Lab7: SOM Processing System

Instructor: Lih-Yih Chiou

Speaker: Willie

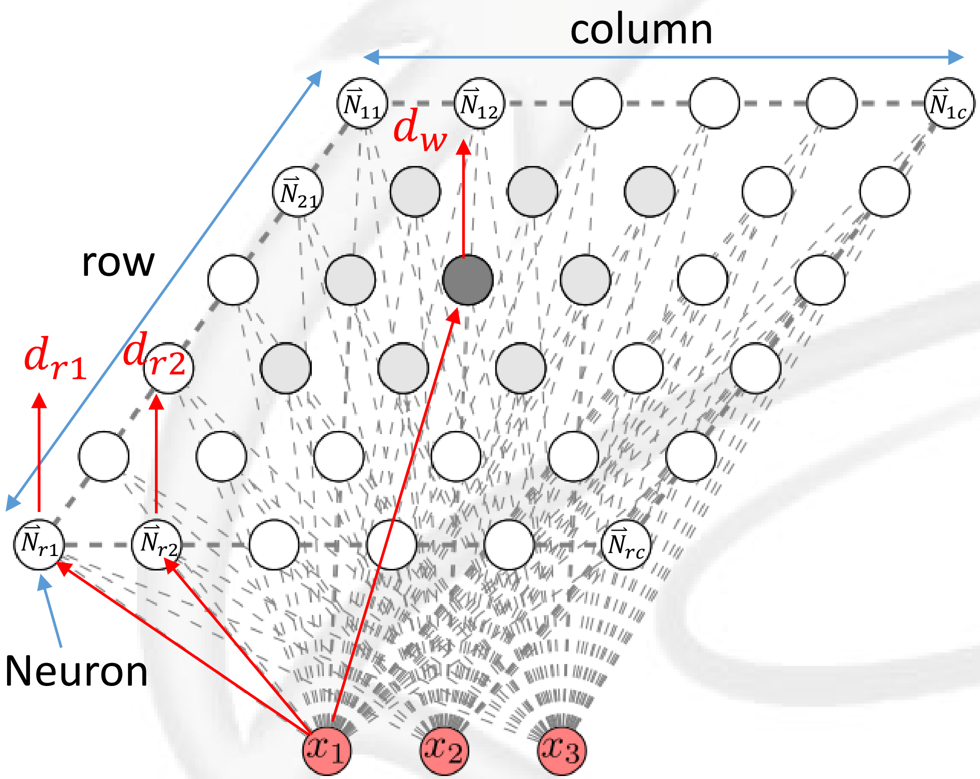
Date: 2022/04/13

EE

LPHPLAB
VLSI Design LAB

Introduction of Self-Organizing Map (SOM)

- Unsupervised machine learning
- Competitive learning



Training

1. Finding smallest distance
2. $\vec{N}_{rc}(t+1) = \vec{N}_{rc}(t) + \alpha h[\,\vec{x}(t) - \vec{N}_{rc}(t)]$,

where

α *is the learning rate*, $0 \le \alpha \le 1$

h *is neighborhood function*

$0 \le h \le 1$, the closer to the winner the larger the value is

Inference

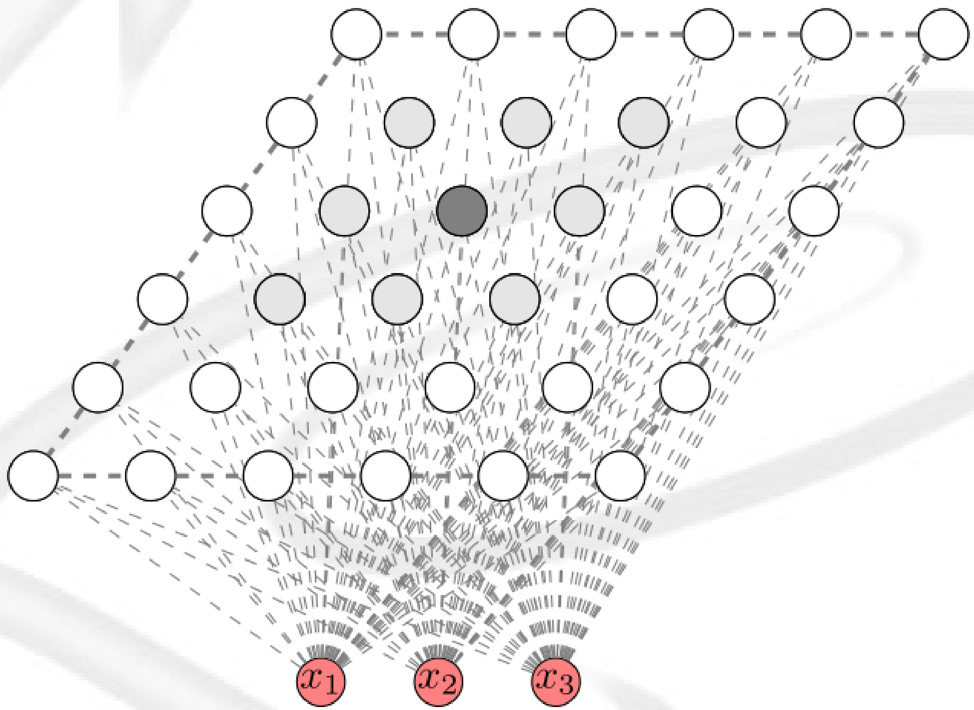
1. Finding smallest distance

Weight vector $\vec{N}_{rc} = [w_1 \cdots w_n]$

Input vector $\vec{x} = [v_1 \cdots v_n]$, where $\{x_1, x_2, x_3\} \in \vec{x}$

Introduction of Self-Organizing Map (SOM)

- Neighborhood function
  - When updating the weights, the surrounding weights are also updated



