Homework 4 –Sorbel Edge Operation with Shift-Register-Based Line Buffer

Handout: 2024/11/12

Due: 3 weeks later

1. (Color Space Transformation)

   The following shows the color conversion of an image pixel from RGB to YC_bC_r (YUV)

$$\begin{bmatrix} Y \\ C_b \\ C_r \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.169 & -0.331 & 0.500 \\ 0.500 & -0.419 & -0.081 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

   where Y is the corresponding intensity of the pixel. Three constant-multiplications and accumulation of three products are required to compute the intensity of an image pixel. For ease of implementation, the multiplications of constants can be approximated by shift-add/sub operations because $0.255 \sim= 2^{-2} + 2^{-5}$; $0.587 \sim= 2^{-1} + 2^{-4}$; $0.114 \sim= 2^{-3} - 2^{-6}$. In general, we can approximate a constant with binary signed digits of 0, 1, or -1 using canonical recoding (minimum recoding) with minimum number of non-zero digits. And thus multiplication of a constant can be realized using only shift-add/sub operations.

2. (Convolution with Sorbel Filters)

   After generating the intensity value (Y) for each pixel, use the following Sorbel convolution masks to find the vertical edges and horizontal edges from the intensity image:

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}, \quad G_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}.$$

   The edge detection algorithm is as follow:

   (1) First, pad the intensity map Y with a boundary of zeros so that the resolution of the output feature maps after the 3x3 filter convolution remain unchanged.

   (2) Center one convolution mask over successive pixels in the original intensity image.

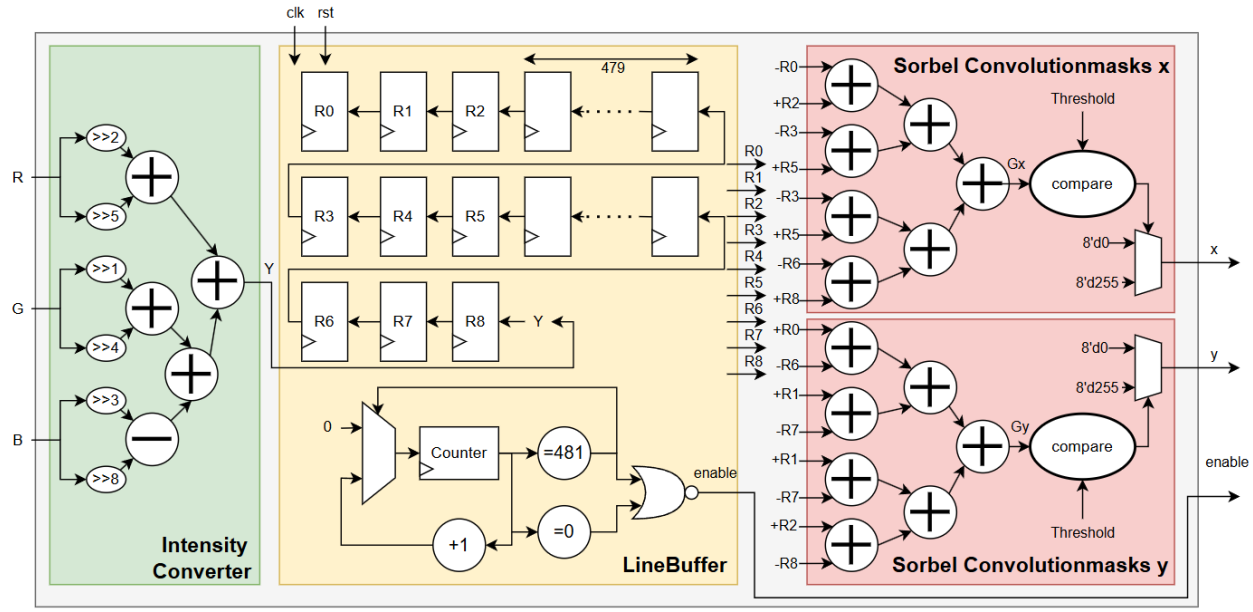
   (3) Multiply the coefficient in each mask by the intensity value of the underlying pixel and sum the 9 products together. Convolution with the 3x3 Sorbel filters generates feature maps of gray-level.

   (4) Select a threshold value Tsh by yourself (About 100~150) to determine whether the pixel belongs to an edge by comparing the value calculated in (2) with the threshold value Tsh. If the convolution results is larger than Tsh, the pixel of the generated edge map is set to be 255; otherwise, the edge map pixel value is set to zero. After the above operations, two edge maps of binary images are generated, one for horizontal edge, the other for vertical edges.

