Lab7: Perspective Distortion Correction(PDC)

Instructor: Lih-Yih Chiou

Speaker: Jay

Date: 2024/04/11

LPHPLAB
VLSI Design LAB

Outline

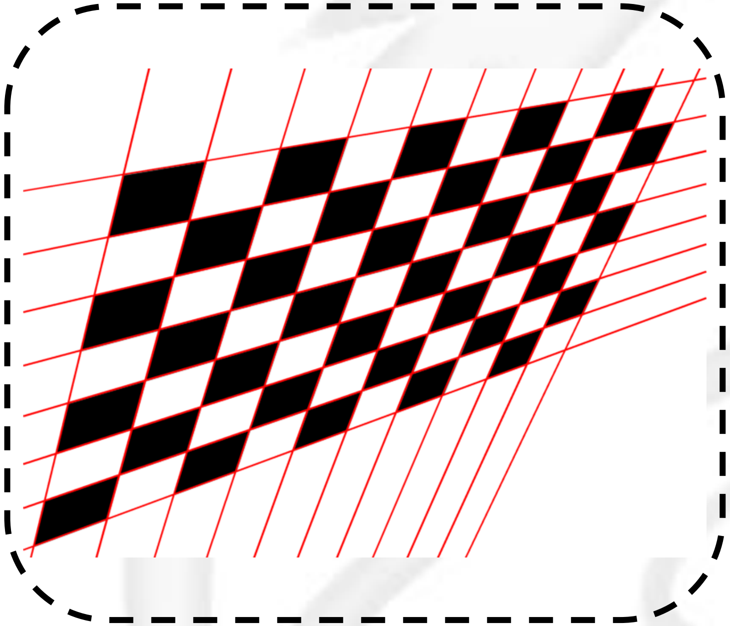
- Introduce to Perspective Transformation
- Hardware description
  - Block diagram
  - I/O Information
  - Memory mapping
  - Flow in Lab7
- Lab7 Implementation
  - Data format
  - Rounding
- Criteria
  - Simulation Result
  - Grading policy
  - Requirement & file format

Introduce to Perspective Transformation

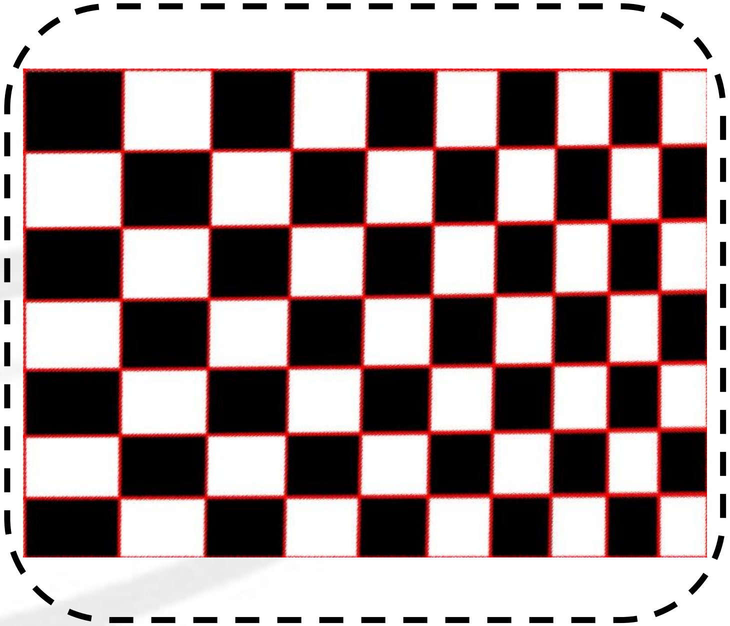
- Perspective Transformation
  - A method applied to transform images, commonly used for converting side-view images into top-view images.
  - Helping in the understanding and interpretation of 2-D images captured from a 3-D world.
  - Application scenarios:
    - Computer Vision and Image Processing
    - Autonomous Driving
    - AI (Data augmentation)

