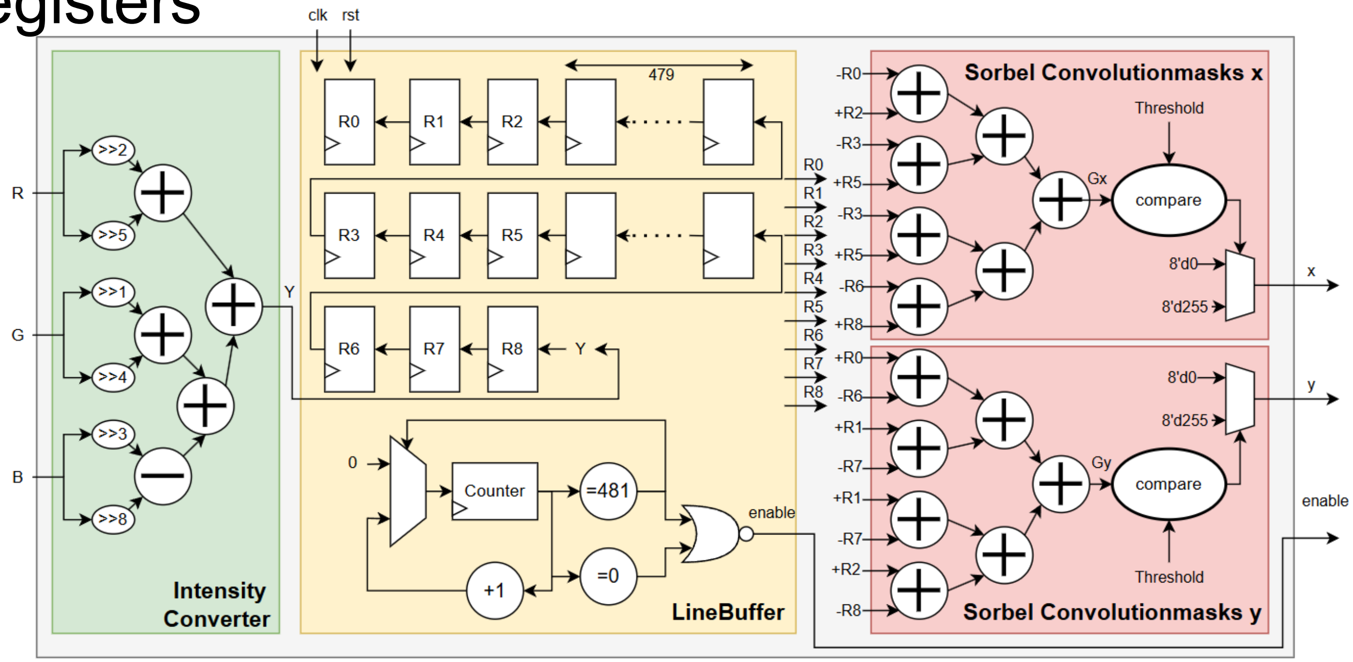
HDL HW4:
Sobel Edge Detector

Outlines

- Color-to-intensity image converter
- Sobel edge convolution unit
  - horizontal edges
  - vertical edges
- Padding in software
- Line buffers
  - shift registers



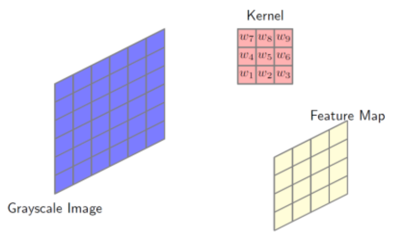
Color Space Transform

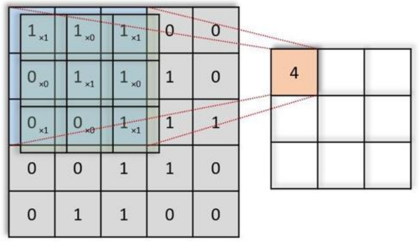
- transform of image color space
  - RGB -> YUV (or YCbCr)
  - Y is the gray-level intensity map of the image
    - ✓image edges could be extracted from Y
- multiplications of constants could be approximated by shift-add/sub
  - e.g., 0.299 ~= $2^{-2} + 2^{-5}$
  - e.g., 0.587 ~= $2^{-1} + 2^{-4}$
  - e.g., 0.114 ~= $2^{-3} – 2^{-6}$