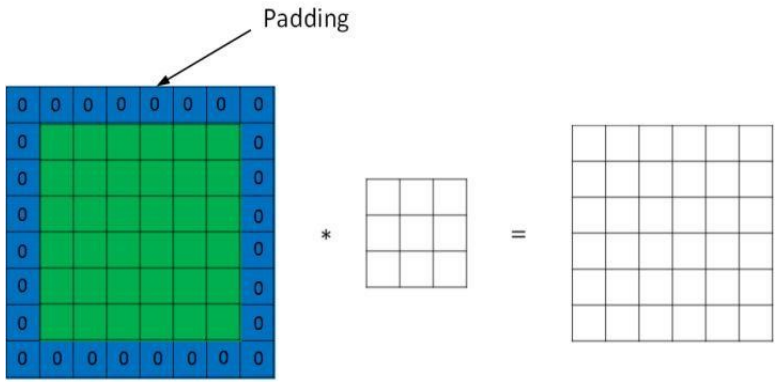
   (5) Calculate the total number of execution cycles required to generate the two edge maps.

3. (Implementation)

The following figure shows the overall architecture of the hardware for Sorbel edge detection. It includes color-to-intensity converter, line buffers, and two convolution units (one with Gx and the other with Gy). Note that row 1 and row 2 of the line buffer have 482 registers each, while row 3 has only 3 registers. Write Verilog codes to implement the hardware. The input memory (storing input RGB color images) and the output memory (storing the computed edge maps) are located in the testbench.



(1) Pad zeros around the boundary of the original images (W*H=480*360) into W*H=482*362 resized image so that the size of the final output edge map remains W*H=480*360 after 3*3 Sorbel convolution. The padding could be done beforehand in the testbench.



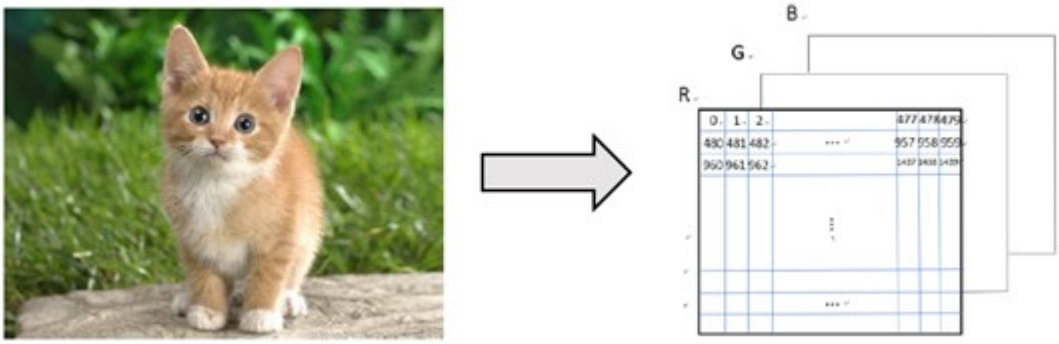
(2) First, fetch the RGB color images from the input memory in the testbench one pixel per cycle, and generate the corresponding intensity values which are stored in a line buffer (constructed with shift registers).

(3) Then, the edge convolution hardware compute the edge map pixels (one pixel per cycle) which are sent to the output memory unit the in testbench.

Appendix:

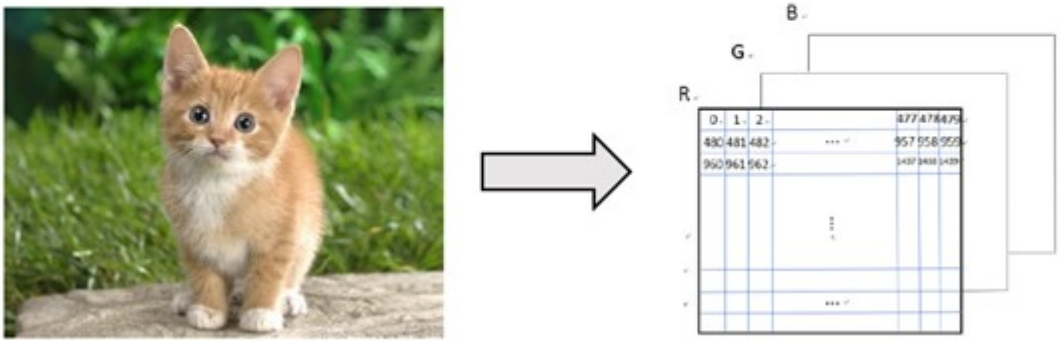
Explanation on Line Buffer Design:

1. As shown below, assume the resolution of the file “cat.bmp” of a colored image is 480*360, which can be split into three input channels: R, G, B. You can obtain the three input channel in the testbench using non-synthesizable statements.

