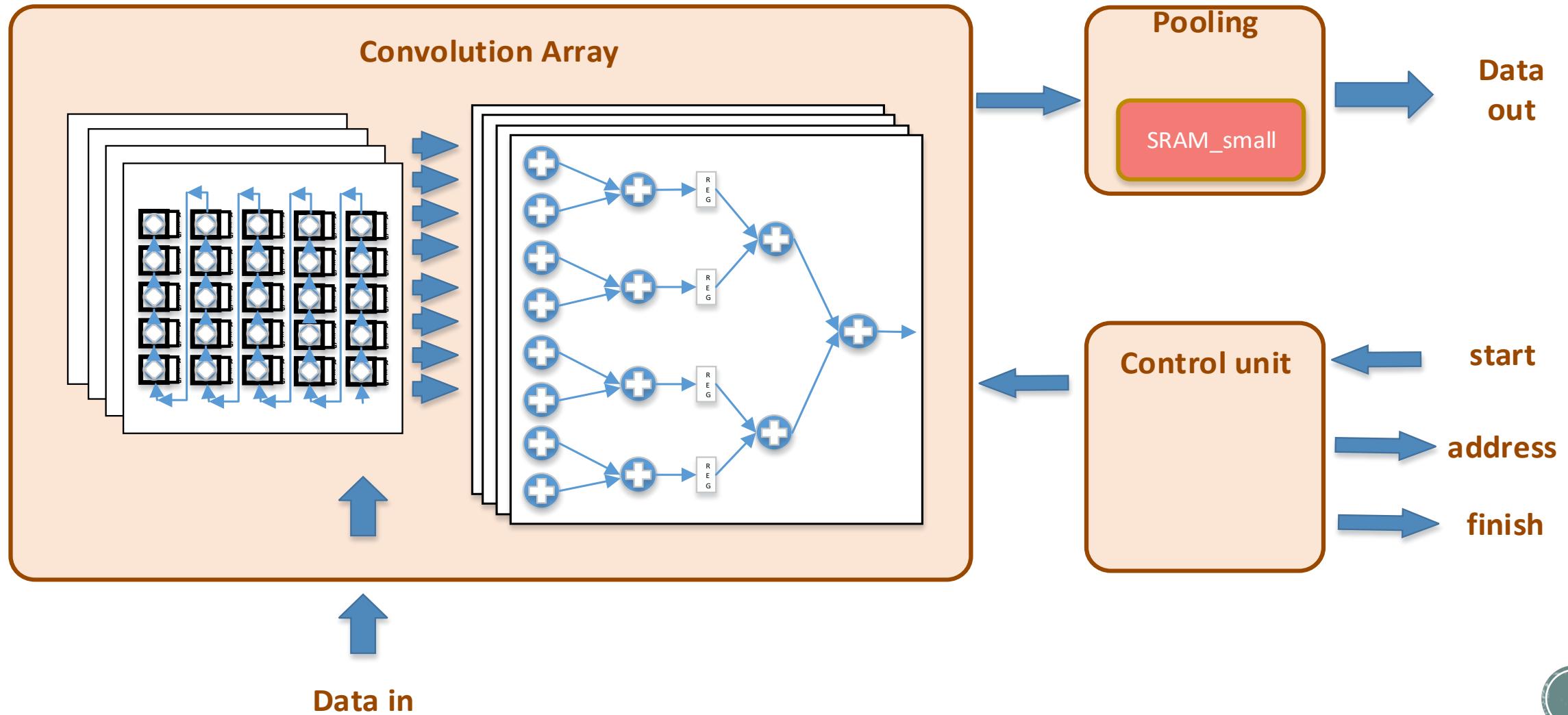
ref: C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao, and J. Cong, “Optimizing fpga-based accelerator design for deep convolutional neural networks,” in FPGA, 2015.



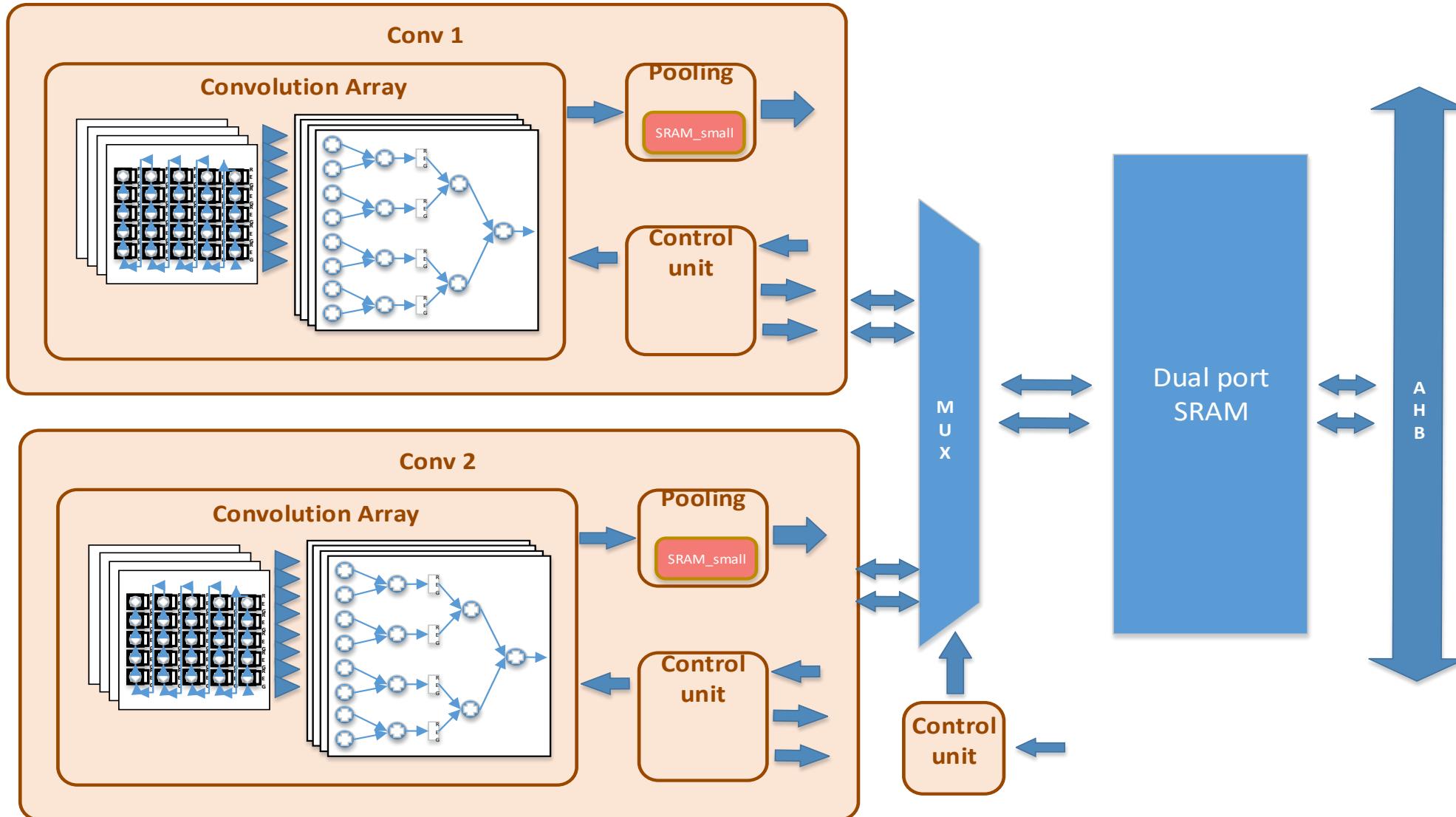
# Architecture – Convolution Accelerator