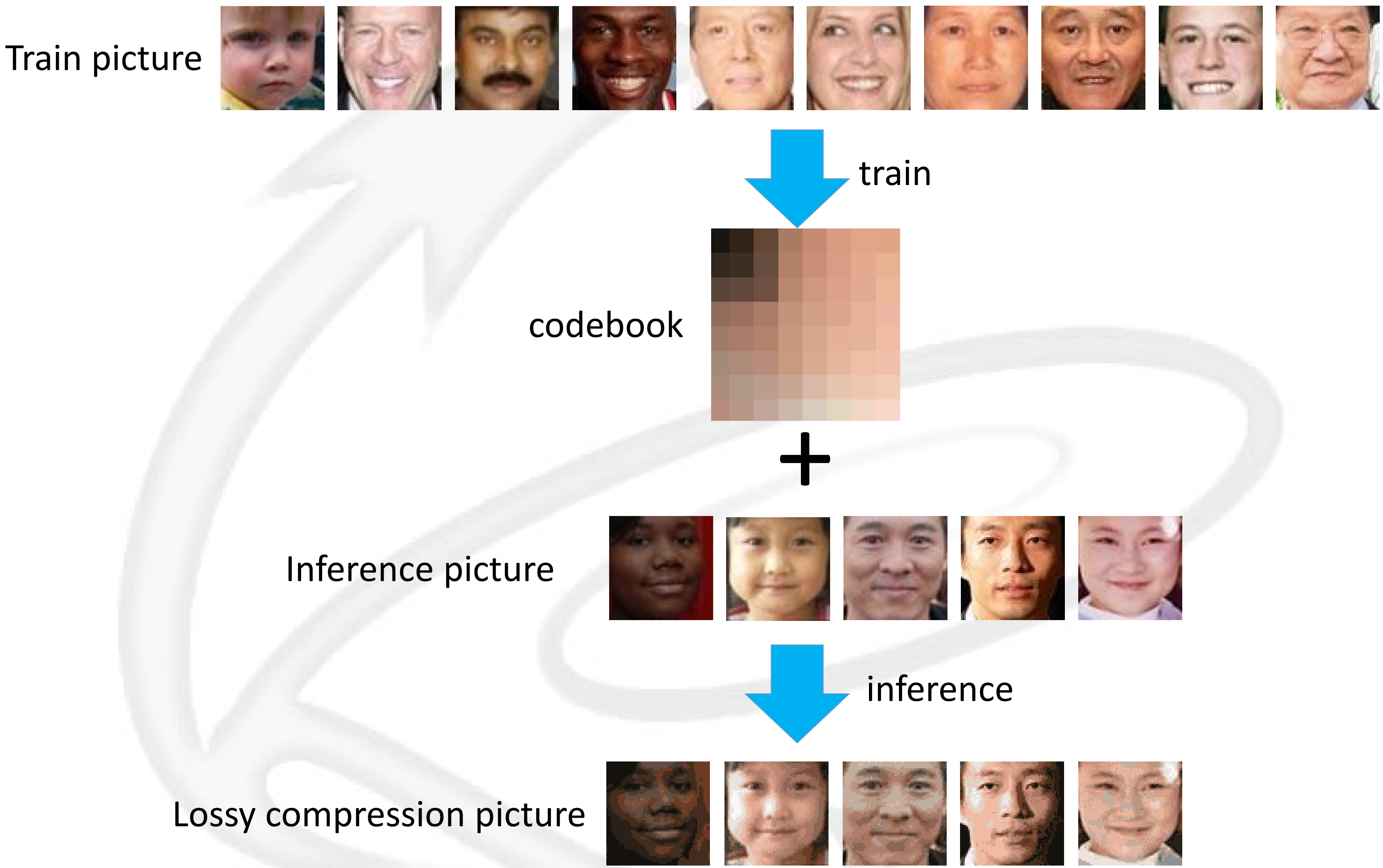
Introduction of Self-Organizing Map (SOM)

- Advantages
  - Easily interpreted and understood
  - Reducing dimensionality
  - Unsupervised machine learning
- Applications
  - Visualization
  - Compression
  - Clustering
  - Speech recognition

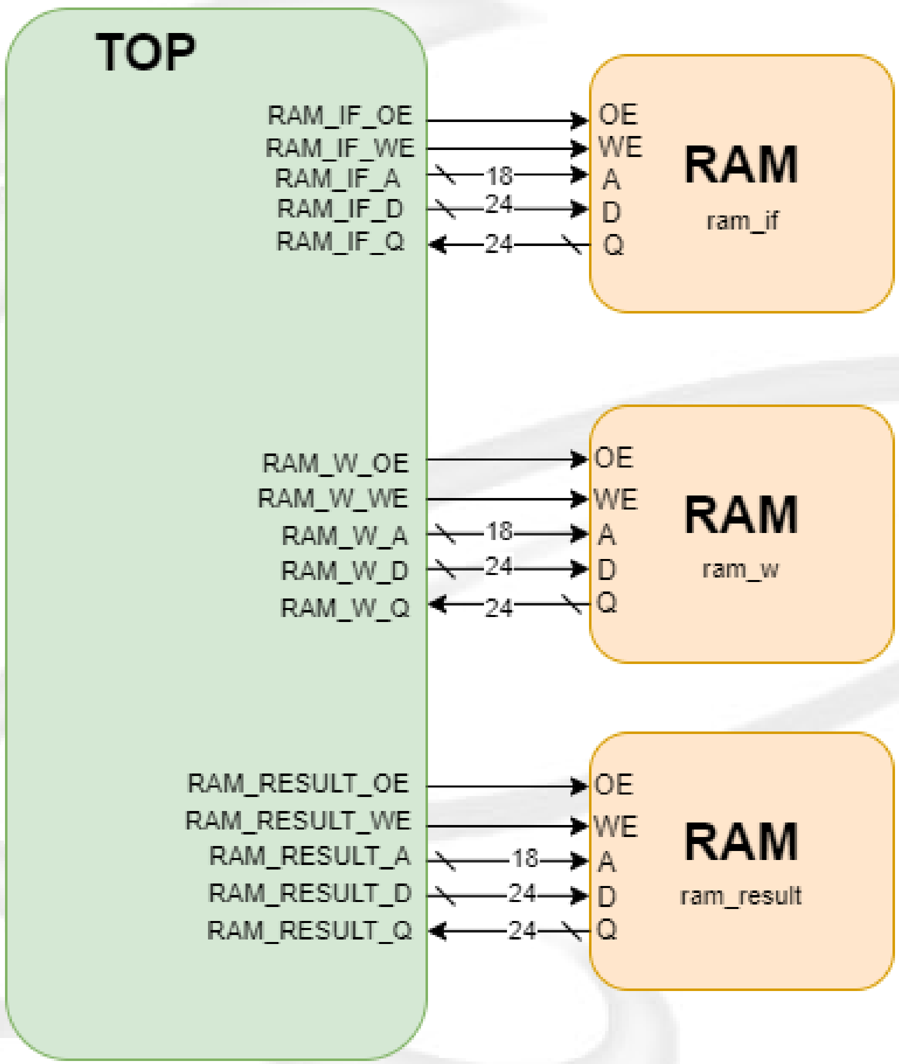
Application

- Training the SOM network to do lossy compression.
- Dataset: KinFaceW
- Image size: 64x64x3(RGB)
- Filter size: 8x8x3(RGB)