Introduce to Perspective Transformation

- Perspective Transformation



Original Image



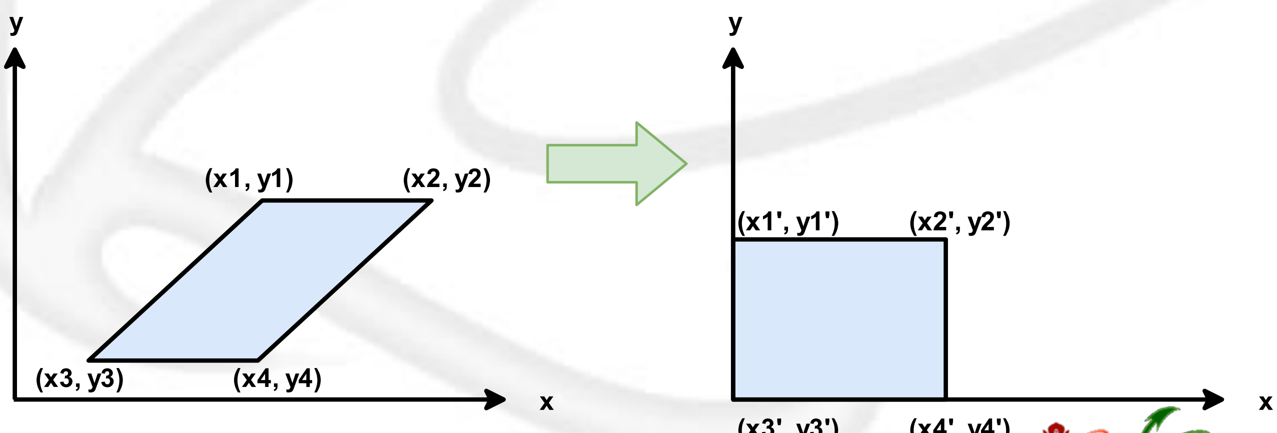
Transformed Image

Introduce to Perspective Transformation

- Perspective Transformation
  - In mathematics, such a transformation can be represented by the following equations:

$$\begin{cases} x_i = a \cdot x_i' + b \cdot y_i' + c \cdot x_i' \cdot y_i' + d \\ y_i = e \cdot x_i' + f \cdot y_i' + g \cdot x_i' \cdot y_i' + h \end{cases}$$

  - If we know the coordinates (x_i, y_i) and (x_i', y_i'), then we can find the variables a to h.



Introduce to Perspective Transformation

- Perspective Transformation
$$\begin{cases} x_i = a \cdot x_i' + b \cdot y_i' + c \cdot x_i' \cdot y_i' + d \\ y_i = e \cdot x_i' + f \cdot y_i' + g \cdot x_i' \cdot y_i' + h \end{cases}$$

  - Find variables a to h
  - $A \cdot x = b$
  - $x = A^{-1} \cdot b$
  - $A^{-1} = \frac{adj(A)}{\det(A)}$

$$\begin{bmatrix} x1' & y1' & x1'y1' & 1 & 0 & 0 & 0 & 0 \\ x1' & y1' & x1'y1' & 1 & 0 & 0 & 0 & 0 \\ x1' & y1' & x1'y1' & 1 & 0 & 0 & 0 & 0 \\ x1' & y1' & x1'y1' & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & x1' & y1' & x1'y1' & 1 \\ 0 & 0 & 0 & 0 & x1' & y1' & x1'y1' & 1 \\ 0 & 0 & 0 & 0 & x1' & y1' & x1'y1' & 1 \\ 0 & 0 & 0 & 0 & x1' & y1' & x1'y1' & 1 \end{bmatrix} \times \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \end{bmatrix} = \begin{bmatrix} x1 \\ x2 \\ x3 \\ x4 \\ y1 \\ y2 \\ y3 \\ y4 \end{bmatrix}$$

$$\begin{bmatrix} x1' & y1' & x1'y1' & 1 \\ x1' & y1' & x1'y1' & 1 \\ x1' & y1' & x1'y1' & 1 \\ x1' & y1' & x1'y1' & 1 \end{bmatrix} \times \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} x1 \\ x2 \\ x3 \\ x4 \end{bmatrix}$$

$$\begin{bmatrix} x1' & y1' & x1'y1' & 1 \\ x1' & y1' & x1'y1' & 1 \\ x1' & y1' & x1'y1' & 1 \\ x1' & y1' & x1'y1' & 1 \end{bmatrix} \times \begin{bmatrix} e \\ f \\ g \\ h \end{bmatrix} = \begin{bmatrix} y1 \\ y2 \\ y3 \\ y4 \end{bmatrix}$$

matrix A

The values of adj(A) and det(A), which are stored in RAM, can be used to find the inverse matrix of A.