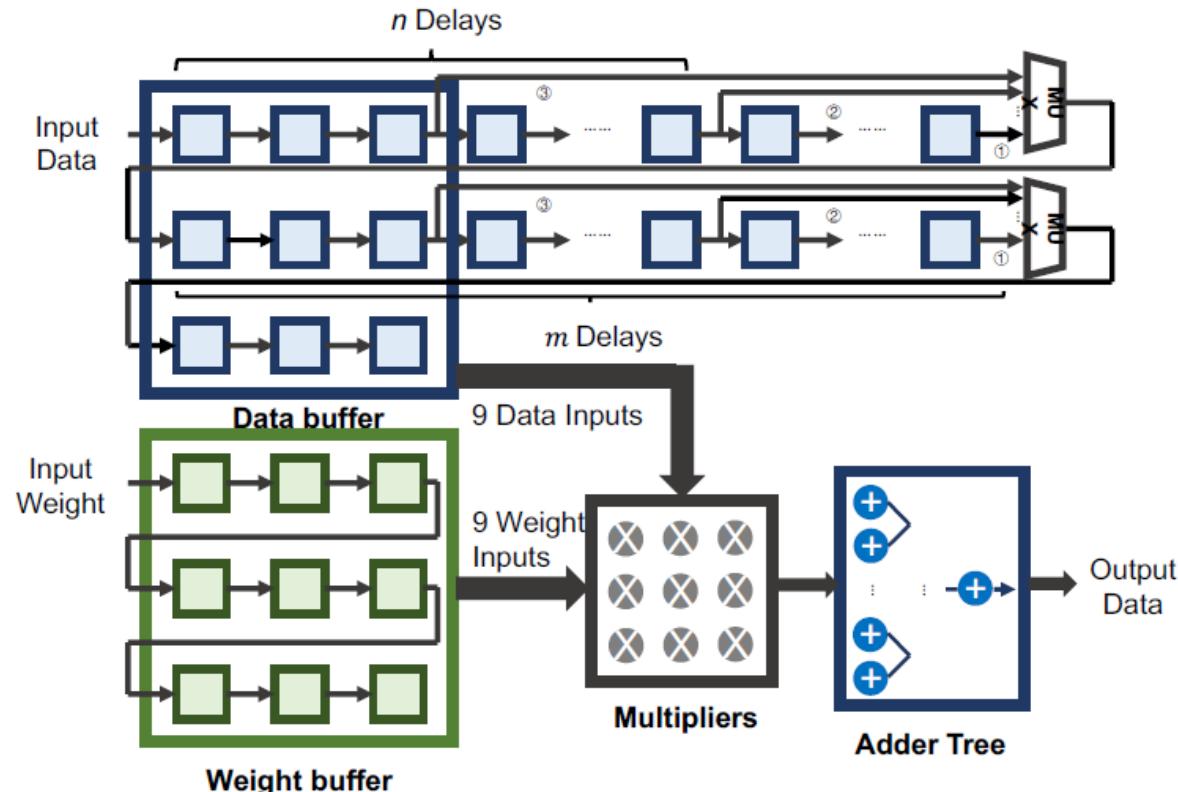
# Architecture – Convolution Accelerator



# 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
    - $800 * 4(\text{kernel size} - 1) + 5$
- Ours
  - Less registers
    - $5 * 5$
  - Take more time
    - 5 data in 1 data out



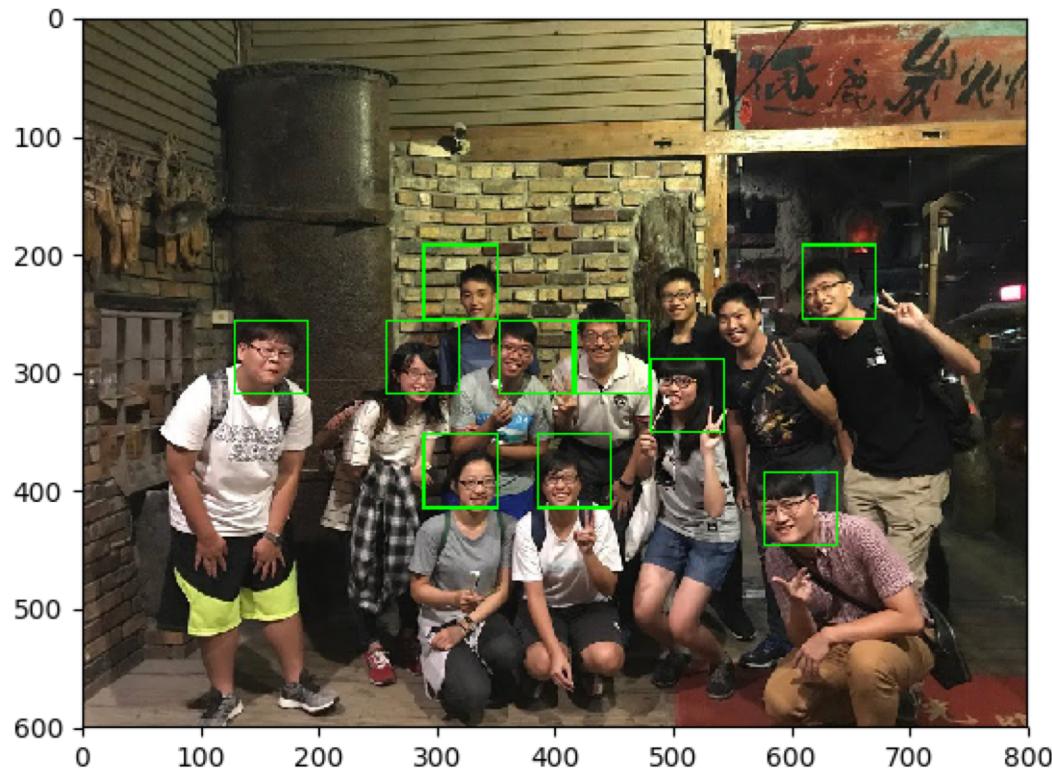
# Architecture – Convolution Accelerator