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

2D Convolution

https://miro.medium.com/max/875/1*wpbLgTW_lopZ6JtDqVByuA.gif

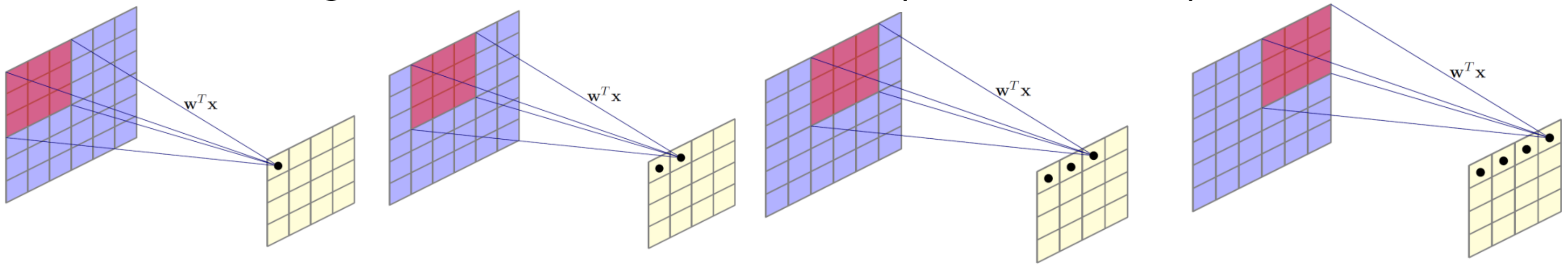




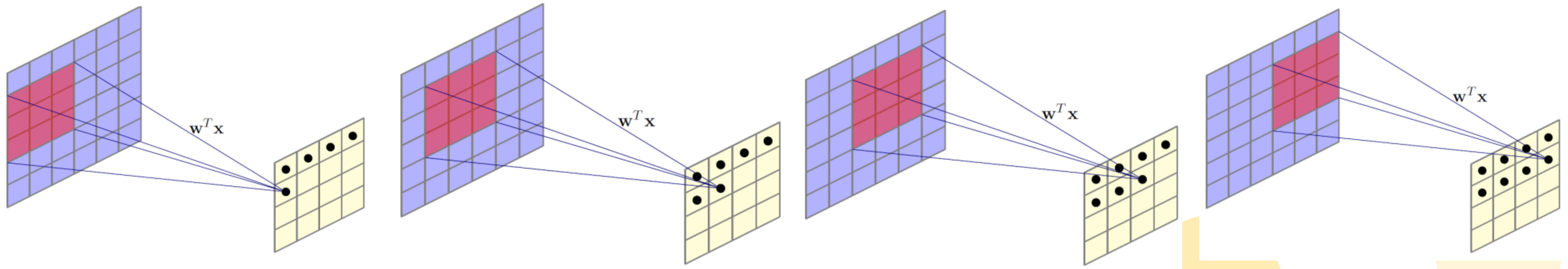
1 ×1	1 ×0	1 ×1	0	0
0 ×0	1 ×1	1 ×0	1	0
0 ×1	0 ×0	1 ×1	1	1
0	0	1	1	0
0	1	1	0	0

4		

◇ convolution to generate the 1st row of an output feature map



◇ convolution to generate the 2nd row of an output feature map



…

Convolution with Sobel Edge Operators

- detect horizontal and vertical edges
  - large intensity difference at edges => large subtract results
  - small intensity difference at non-edges

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

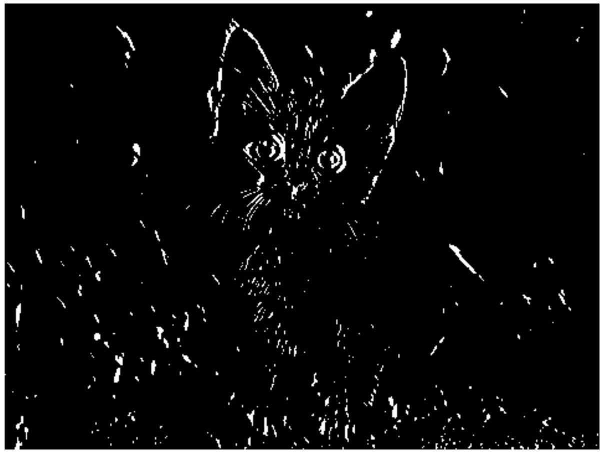
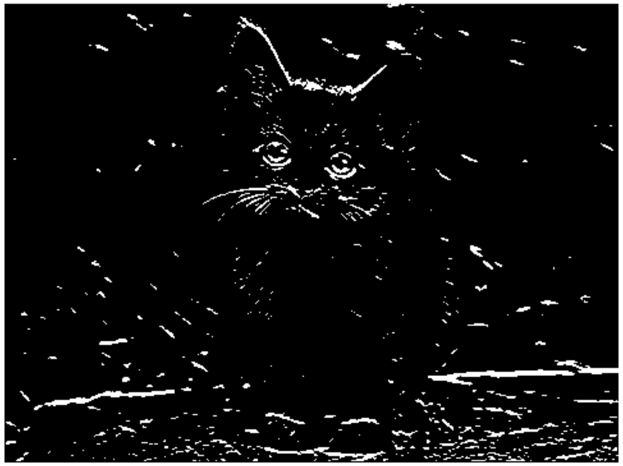
- select a fixed threshold T_{sh} to convert convolution results into binary images (with of 0 or 255)
- steps
  - pad boundary with zeros
  - convolution with Sobel operators
  - comparison with T_{sh}
  - replacing the central pixel with either 0 or 255