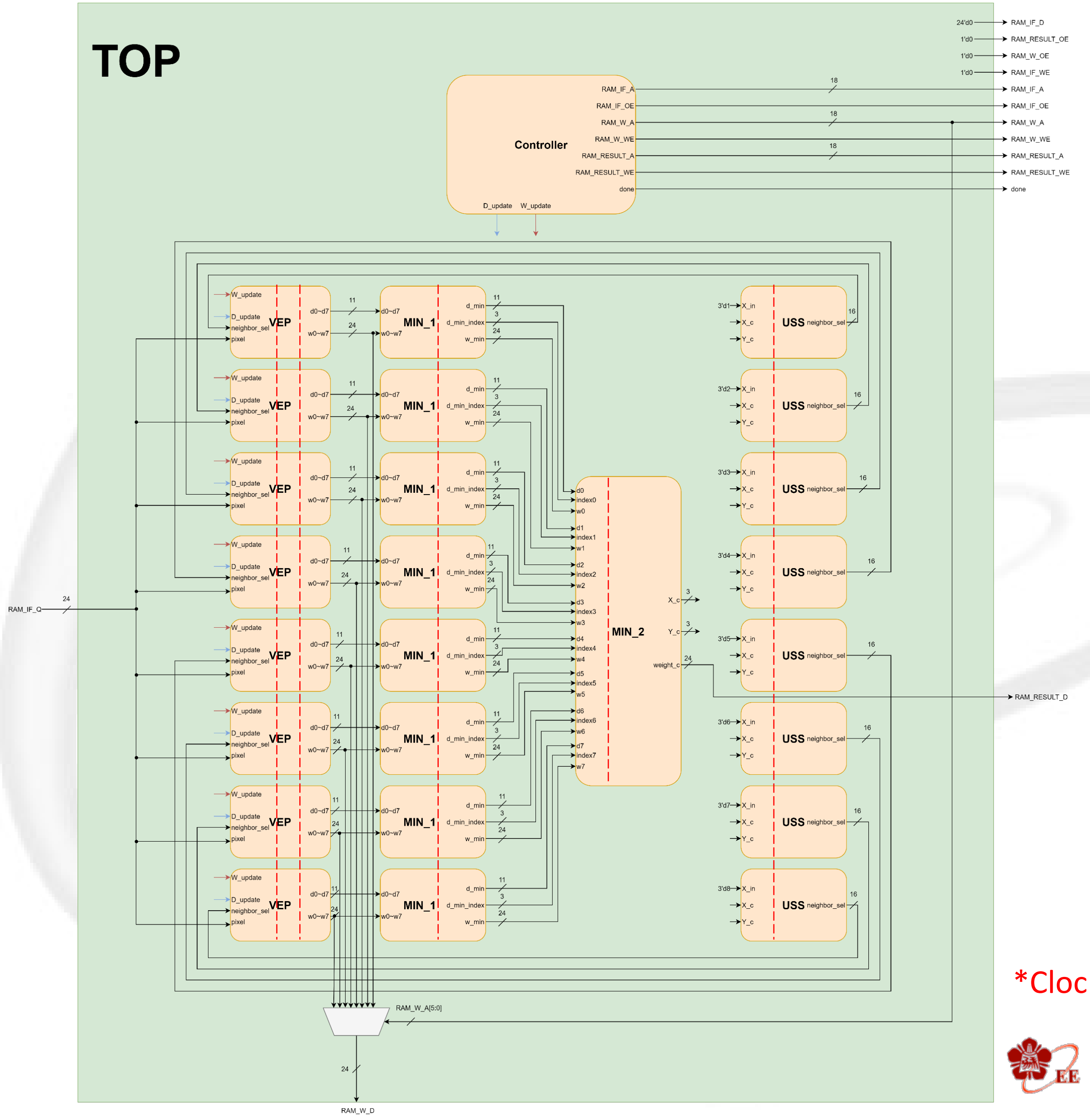
Application



Architecture(external)



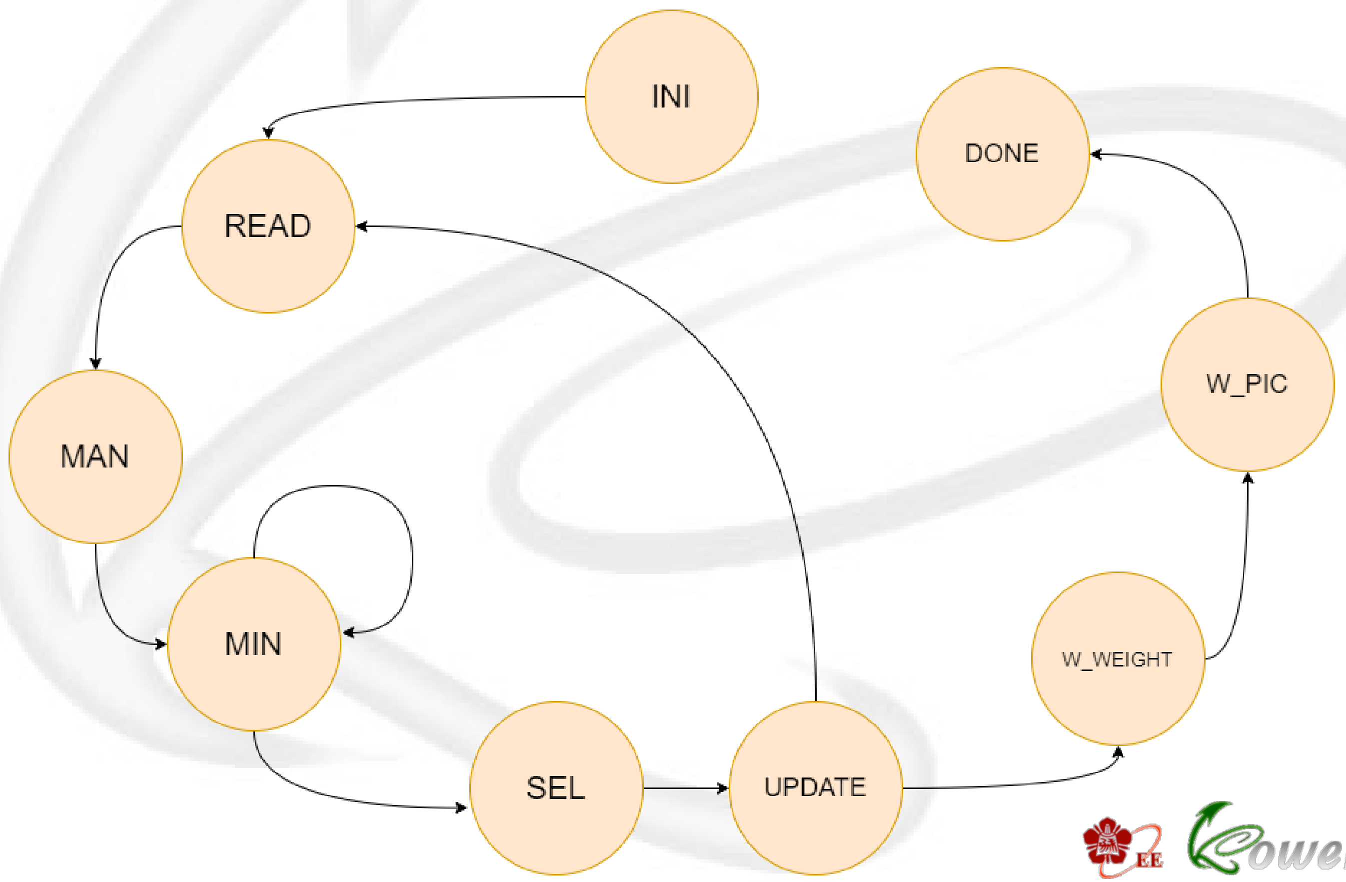
Architecture(internal)