adj(A) -> adjoint matrix of A
det(A) -> determinant of A

Introduce to Perspective Transformation

Perspective Transformation

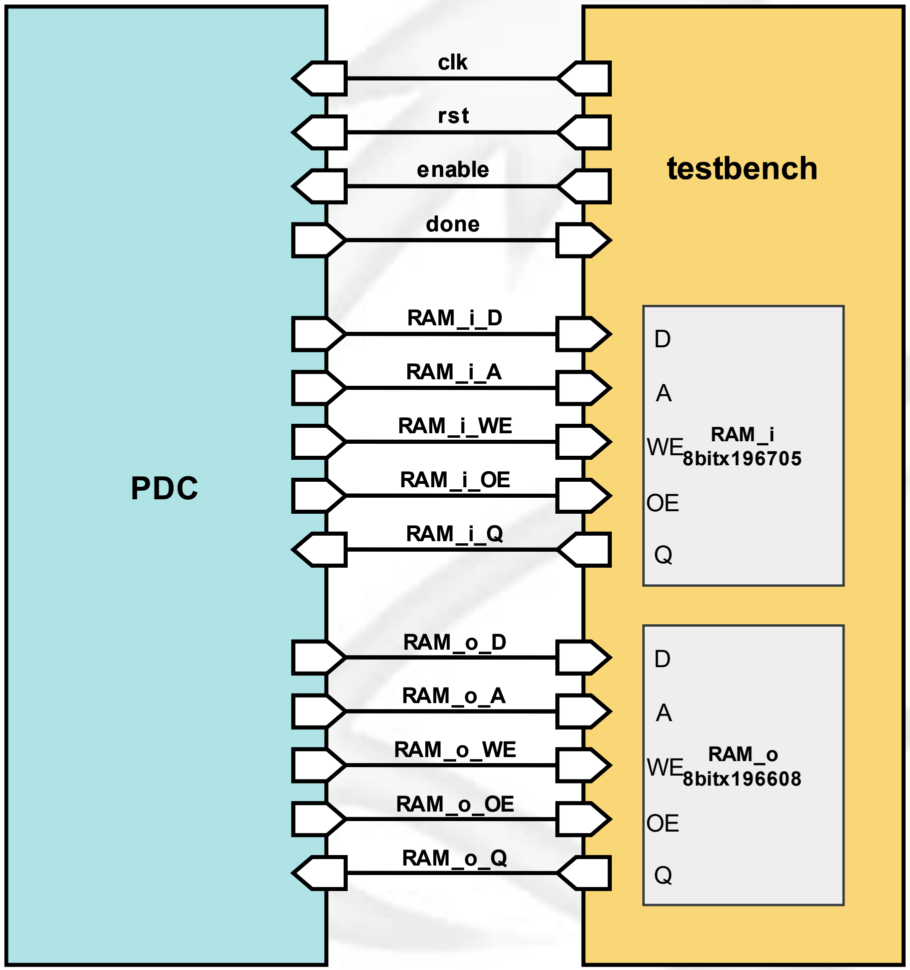
- Adjoint matrix (伴隨矩陣)
  - The adjoint matrix of A is the transpose of its cofactor matrix and is denoted by adj(A). → adj(A) = C^T (C: Cofactor Matrix)
  - Cofactor Matrix (餘因子矩陣)
    - $C_{i,j} = (-1)^{i+j} \cdot \det(M_{i,j})$
    - $M_{i,j}$ is the (n-1)x(n-1) matrix made by removing the ROW i and COLUMN j of A.

$$\operatorname{adj}(\mathbf{A}) = \mathbf{C}^\mathsf{T} = \begin{bmatrix} +\begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix} & -\begin{vmatrix} a_{12} & a_{13} \\ a_{32} & a_{33} \end{vmatrix} & +\begin{vmatrix} a_{12} & a_{13} \\ a_{22} & a_{23} \end{vmatrix} \\ -\begin{vmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{vmatrix} & +\begin{vmatrix} a_{11} & a_{13} \\ a_{31} & a_{33} \end{vmatrix} & -\begin{vmatrix} a_{11} & a_{13} \\ a_{21} & a_{23} \end{vmatrix} \\ +\begin{vmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{vmatrix} & -\begin{vmatrix} a_{11} & a_{12} \\ a_{31} & a_{32} \end{vmatrix} & +\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} \end{bmatrix}$$

…Example of 3x3 matrix

Hardware description

Block Diagram



1.adj(A) ...40-bit integer (signed)
2.det(a) ...40-bit integer & 32-bit fraction (signed)
3.coordinate of original image:
(x1, y1)~(x4, y4)
4.original image(size:256*256, RGB)