- 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 3<sup>rd</sup> 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)



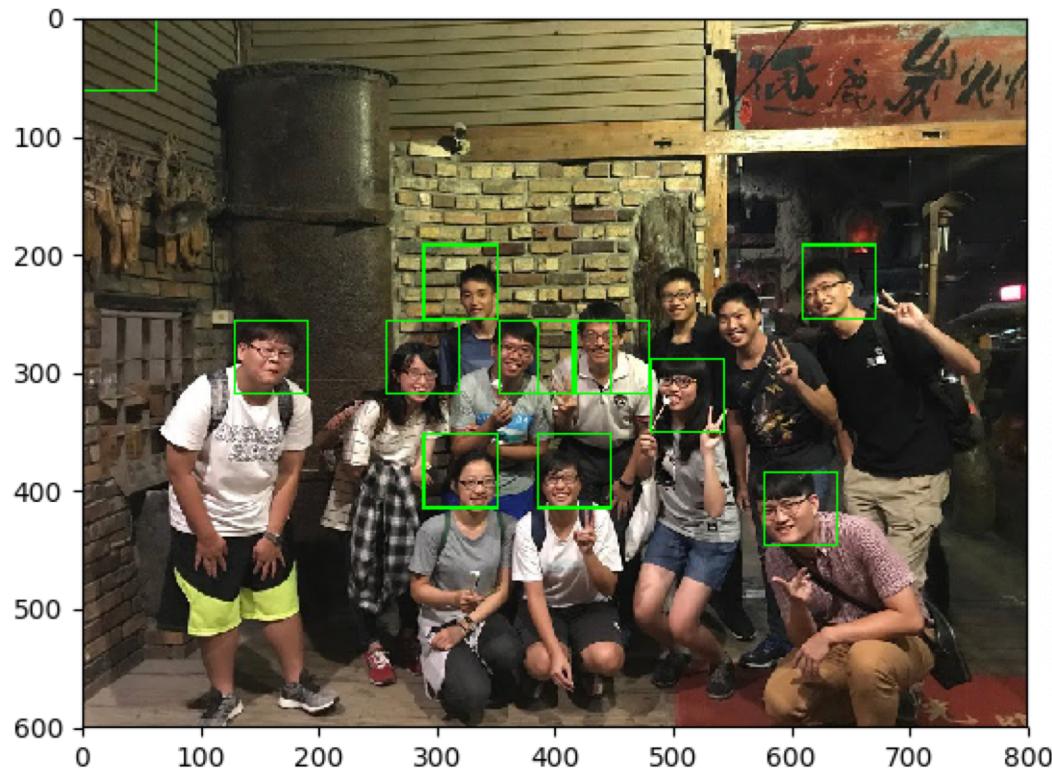
# Architecture - Verification

- Model training by Keras (floating point 32 bits)



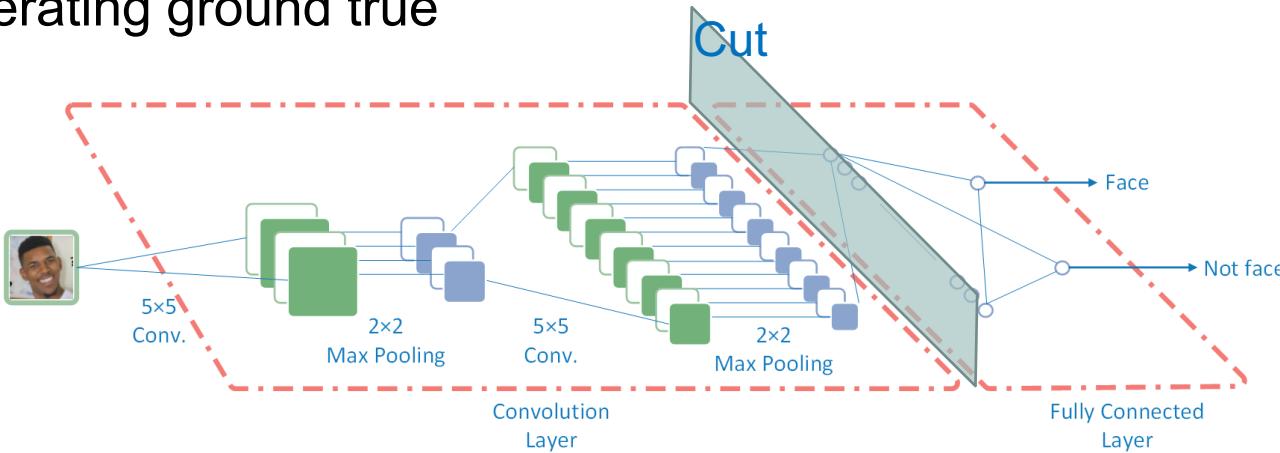
# Architecture - Verification

- Convert weights to 16 bits fixed point



# Architecture - Verification

- Generating ground true



```
yutongshen — yutongshen@gpuserval-System-Product-Name:...
```

01200000
00700000
00400000
00280000
004E0000
00490000
00470024
00870000
00000000
01170000
004F0080
00000000
007C0095
02400000
00870000
00000000
00000000
00520000
00000030