*Clock and reset pin is ignored

Each module - Controller

- Controller
  - All operations initiated at the positive edge trigger
  - You can design your own states



Each module - Controller

- Reference states
  - INI
  - READ
    - read input pixel
  - MAN
    - calculate manhattan distance
  - MIN
    - find smallest distance
  - SEL
    - calculate neighborhood function
  - UPDATE
    - update weight value
  - W_WEIGHT
    - write weight result
  - W_PIC
    - write compress picture result
  - DONE

Each module - Controller

- Controller

Signal	I/O	bit	Description
clk	Input	1	Clock
rst	Input	1	Reset signal, active high
D_update	Output	1	Distance update enable, active high
W_update	Output	1	Weight update enable, active high
RAM_IF_A	Output	18	Input feature RAM address
RAM_IF_OE	Output	1	Input feature RAM output enable
RAM_W_A	Output	18	Weight RAM address
RAM_W_WE	Output	1	Weight RAM write enable
RAM_RESULT_A	Output	18	Result RAM address
RAM_RESULT_WE	Output	1	Result RAM write enable
done	Output	1	Pull to 1 if the system is done

Each module - VEP

- Each VEP has 8x24-bit weight memory
- Weight memory initial reset to (R,G,B) = (125,125,125)
- Weight update function
  - ➔ $\vec{N}_{rc}(t+1) = \quad \vec{N}_{rc}(t) + \alpha h[\,\vec{x}(t) - \vec{N}_{rc}(t)]$
    - α => learning rate = 0.25
    - h => neighborhood function

Each module - VEP

- Neighborhood function

0	0	0	0	0	0	0	0
0	0.125	0.125	0.125	0.125	0.125	0	0
0	0.125	0.25	0.25	0.25	0.125	0	0
0	0.125	0.25	1	0.25	0.125	0	0
0	0.125	0.25	0.25	0.25	0.125	0	0
0	0.125	0.125	0.125	0.125	0.125	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Each module - VEP

Y

X

(0,0)	(0,1)	(0,2)	(0,3)	(0,4)	(0,5)	(0,6)	(0,7)	→ VEP0
(1,0)	(1,1)	(1,2)	(1,3)	(1,4)	(1,5)	(1,6)	(1,7)	→ VEP1
(2,0)	(2,1)	(2,2)	(2,3)	(2,4)	(2,5)	(2,6)	(2,7)	→ VEP2
(3,0)	(3,1)	(3,2)	(3,3)	(3,4)	(3,5)	(3,6)	(3,7)	→ VEP3
(4,0)	(4,1)	(4,2)	(4,3)	(4,4)	(4,5)	(4,6)	(4,7)	→ VEP4
(5,0)	(5,1)	(5,2)	(5,3)	(5,4)	(5,5)	(5,6)	(5,7)	→ VEP5
(6,0)	(6,1)	(6,2)	(6,3)	(6,4)	(6,5)	(6,6)	(6,7)	→ VEP6
(7,0)	(7,1)	(7,2)	(7,3)	(7,4)	(7,5)	(7,6)	(7,7)	→ VEP7

Each module - VEP

- VEP

Signal	I/O	Bit	Description
clk	Input	1	Clock
rst	Input	1	Reset signal, active high
W_update	Input	1	Weight update enable, active high
D_update	Input	1	Distance update enable, active high
neighbor_sel	Input	16(2x8)	Neighborhood function of 8 VEP weights (00=>1, 01=>0.25, 10=>0.125, 11=>0)
pixel	Input	24	Input pixel from RAM_if
d0~d7	Output	11	Manhattan distance between 8 weights
w0~w7	Output	24	8 weights

Each module – MIN_1

- MIN_1

Signal	I/O	bit	Description
clk	Input	1	Clock
rst	Input	1	Reset signal, active high
d0~d7	Input	11	Manhattan distance between 8 weights
w0~w7	Input	24	8 weights
d_min	Output	11	Minimum distance between d0~d7
d_min_index	Output	3	Index of minimum distance
W_min	output	24	Weight of minimum distance

Each module – MIN_2

- MIN_2

signal	I/O	bit	Description
clk	Input	1	clock
rst	Input	1	Reset signal, active high
d0~d7	Input	11	Minimum distance from MIN_1
w0~w7	Input	24	Weight of minimum distance from MIN_1
index0~index7	Input	3	Index of minimum distance from MIN_1
X_c	Output	3	The X coordinate of center weight
Y_c	Output	3	The Y coordinate of center weight
weight_c	output	24	Center weight