Convolution with Sobel Edge Operators

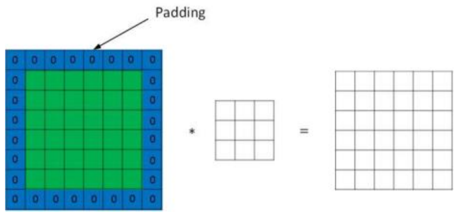
- detect horizontal and vertical edges

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

- select a fixed threshold T_{sh} to convert convolution results into binary images (with of 0 or 255)

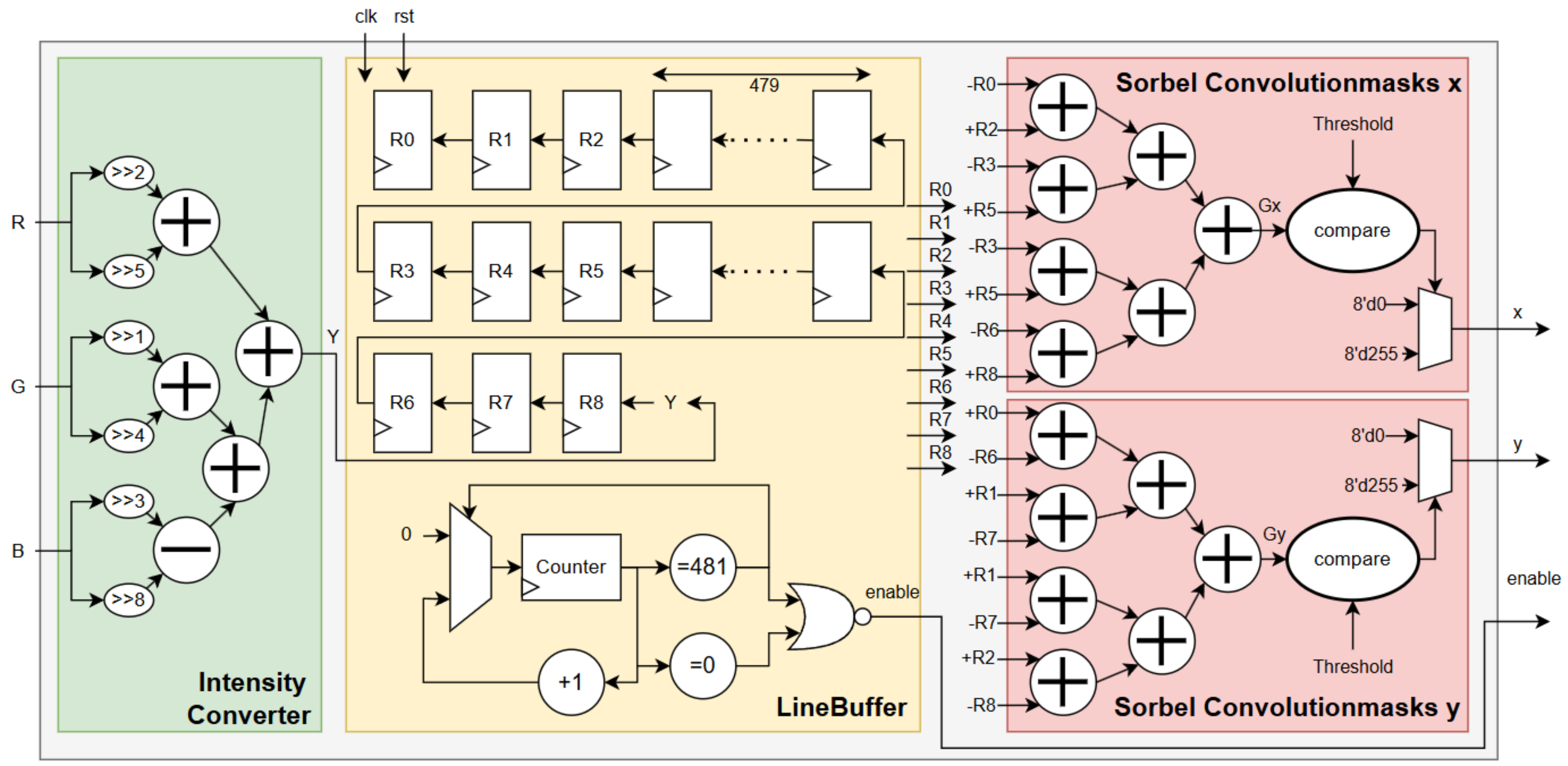
Implementation



- pad original image (W*H=480*360) into 482*362
  - add zeros around the boundary of the original image
- fetch pixels from testbench
  - no need to create hardware memory to store image pixels
- design hardware line buffer (shift registers)
  - store the converted intensity pixels
  - provide data for Sobel convolution (both G_x, and G_y)
  - realize scan of pixels from left to right, and from top to down