# Architecture - Verification

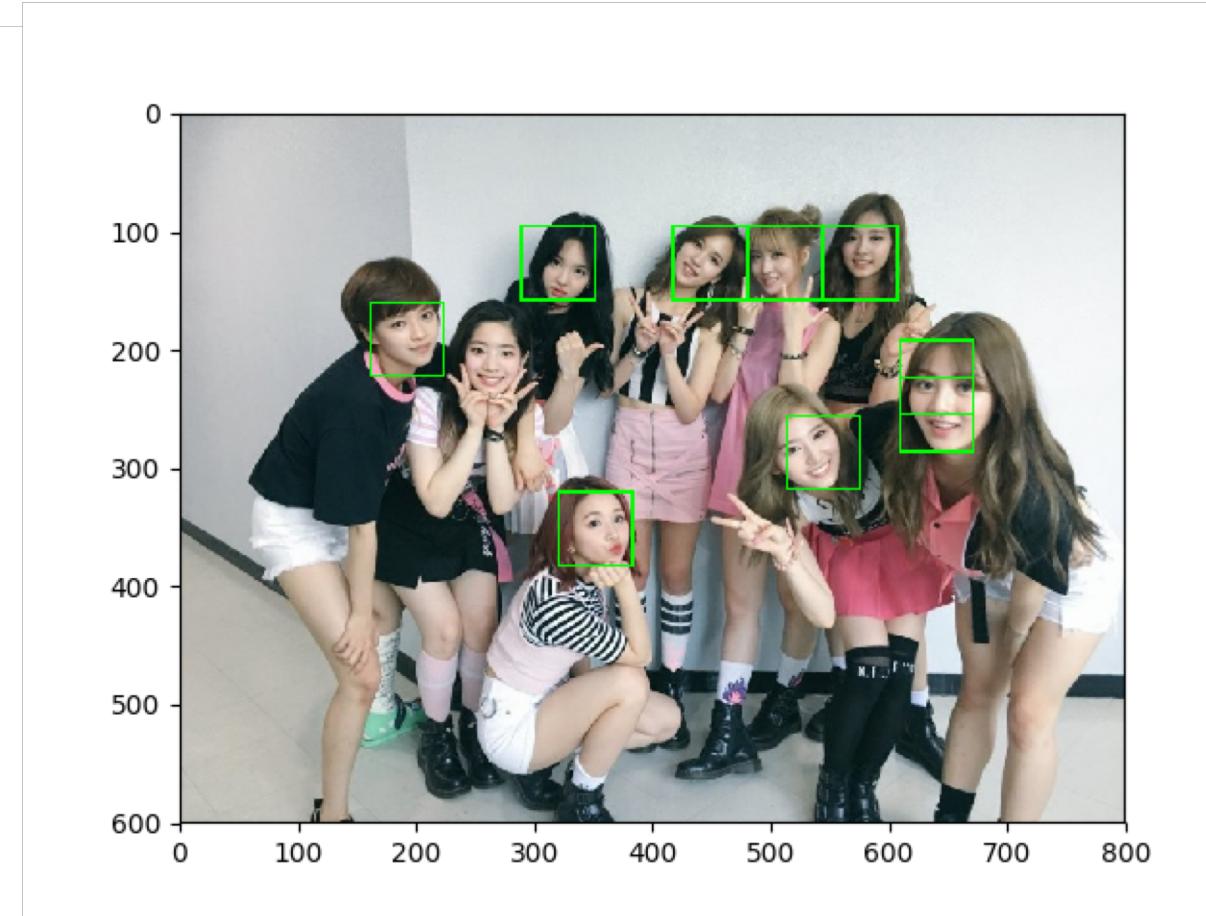
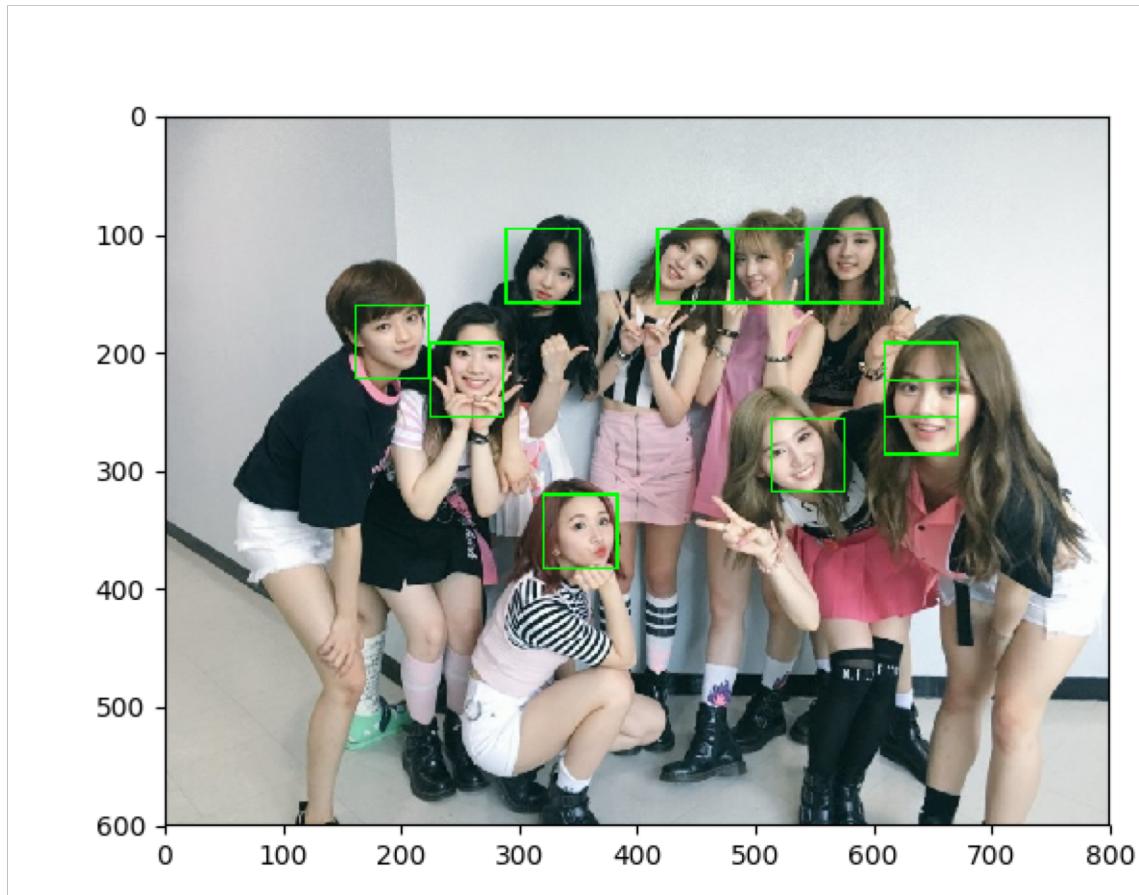
- Result

```
yutongshen — yutongshen@ideal124:P76061386 — ssh -Y yuto...
DRAM[320041] = 00970000, pass
DRAM[320042] = 00b40000, pass
DRAM[320043] = 01200000, pass
DRAM[320044] = 007d0000, pass
DRAM[320045] = 00000000, pass
DRAM[320046] = 00280000, pass
DRAM[320047] = 004e0000, pass
DRAM[320048] = 00490000, pass
DRAM[320049] = 00470024, pass
DRAM[320050] = 00870000, pass
DRAM[320051] = 00000000, pass
DRAM[320052] = 01170000, pass
DRAM[320053] = 004f0080, pass
DRAM[320054] = 00000000, pass
DRAM[320055] = 007c0095, pass
DRAM[320056] = 024d0000, pass
DRAM[320057] = 00070000, pass
DRAM[320058] = 00000000, pass
DRAM[320059] = 00000000, pass
DRAM[320060] = 00520000, pass
DRAM[320061] = 00000030, pass

*****
**          **      |__| |
** Congratulations !!    **    / 0.0  |
**          **    /-----|
** Simulation PASS!!    **   /A ^ A \ |
**          **   |A ^ A ^ |w|
*****      **  \M____m__|_|
Simulation complete via $finish(1) at time 57546230 NS + 0
./sim/top_tb.sv:191      $finish;
ncsim> exit
[yutongshen@ideal124 P76061386]$
```