Each module - USS

- USS

signal	I/O	bit	Description
clk	input	1	Clock
rst	Input	1	Reset signal, active high
X_in	Input	3	USS module index
X_c	Input	3	The X coordinate of center weight
Y_c	Input	3	The Y coordinate of center weight
neighbor_sel	Output	16(2x8)	Neighborhood function of 8 VEP weights (00=>1, 01=>0.25, 10=>0.125, 11=>0)

Each module - TOP

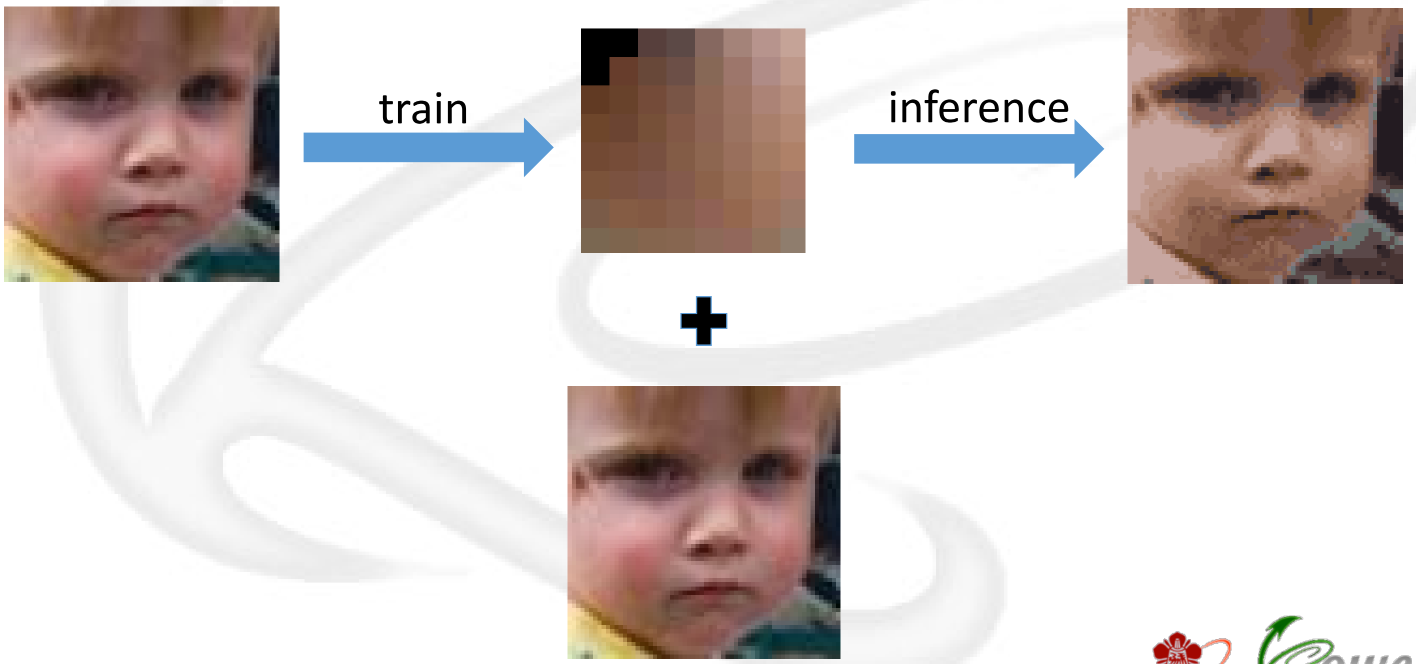
TOP

Signal	I/O	bit	Description
clk	Input	1	Clock
rst	Input	1	Reset signal, active high
RAM_IF_Q RAM_W_Q RAM_RESULT_Q	Input	24	Data output from RAM
RAM_IF_OE RAM_W_OE RAM_RESULT_OE	Output	1	RAM output enable signal
RAM_IF_WE RAM_W_WE RAM_RESULT_WE	Output	1	RAM write enable signal
RAM_IF_A RAM_W_A RAM_RESULT_A	Output	18	RAM address
RAM_IF_D RAM_W_D RAM_RESULT_D	output	24	Data written into RAM
done	Output	1	Pull to 1 if the system is done

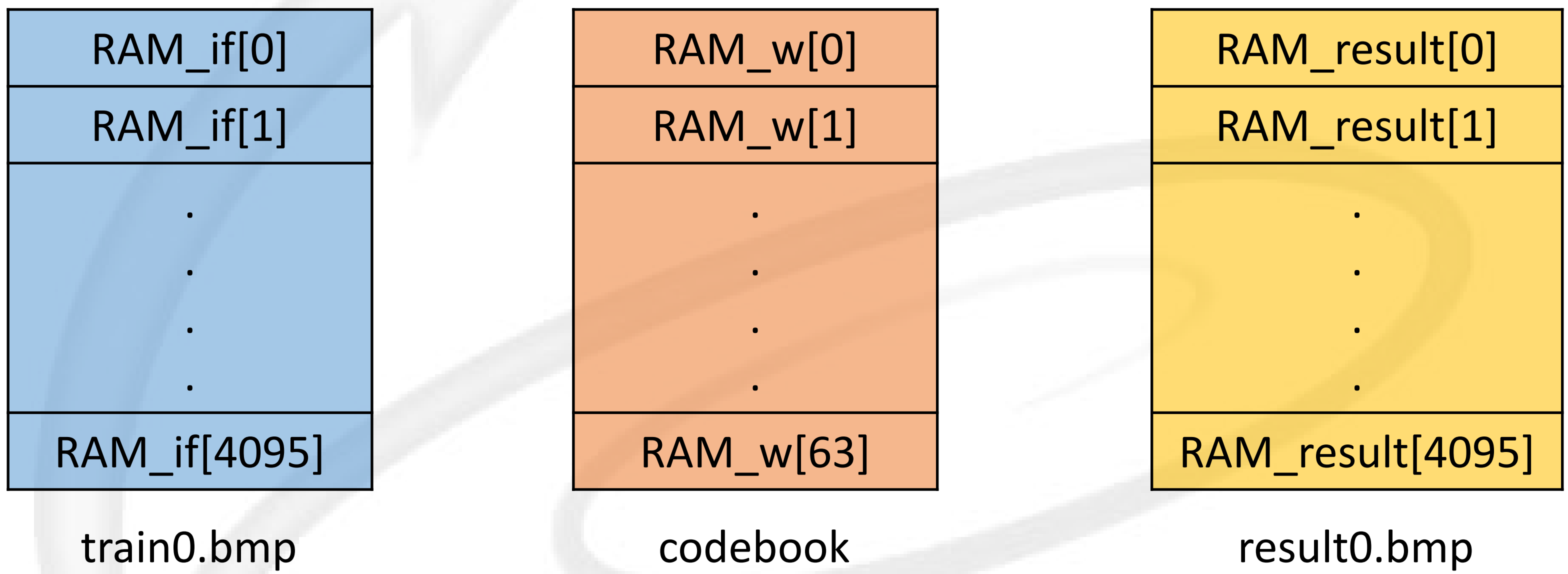
Lab7_1

Introduction

- Using one picture to train a codebook
- Compress the same picture with trained codebook