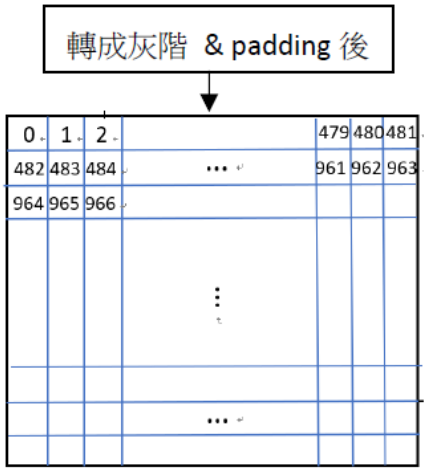


1. **(假設輸入圖片為: "cat.bmp"(480*360))**(下圖為取 RGB 示意圖)

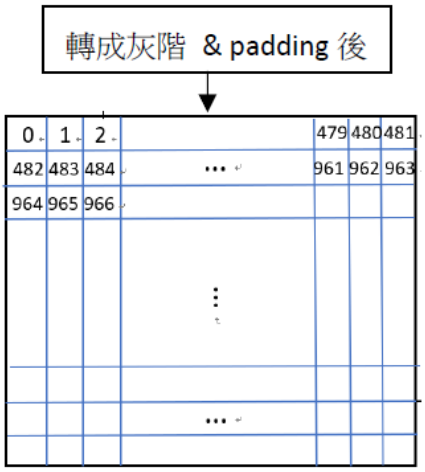
   →取 RGB 這動作是在 testbench 裡面處理。



2. Convert the three RGB input channels into gray-level images. Then, use padding to change the images into resolution of 482*362 by adding zeros along the boundary of the images. Thus, after convolution by 3*3 filters, the resolution of the outputs is still 480*360. You can perform the padding in the testbench, not in the hardware.



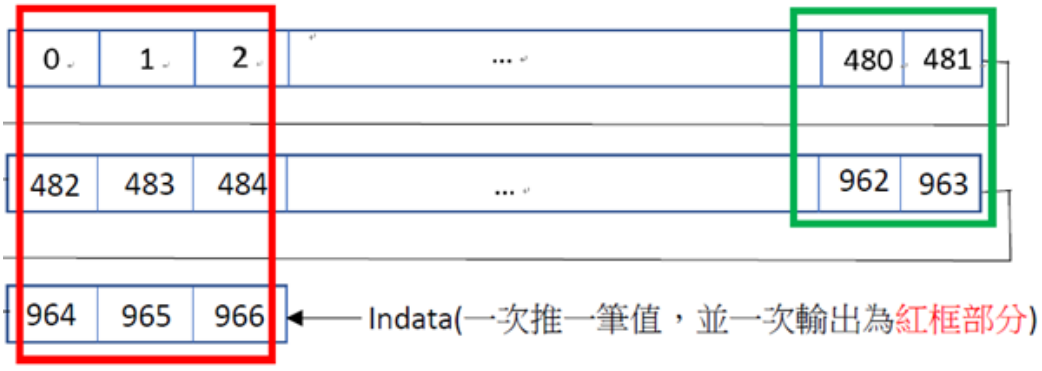
2. 取出 RGB 的 data 後，先轉換成灰階圖片，並且轉換出來的灰階圖片需做 padding 的動作，所以長寬將變為 482*362。(轉換成灰階 & padding 這個動作皆是在 testbench 做)



3. Use synthesizable statement to design a line buffer which stores three lines of pixels as shown below. Perform convolution using the Sorbel edge detectors

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}, \quad G_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

The outputs are the vertical and horizontal edges of the inputs after the Sobel convolution Gx and Gy. Note that the red block shows the pixels for Sobel convolution. You need to skip the pixels in the last two columns in order to generate outputs of the same resolution.



※〈注意〉: 綠框部分是會需要被跳過的

2..**Linebuffer**(再經由 linebuffer 取出的 9 個 pixel 分別去對 Gx, Gy 去做相乘(此處就像是 Convolution 的概念)，linebuffer 內部值即為做完 padding 後的灰階圖片之 data。)

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}, \quad G_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}.$$

乘加完後所得知最後圖像即分別為"vertical edges" and "horizontal edges" Figure

※〈注意〉：綠框部分是會需要被跳過的

Report Requirement

1. Design Complier(70%)

submitted files should include :

I. Verilog RTL code **(10%)**

II. Verilog RTL testbench**(10%)**

III. gate-level code(including sdf) & testbench**(10%)**

report should include (PDF 報告類須含有):

I. figure of overall architecture (架構圖) **(10%)**

II. both RTL and gate-level simulation waveforms, including explanations (RTL 波形 & gate-level 波形並解釋) **(10%)**

III. area information and critical path delay (Area 資訊和 critical path 資訊) **(5%)**

IV. original input image (of cat), image of horizontal edges and vertical edges (原貓咪圖**(0%)**、水平邊緣圖片**(5%)**、垂直邊緣圖片) **(5%)**

V. Comments (心得) **(5%)**

2. Xilinx Vivado(30%)

Name the project as HDL_HW4_MXXXXXXXXX

submitted files should include (檔案類須含有):

I. HDL HW4_MXXXXXXXXX.xpr.zip**(10%)**

II. xdc、wcfg**(5%)**

report should include (PDF 報告類須含有):

I. Simulation waveforms of both behavior level and post-implementation, including explanations (Behavior 波形 & post-implement 波形並解釋) **(10%)**

II. Snapshots of project summary-overview (Project Summary-Overview 截圖) **(5%)**

Compress all the above files into a single file and name it as HDL_HW4_MXXXXXXXXX.7z (以上打包成 MXXXXXXXXX.zip 壓縮檔並繳交)