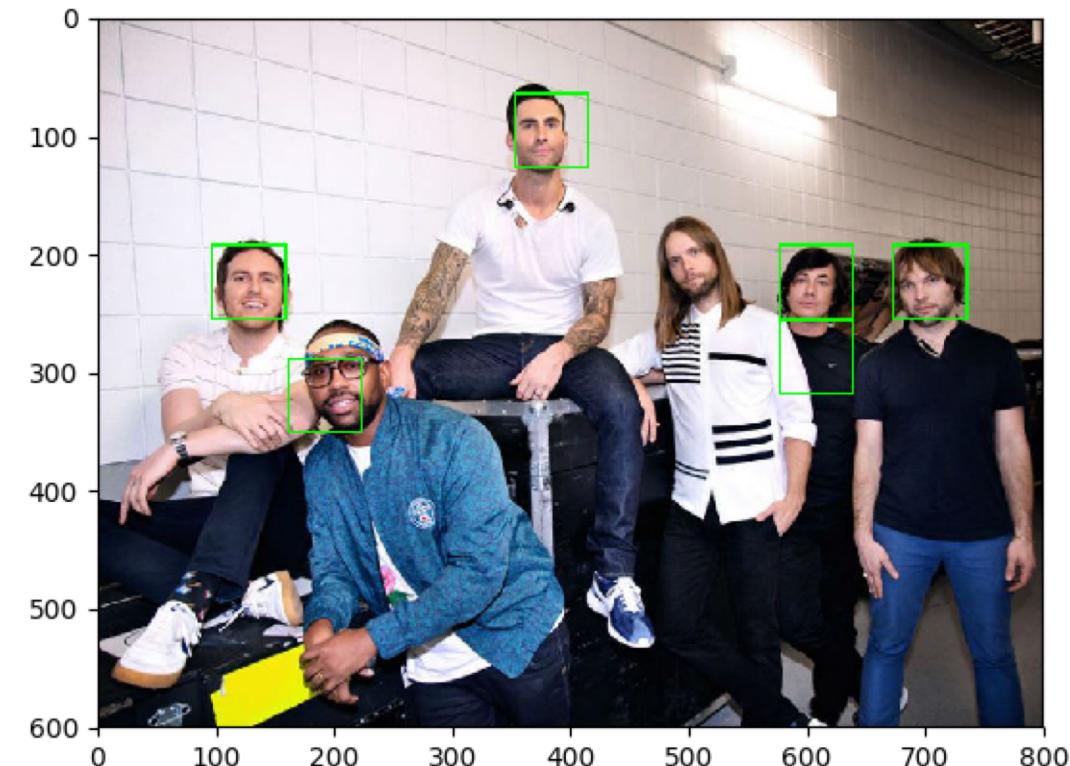
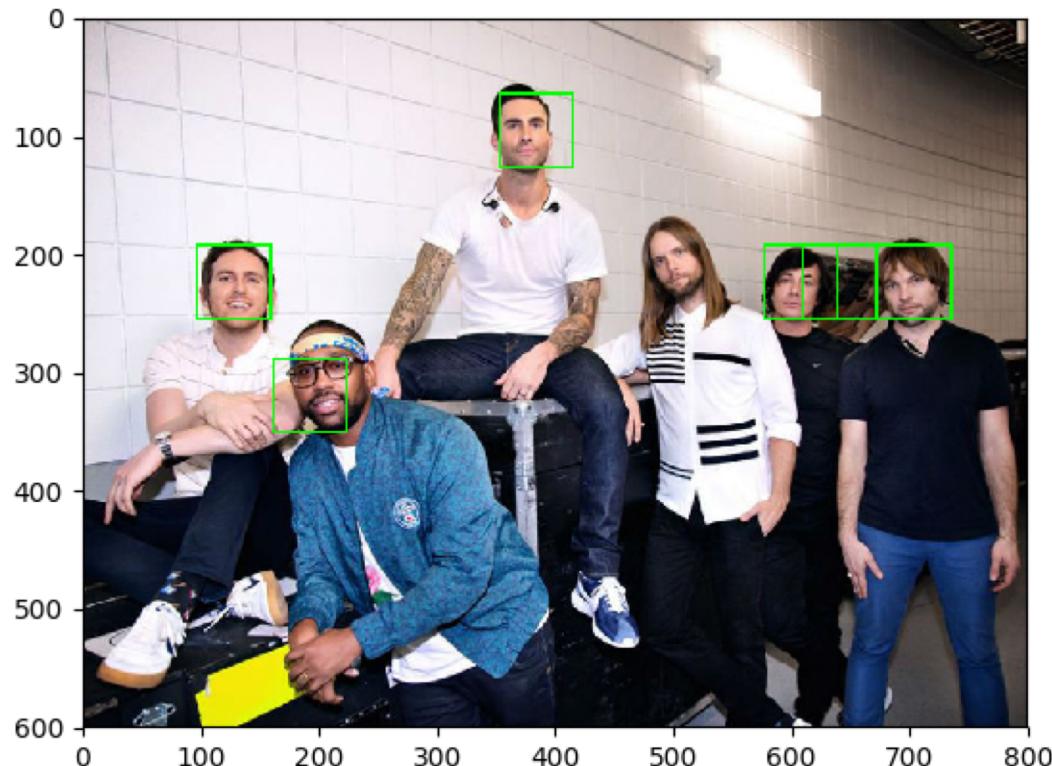
# Architecture - Verification