RAM

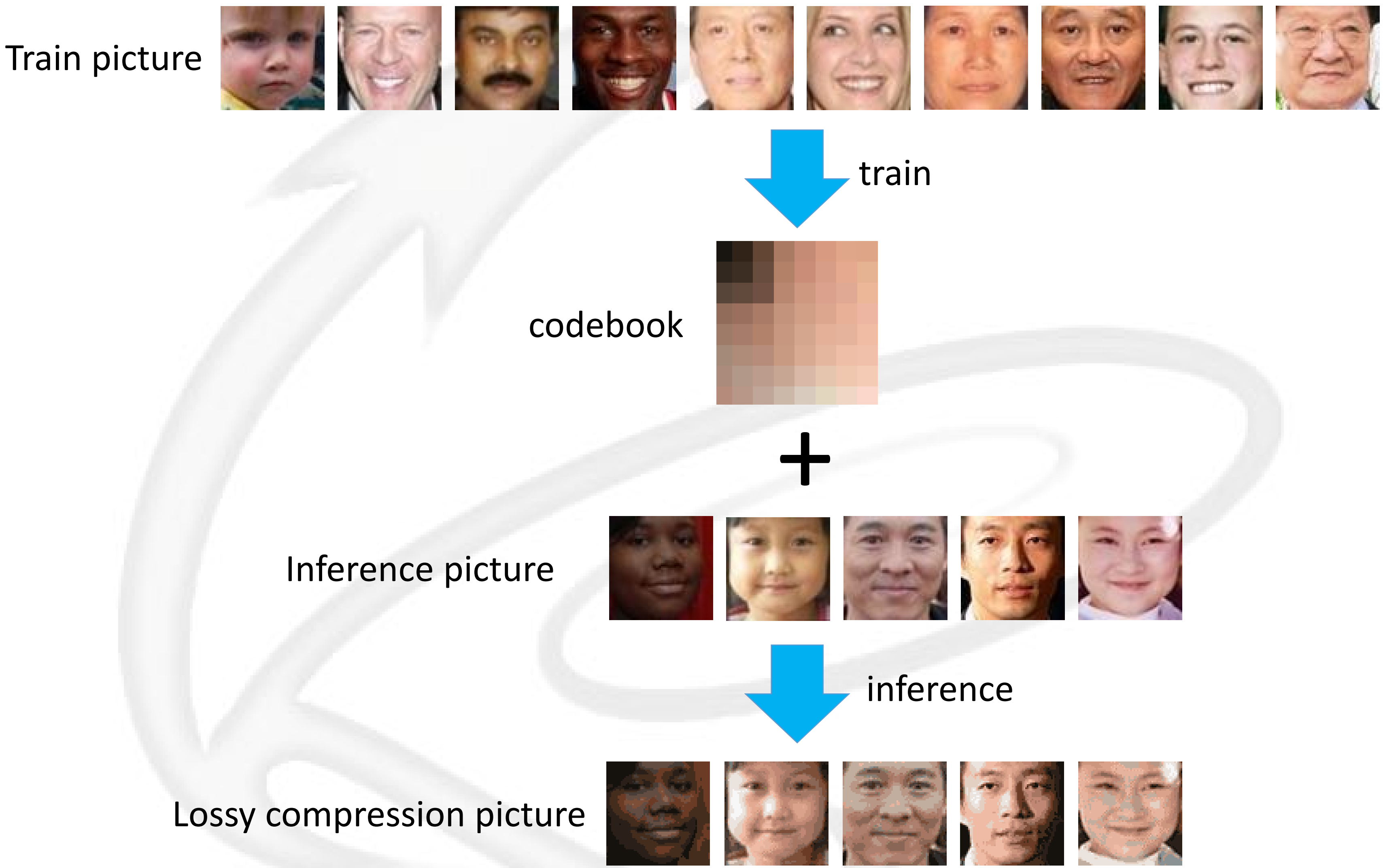

RAM_if[0]
RAM_if[1]
RAM_if[4095]
train0.bmp
RAM_w[0]
RAM_w[1]
RAM_w[63]
codebook
RAM_result[0]
RAM_result[1]
RAM_result[4095]
result0.bmp