Transformed image (your output)
(size:256*256, RGB)

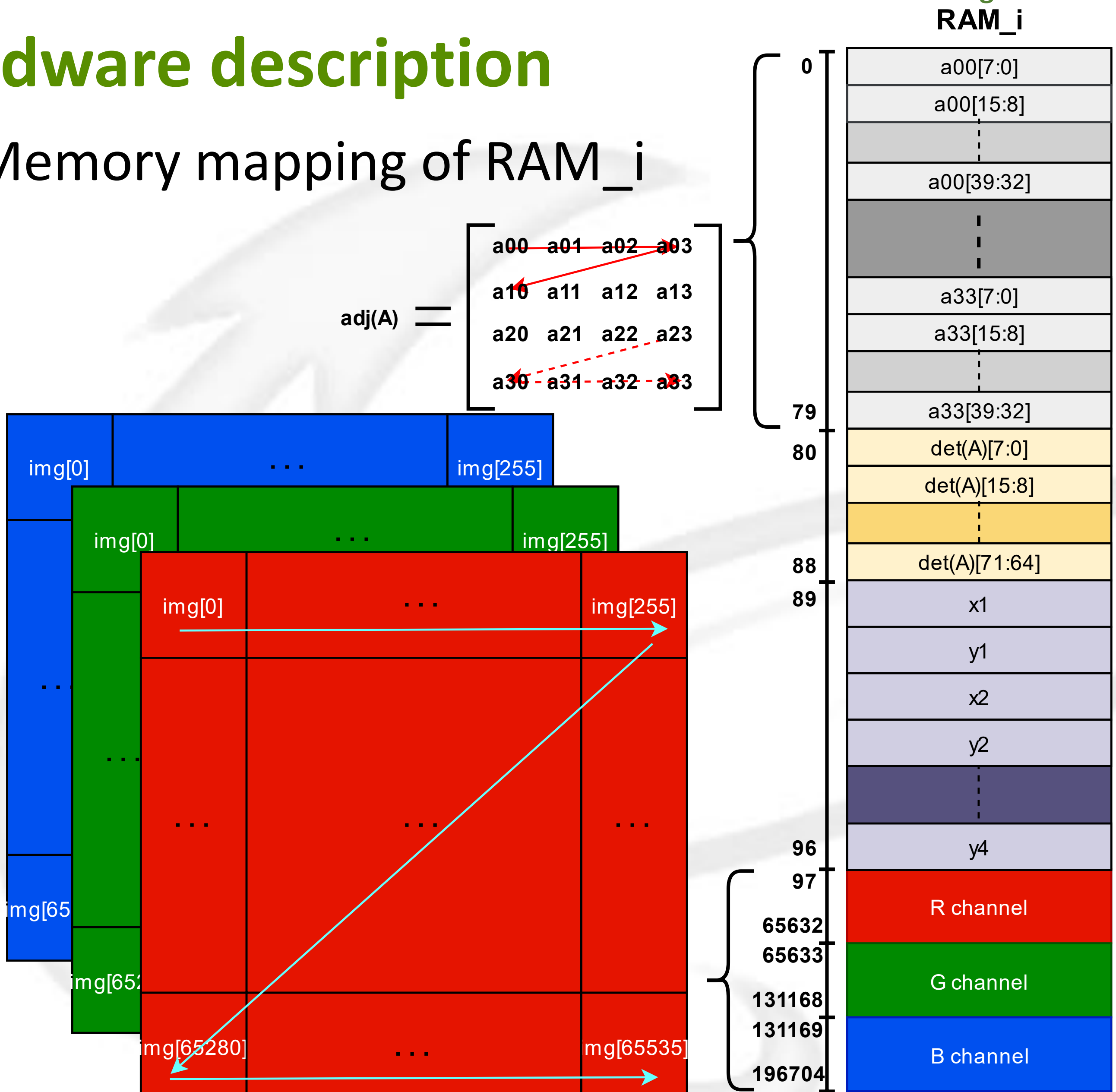
Hardware description

I/O Information

Signal	I/O	width	Desc.
clk	I	1	positive-edged triggered
rst	I	1	asynchronous positive-edged triggered
enable	I	1	enable signal to start processing
*_Q	I	8	8-bit data to be transmitted
*_OE	O	1	Active high read enable signal
*_WE	O	1	Active high write enable signal
*_A	O	18	Address
*_D	O	8	Data
done	O	1	Finish signal