- 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



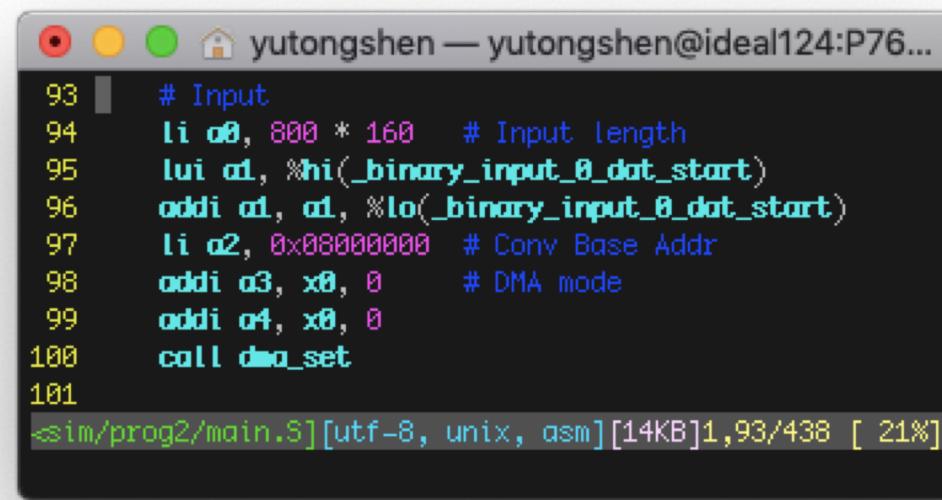
# Timing Analysis

- Cutting 4 segments



# Timing Analysis

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



The screenshot shows a terminal window with the following assembly code:

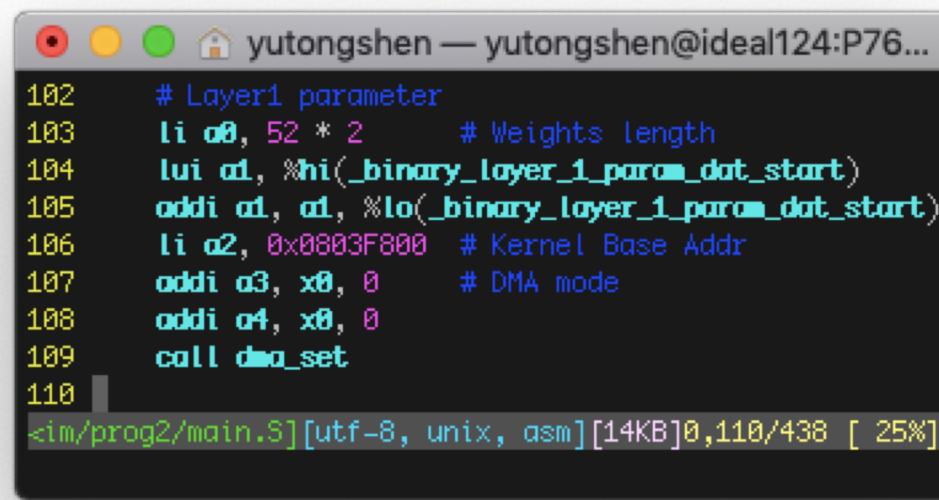
```
93 # Input
94 li a0, 800 * 160    # Input length
95 lui a1, %hi(_binary_input_0_dat_start)
96 addi a1, a1, %lo(_binary_input_0_dat_start)
97 li a2, 0x00000000  # Conv Base Addr
98 addi a3, x0, 0      # DMA mode
99 addi a4, x0, 0
100 call dma_set
101
```

The code is annotated with comments explaining its purpose. It initializes registers a0, a1, a2, a3, and a4 with specific values. Register a0 contains the input length (800 \* 160). Registers a1 and a2 are used for DMA operations, with a1 pointing to the start of the input data in memory. Register a3 is set to 0 to indicate DMA mode. Register a4 is also set to 0. Finally, the code calls the `dma_set` function.



# Timing Analysis

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



The image shows a terminal window with a dark background and light-colored text. The window title bar reads "yutongshen — yutongshen@ideal124:P76...". The code listed is assembly language:

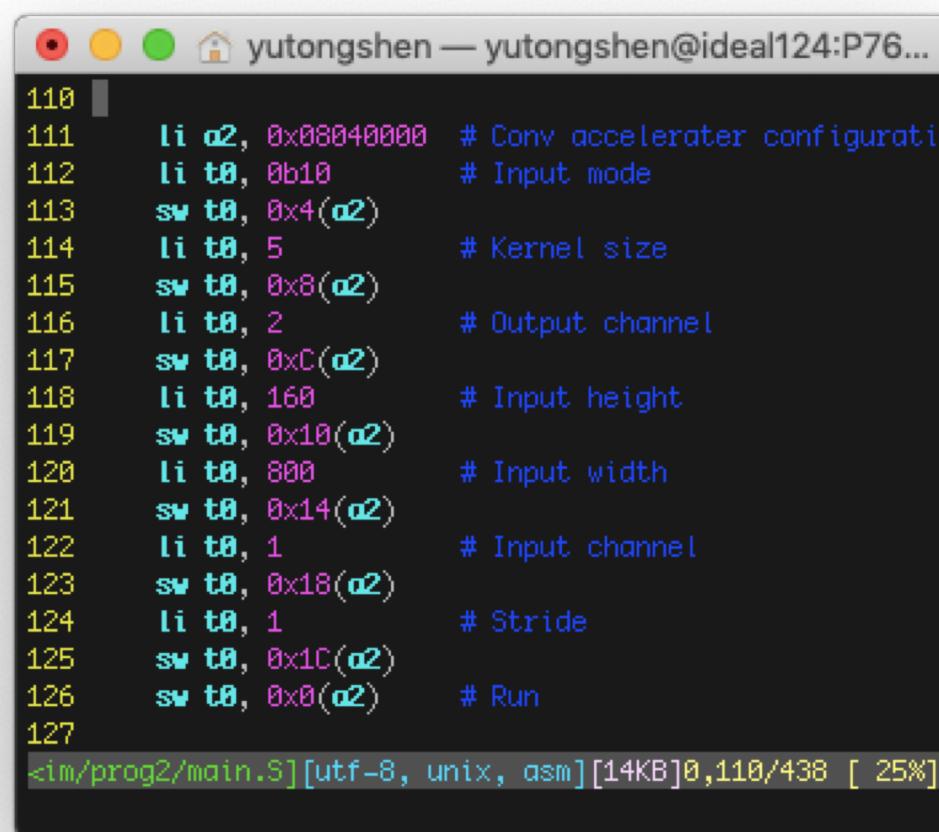
```
102    # Layer1 parameter
103    li a0, 52 * 2      # Weights length
104    lui a1, %hi(_binary_layer_1_param_dat_start)
105    addi a1, a1, %lo(_binary_layer_1_param_dat_start)
106    li a2, 0x0003F800  # Kernel Base Addr
107    addi a3, x0, 0      # DMA mode
108    addi a4, x0, 0
109    call dma_set
110
```

At the bottom of the terminal window, the status bar displays: <im/prog2/main.S> [utf-8, unix, asm] [14KB] 0,110/438 [ 25%]