Lab7_2

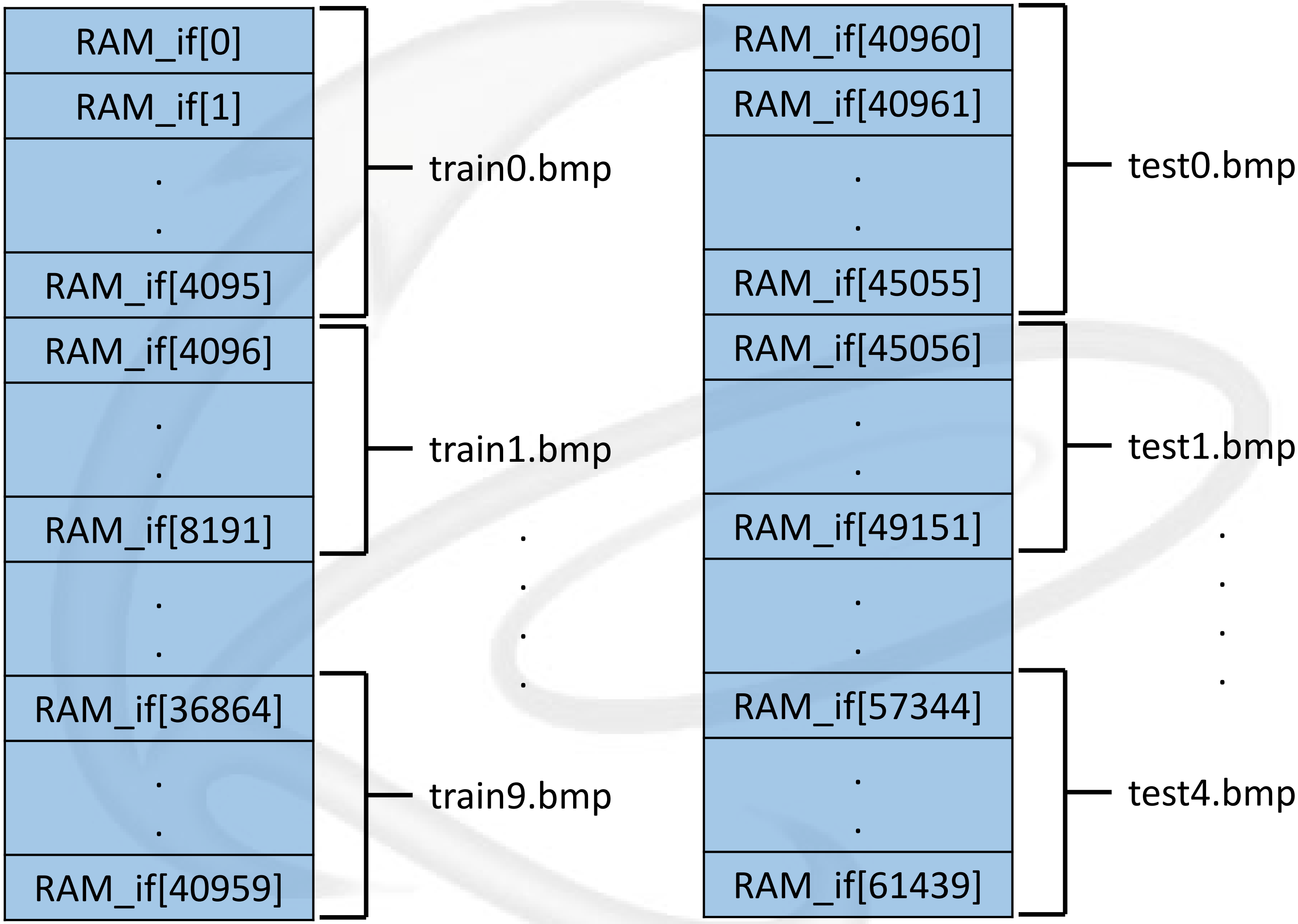
Introduction

- Using ten photos to train a codebook
- Compress different five photos with trained codebook

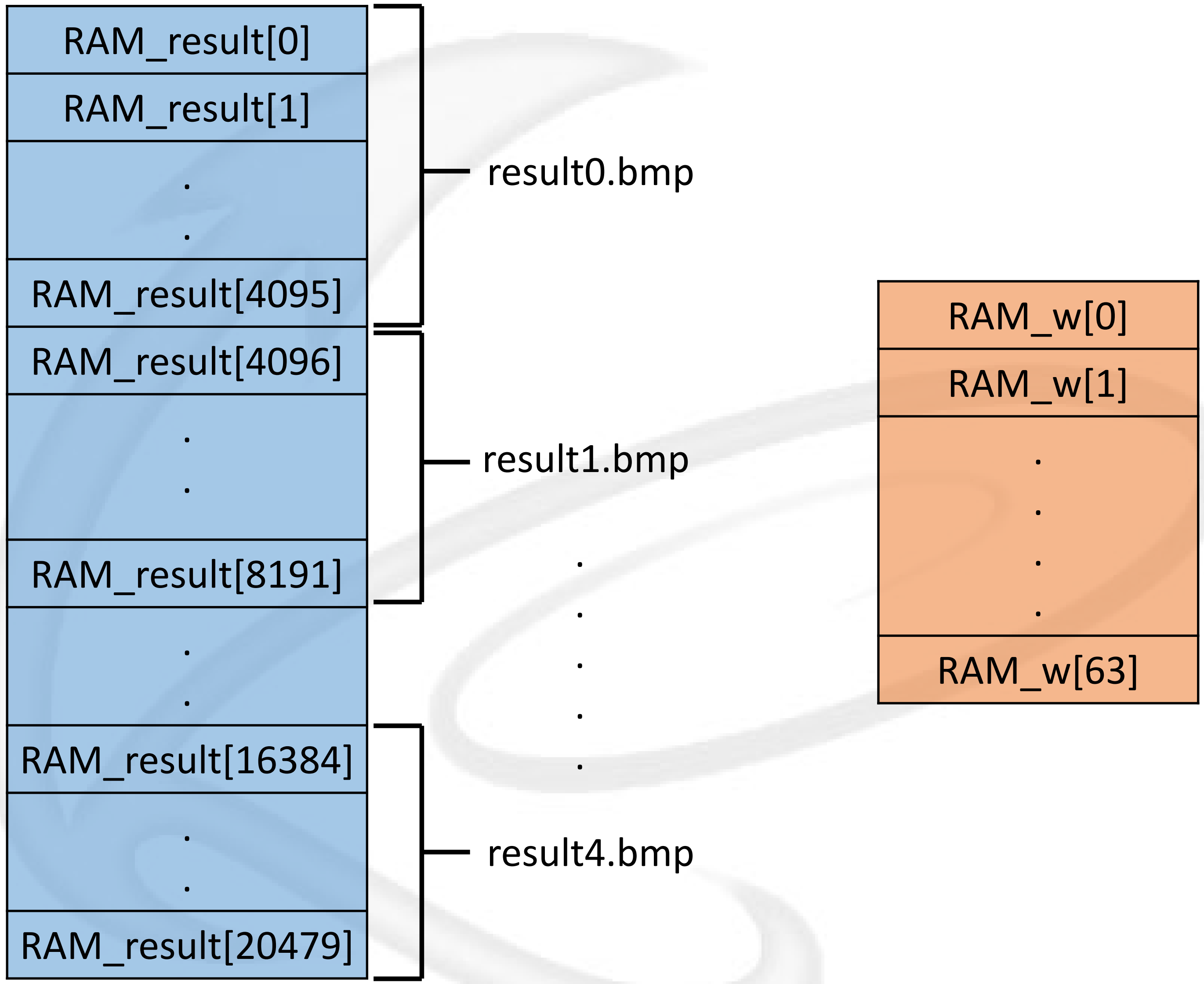
Introduction



RAM


RAM_if[0]
RAM_if[1]
RAM_if[4095]
RAM_if[4096]
RAM_if[8191]
RAM_if[36864]
RAM_if[40959]
train0.bmp
train1.bmp
train9.bmp
RAM_if[40960]
RAM_if[40961]
RAM_if[45055]
RAM_if[45056]
RAM_if[49151]
RAM_if[57344]
RAM_if[61439]
test0.bmp
test1.bmp
test4.bmp