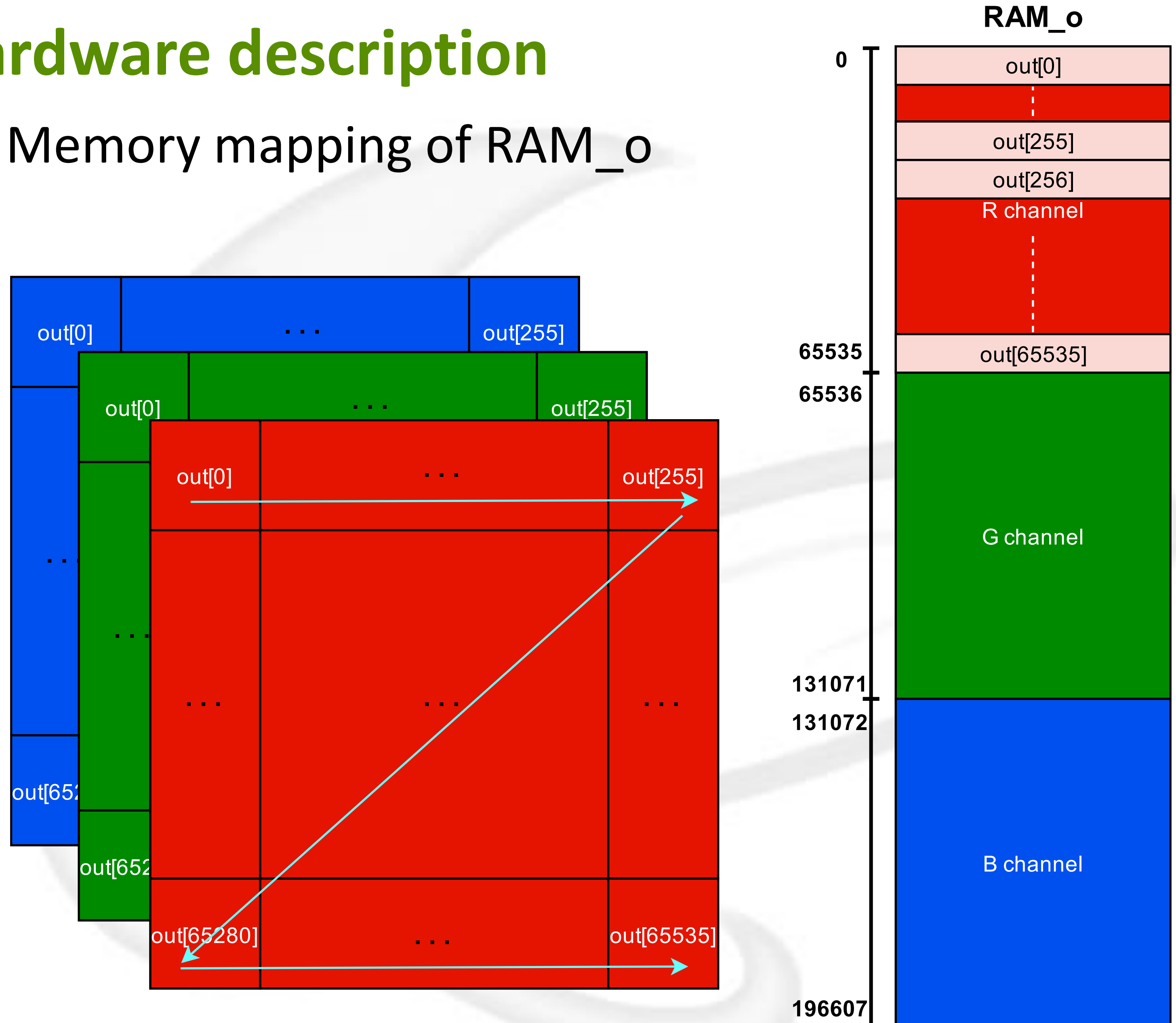
Hardware description

- Memory mapping of RAM_i



Hardware description

- Memory mapping of RAM_o

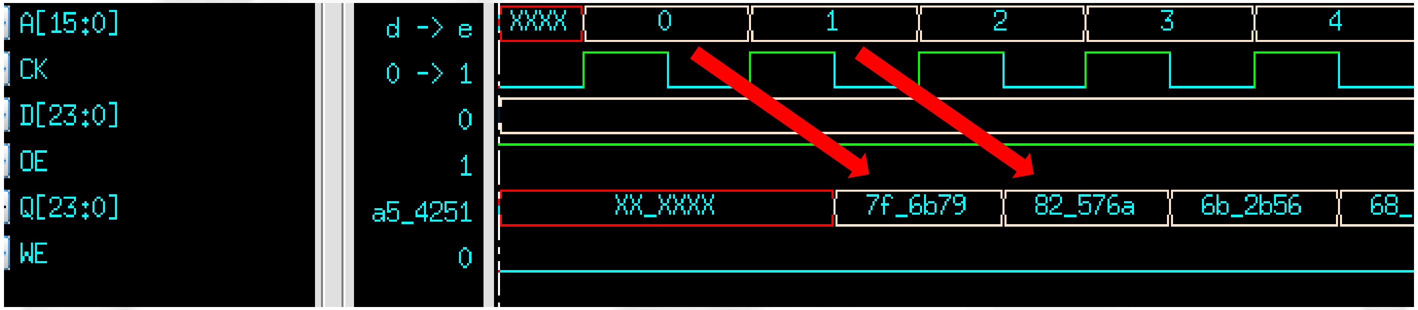


Flow in Lab7(1/2)

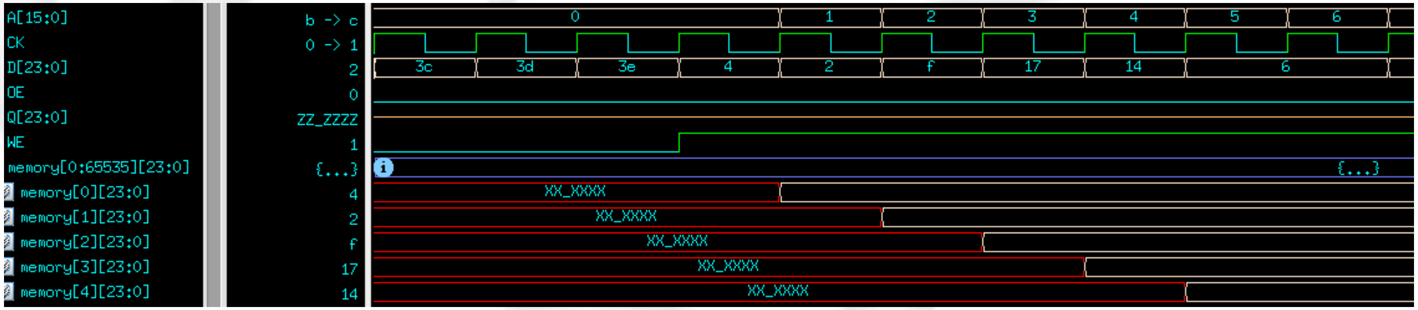
- Step1
  - Compute the inverse matrix of A using adj(A), det(A), and the coordinates (x1, y1) to (x4, y4).
- Step2
  - Use the formula to calculate the pixel addresses of both the destination and source.
- Step3
  - Read data from RAM_i and write it into RAM_o.
  - Repeat steps 1-3 until the entire input image is processed.

Flow in Lab7(2/2)

- The timing information for Read/Write SRAM
  - Read operation(delay one cycle)



  - ✓ The memory will output values on the negative edge, and you need to capture data on the positive edge
  - Write operation



Lab7 Implementation

Data Format

- Adjoint matrix of A (adj(A))
  - Size: 16 (4*4)
  - Signed input data with 2's complement representation.
    - 40-bit signed integer
- Determinant of A (det(A))
  - Size: 1
  - Signed input data with 2's complement representation.
    - 40-bit signed integer + 32bit fraction
- Image pixel 、coordinates (x1, y1) to (x4, y4)
  - Unsigned 8-bit

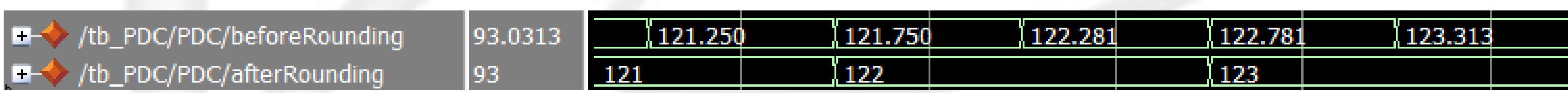
Lab7 Implementation

Rounding

- You need to round the result of the x_i and y_i.
  - Greater than or equal to 5, round up
  - less than 5, round down.