# Timing Analysis

- Step 3. Setting convolutional accelerator



The screenshot shows a terminal window with a dark background and light-colored text. The window title is "yutongshen — yutongshen@ideal124:P76...". The text in the window is a sequence of assembly-like commands, each preceded by a line number from 110 to 127. The commands are:

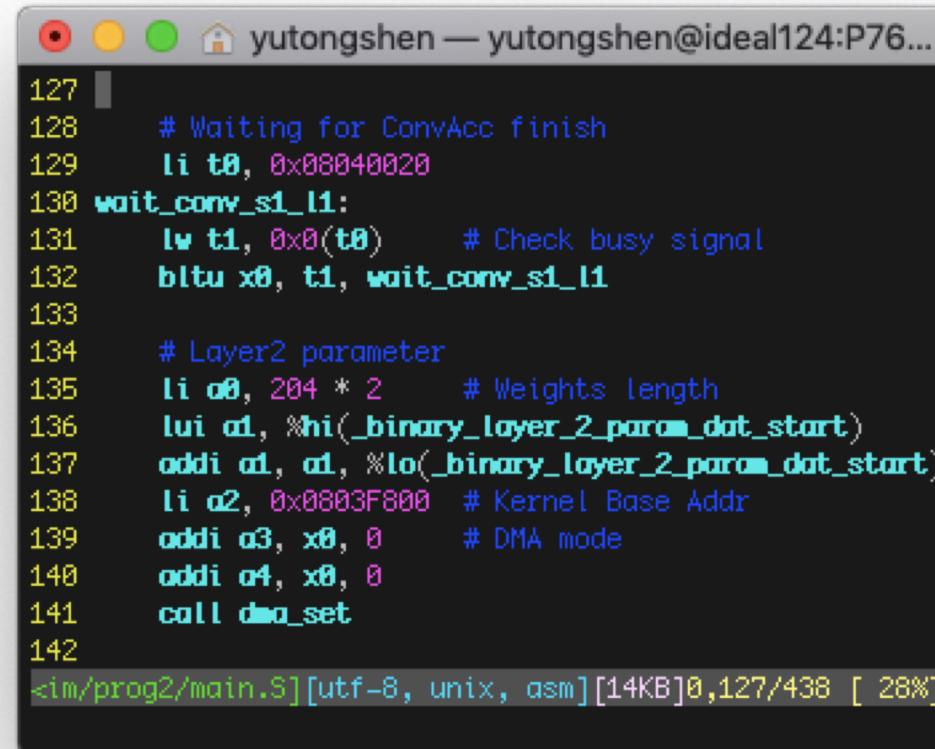
```
110
111    li a2, 0x00040000 # Conv accelerator configurati
112    li t0, 0b10        # Input mode
113    sw t0, 0x4(a2)
114    li t0, 5           # Kernel size
115    sw t0, 0x8(a2)
116    li t0, 2           # Output channel
117    sw t0, 0xC(a2)
118    li t0, 160         # Input height
119    sw t0, 0x10(a2)
120    li t0, 800         # Input width
121    sw t0, 0x14(a2)
122    li t0, 1           # Input channel
123    sw t0, 0x18(a2)
124    li t0, 1           # Stride
125    sw t0, 0x1C(a2)
126    sw t0, 0x0(a2)     # Run
127
```

At the bottom of the terminal window, the status bar displays: <im/prog2/main.S> [utf-8, unix, asm] [14KB] 0,110/438 [ 25%]



# Timing Analysis

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



The screenshot shows a terminal window with a dark background and light-colored text. The title bar reads "yutongshen — yutongshen@ideal124:P76...". The code listed is assembly language:

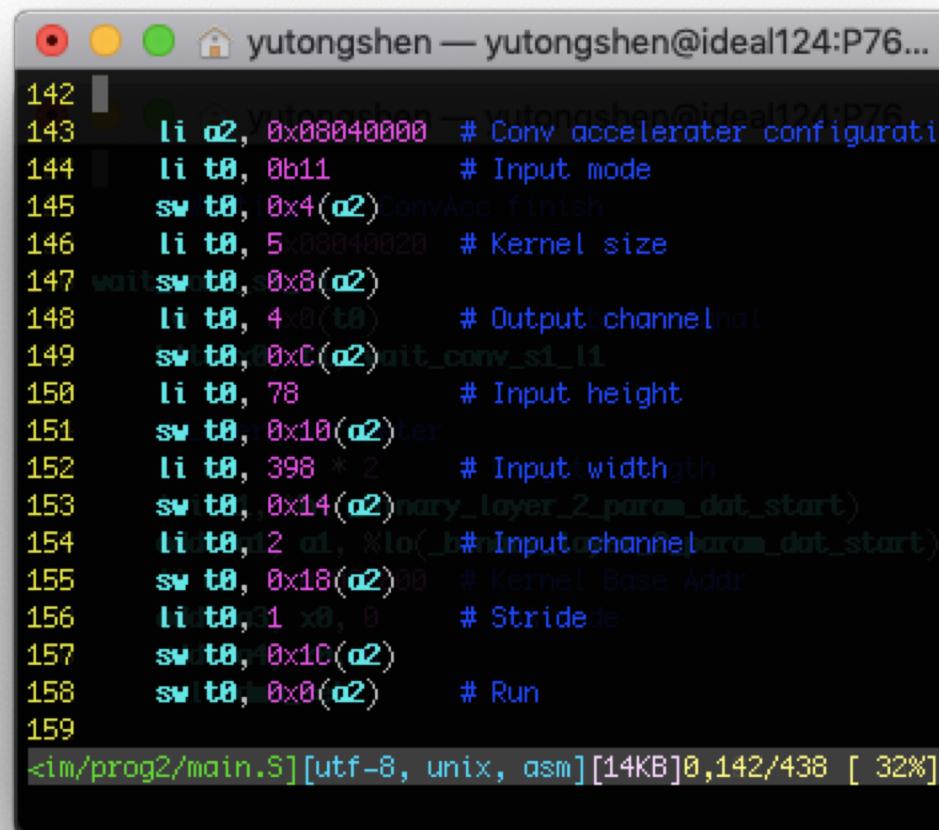
```
127
128     # Waiting for ConvAcc finish
129     li t0, 0x00040020
130 wait_conv_s1_l1:
131     lw t1, 0x0(t0)      # Check busy signal
132     bltu x0, t1, wait_conv_s1_l1
133
134     # Layer2 parameter
135     li a0, 204 * 2      # Weights length
136     lui a1, %hi(_binary_layer_2_param_dat_start)
137     addi a1, a1, %lo(_binary_layer_2_param_dat_start)
138     li a2, 0x0003F800  # Kernel Base Addr
139     addi a3, x0, 0       # DMA mode
140     addi a4, x0, 0
141     call dma_set
142
```

At the bottom of the terminal window, the status bar displays: <im/prog2/main.S> [utf-8, unix, asm] [14KB] 0,127/438 [ 28%].