RAM


RAM_result[0]
RAM_result[1]
RAM_result[4095]
RAM_result[4096]
RAM_result[8191]
RAM_result[16384]
RAM_result[20479]
result0.bmp
result1.bmp
result4.bmp
RAM_w[0]
RAM_w[1]
RAM_w[63]

Simulation command

Problem	Command
Lab7_1(pre-sim)	ncverilog top_tb.v +define+X (WEIGHT, RESULT, FULL)
Lab7_1 (pre-sim with waveform)	ncverilog top_tb.v +access+r +define+FSDB+X
Lab7_1(post-sim)	ncverilog top_tb.v +define+syn+X
Lab7_1 (post-sim with waveform)	ncverilog top_tb.v +access+r +define+FSDB+syn+X
Lab7_2(pre-sim)	ncverilog top_tb.v +define+X (WEIGHT, RESULT0, RESULT1, RESULT2, RESULT3, RESULT4, FULL)
Lab7_2 (pre-sim with waveform)	ncverilog top_tb.v +access+r +define+FSDB+X
Lab7_2(post-sim)	ncverilog top_tb.v +define+FSDB+syn+X
Lab7_2 (post-sim with waveform)	ncverilog top_tb.v +access+r +define+FSDB+syn+X

Lab session 7

- Please complete Lab session 7
  - Allow 1-2 members a group
  - Lab 7_1
    - Design a simple SOM system which can train a codebook and compress one picture
      - Make sure you can implement your design under 20ns clock period
  - Lab 7_2
    - Design a simple SOM system which can train 10 training pictures and compress 5 testing picture
      - Make sure you can implement your design under 20ns clock period

Lab session 7

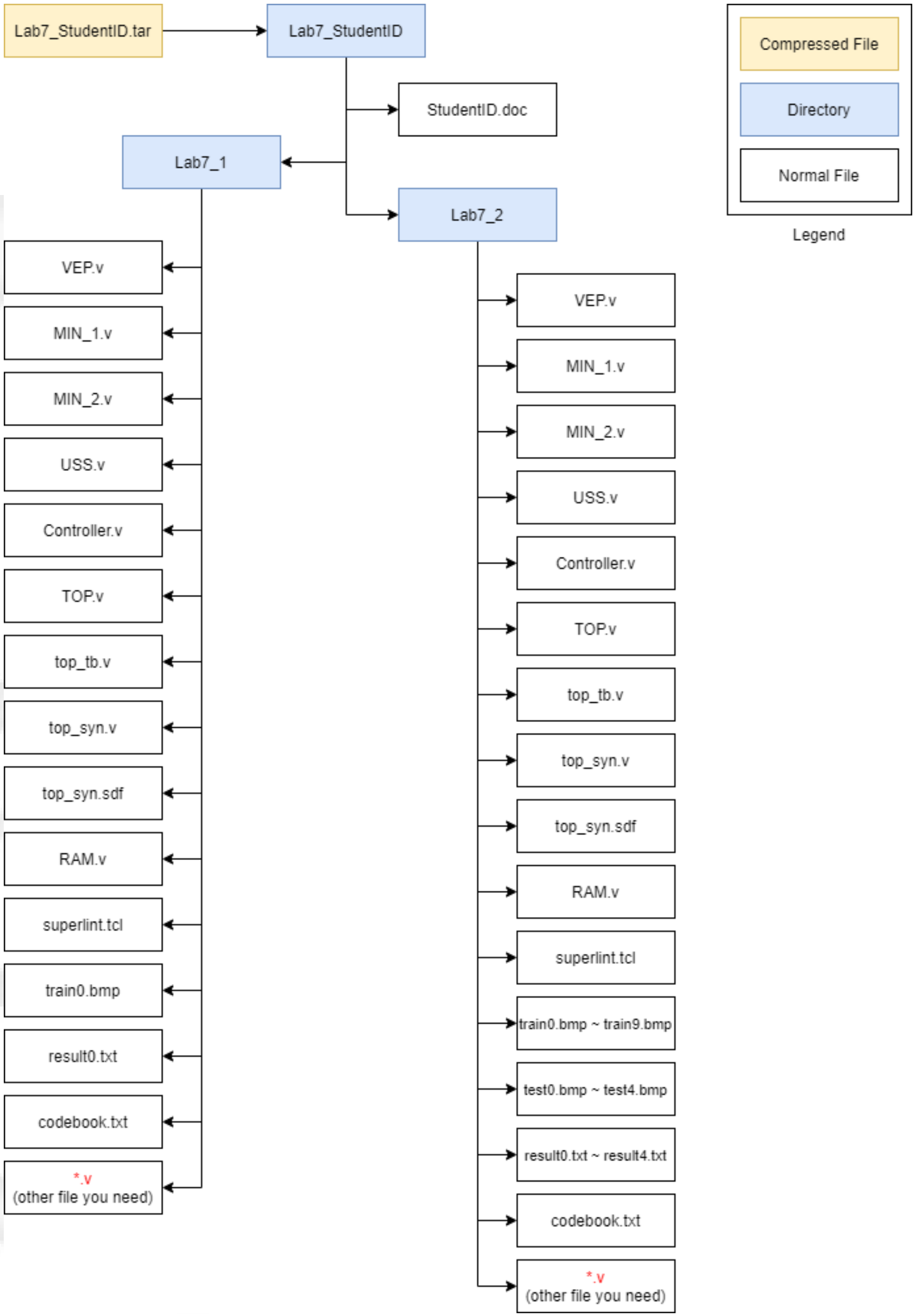
- Attention
  - Make sure all your Verilog code can be compiled on the environment in SoC Lab
  - After uploading files, please be sure to download and simulate again. If an error file is submitted and due time is missed, it will be treated as late. The late submission rules are the same as the syllabus

Lab session 7

- Grading policy
  - Lab7_1 (45%)
    - RTL pass (15%)
    - SYN pass (10%)
    - Report (10%)
    - PA (5%)
    - Superlint >= 90% (5%)
  - Lab7_2 (45%)
    - RTL pass (15%)
    - SYN pass (10%)
    - Report (10%)
    - PA (5%)
    - Superlint>=90% (5%)
  - Demo (10%)

Lab session 7

- File hierarchy



Thanks for listening