$$\begin{cases} \boxed{x_i} = \boldsymbol{a} \cdot x_i' + \boldsymbol{b} \cdot y_i' + \boldsymbol{c} \cdot x_i' \cdot y_i' + \boldsymbol{d} \\ \boxed{y_i} = \boldsymbol{e} \cdot x_i' + \boldsymbol{f} \cdot y_i' + \boldsymbol{g} \cdot x_i' \cdot y_i' + \boldsymbol{h} \end{cases}$$

- Ex:



Lab7 Implementation

- Fixed point division (Ex:4-bit integer + 3bit fraction)
  - Ex:4.5 / 2.0 = 2.25

    $= 0100\_100_{base2} / 0010\_000_{base2}$

    $= 36_{base10} / 16_{base10}$

    $= 2_{base10} = 0000\_010_{base2}$ $\mathbf{!= 0010\_010_{base2}}$

  - The dividend must be left-shifted by the fraction bit before performing the division.

    $= (0100\_100_{base2} << 3) / 0010\_000_{base2}$

    $= 288_{base10} / 16_{base10}$

    $= 18_{base10} = 0010\_010_{base2}$ $\mathbf{== 0010\_010_{base2}}$

Criteria

- Simulation result
  - Pass

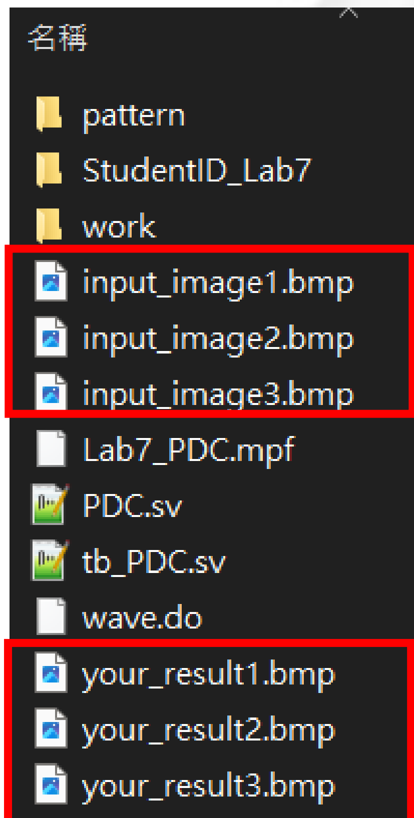
```
VSIM 5> run -all
# ********************************
# **      Simulation Start      **
# ********************************
# ============ Pattern 1 PASS !!! ============
# ============ Pattern 2 PASS !!! ============
# ============ Pattern 3 PASS !!! ============
#
#
#  ***************************
#  **                       **       |__||
#  **  Congratulations !!   **      / O.O  |
#  **                       **    /_____   |
#  **  Simulation PASS!!    **   /^ ^ ^ \  |
#  **                       **  |^ ^ ^ ^ |w|
#  ***************************   \m___m__|_|
#
#
# ** Note: $finish    : E:/HDL_course_prepare/Lab7_PDC/tb_PDC.sv(166)
#    Time: 9834035 ns  Iteration: 1  Instance: /tb_PDC
# 1
# Break in Module tb_PDC at E:/HDL_course_prepare/Lab7_PDC/tb_PDC.sv
```

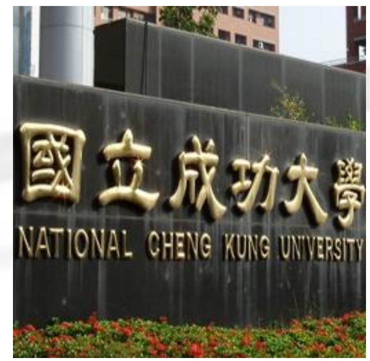
  - Failed

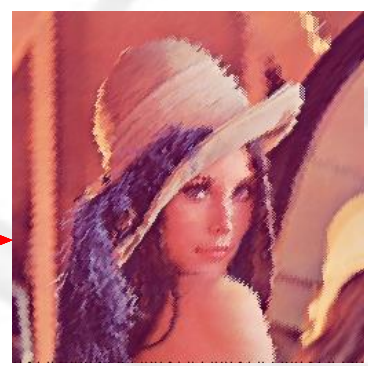
```
VSIM 9> run -all
# ********************************
# **      Simulation Start      **
# ********************************
#  ***************************
#  **                       **       |__||
#  **  OOPS!!               **      / X,X  |
#  **                       **    /_____   |
#  **  Simulation Failed!!  **   /^ ^ ^ \  |
#  **                       **  |^ ^ ^ ^ |w|
#  ***************************   \m___m__|_|
# Error, Pattern 1, RAM_o[    1] = 224, expect = 226
#  -------------- Simulation stop --------------
```

Criteria

- Simulation result – Visualization
  - ➔ It will generate the input picture and your output result in a BMP file when your simulation is finished.