# Timing Analysis

- Step 5. Setting convolutional accelerator



The screenshot shows a terminal window with the following assembly code:

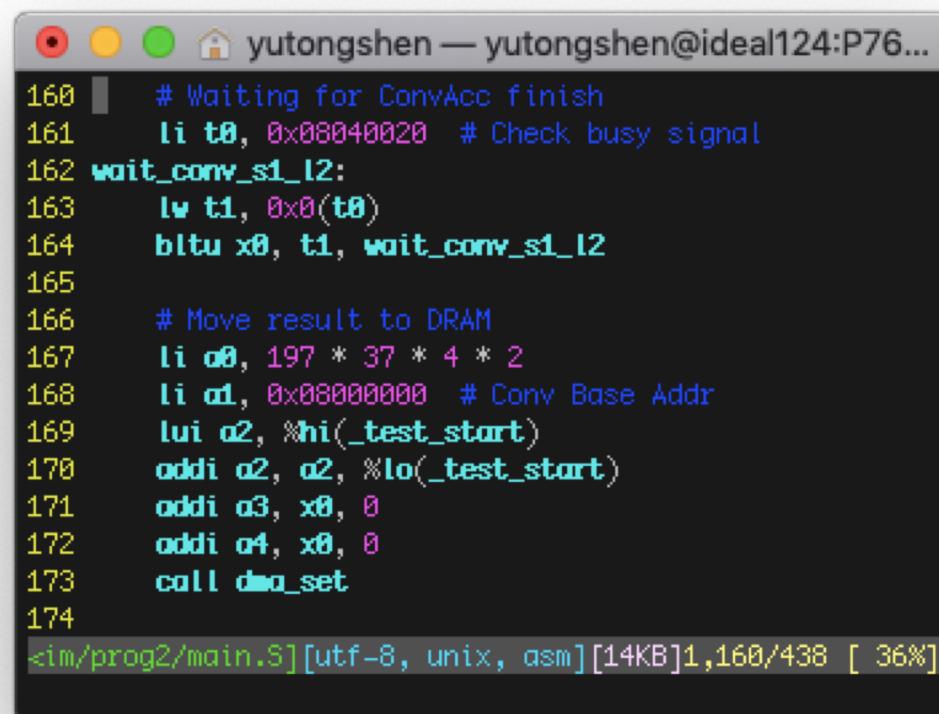
```
142
143     li a2, 0x00040000 # Conv accelerator configuration
144     li t0, 0b11          # Input mode
145     swt0, 0x4(a2)ConvAcc_finish
146     li t0, 5<00040020 # Kernel size
147     waitswt0, 0x8(a2)
148     li t0, 4<t0          # Output channel num
149     swt0, 0xC(a2)out_conv_size_l1
150     li t0, 78             # Input height
151     swt0, 0x10(a2)ter
152     li t0, 398 * 2        # Input widthngth
153     swt0, 0x14(a2)nary_layer_2_param_dat_start
154     lidt0, 12 ad, %lo(_# Input channel baron_dat_start)
155     swt0, 0x18(a2)00 # Kernel Base Addr
156     lidt0, 1 <t0, 0      # Stride le
157     swt0, 0x10(a2)
158     swt0, 0x0(a2)        # Run
159
```

The code is written in assembly language and includes comments explaining the parameters being set for a convolutional accelerator. The assembly code is part of a file named `main.S`.



# Timing Analysis

- Step 6. Moving result to DRAM



```
yutongshen — yutongshen@ideal124:P76...
160    # Waiting for ConvAcc finish
161    li t0, 0x00040020  # Check busy signal
162    wait_conv_s1_l2:
163    lw t1, 0x0(t0)
164    bltu x0, t1, wait_conv_s1_l2
165
166    # Move result to DRAM
167    li a0, 197 * 37 * 4 * 2
168    li a1, 0x00000000  # Conv Base Addr
169    lui a2, %hi(_test_start)
170    addi a2, a2, %lo(_test_start)
171    addi a3, x0, 0
172    addi a4, x0, 0
173    call dma_set
174

<im/prog2/main.S>[utf-8, unix, asm][14KB]1,160/438 [ 36%]
```



# Timing Analysis

Cycle	937,000	1,130,000	2,075,000	2,148,000
-------	---------	-----------	-----------	-----------

AHB				
CPU	Bootloader			
DMA				
Conv				

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
-------	-----------	-----------	-----------

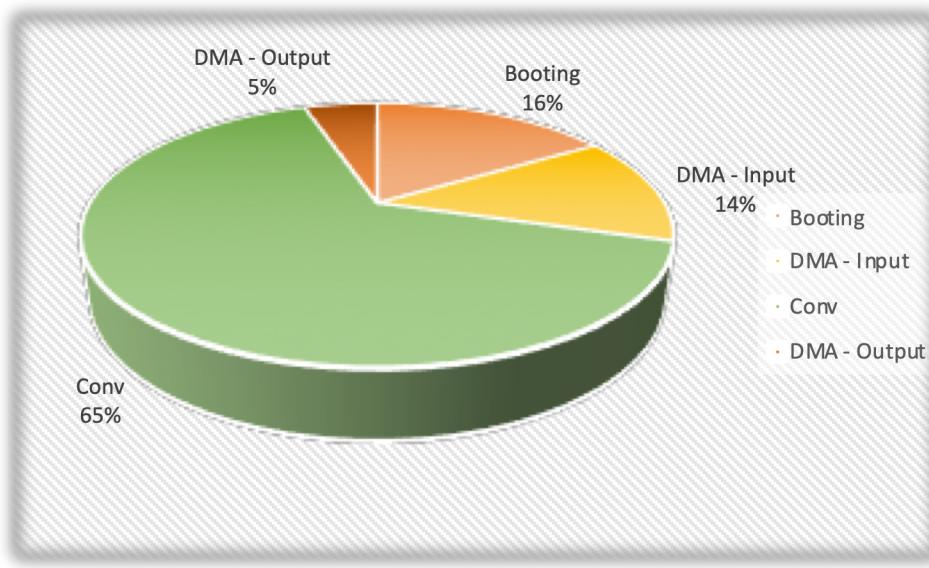
AHB			
CPU			
DMA	Move image to Conv		Move result to DRAM
Conv		Calculate segment3	

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