Criteria

- Grading policy(100%)
  - Lab7
    - Simulation pass (90%)
    - Report (10%)

Lab7 Requirement & file format

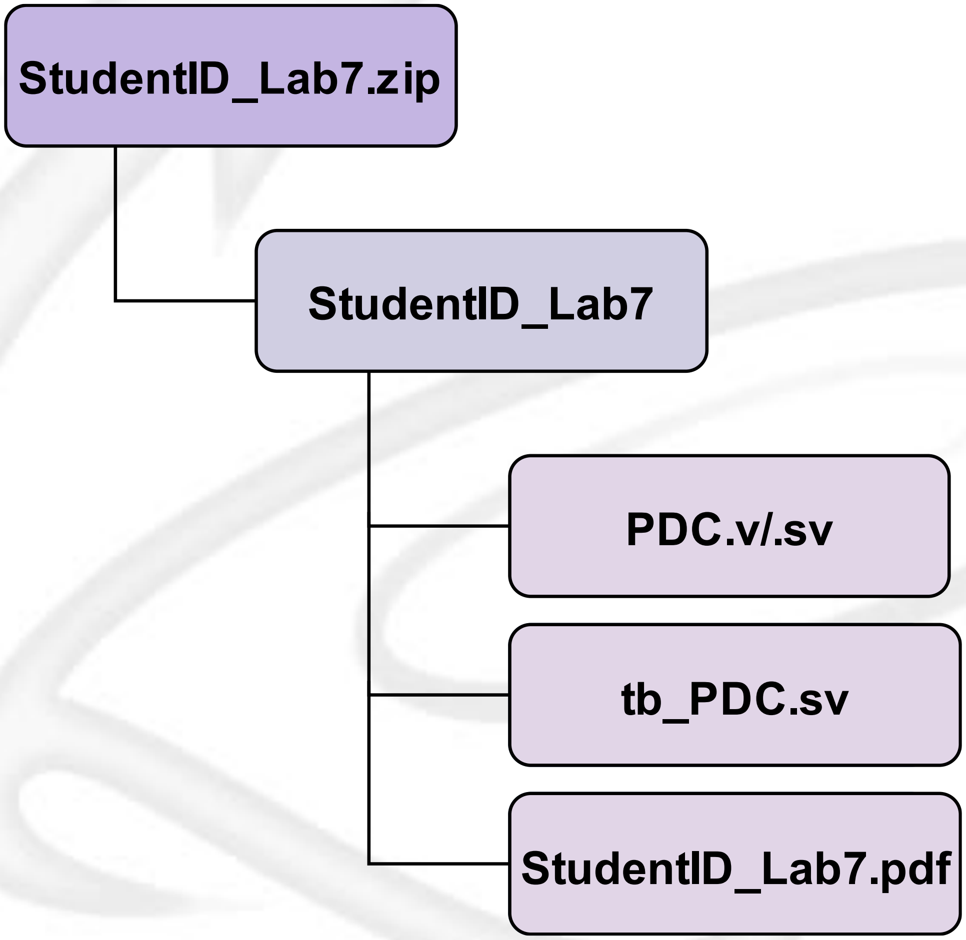
- You must finish PDC.v/.sv and pass all patterns
- For Lab7, you need to submit
  - PDC.v / PDC.sv
  - tb_PDC.sv
  - StudentID_Lab7.pdf
- Deadline:2024/04/18 08:59 a.m. (No late submission)

Lab7 Requirement & file format

- Friendly reminder
  - Please complete the assignment by your own, discussion with peers is recommended, but do not cheat.
  - Warning! Any dishonesty found will result in zero grade.
  - Warning! Any late submission will also receive zero.
  - Warning! Please make sure that your code can be compiled in Modelsim, any dead body that we cannot compile will also receive zero.
  - Warning! Please submit your work according to the specified file format, making sure not to include any unnecessary files. Any unnecessary file found, will lead to 10% deduction from the overall score.

Lab7 Requirement & file format

- File format



Thanks for listening