  - send results back to testbench
  - calculate toal numb er of execution cycles required to generate the two edge maps
- synthesize into gate0level netlists

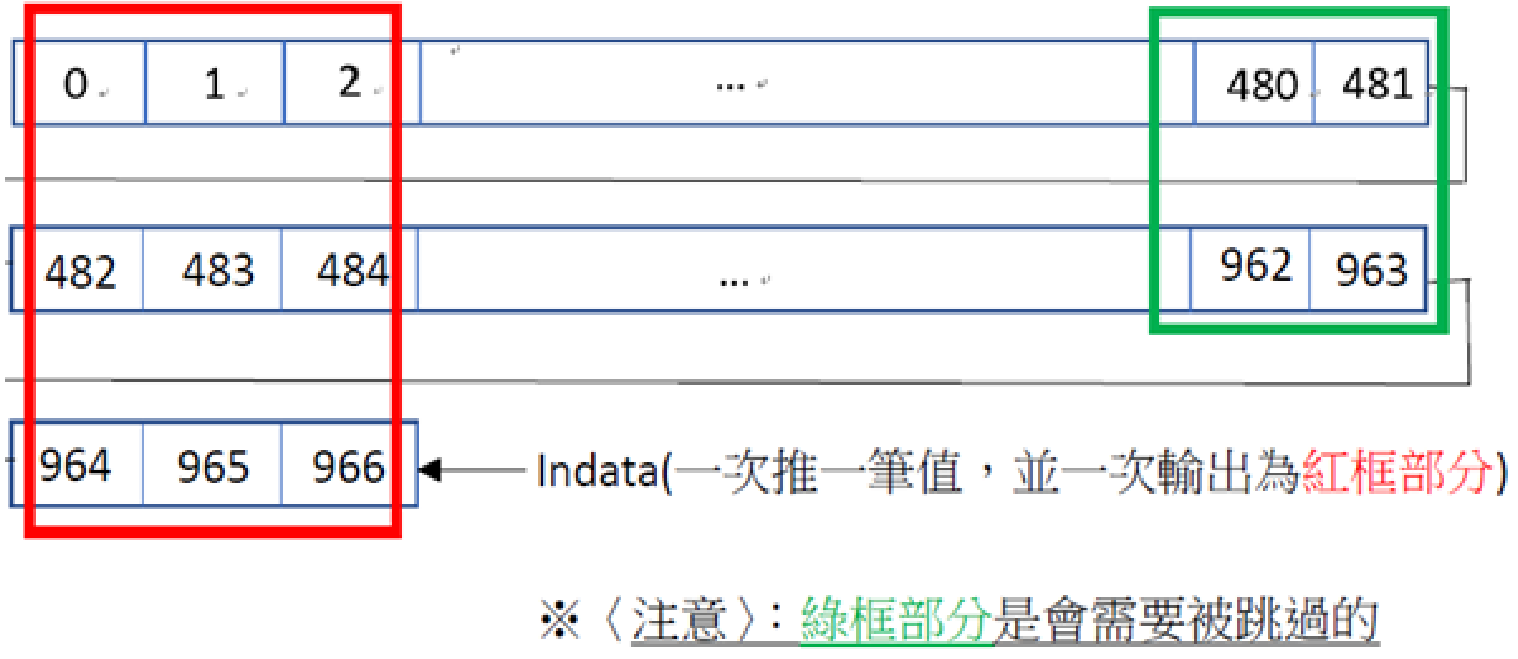
Hardware Architecture

- overall architecture
  - one intensity converter
  - line buffer
  - two convolution units, one with Gx, the other with Gy



Line Buffer and Convolution

- data movement to generate 1st row of edge map
  - first convolution for 3x3 intensity block centerd at 483
  - last convolution for 3x3 intensity block centered at 962
  - generate 1st output row (480 edge map pixels)
  - concurrently generate two 1st output rows (with G_x and G_y)
- similarly for generating other rows of edge maps



0
1
2
...
480
481
482
483
484
...
962
963
964
965
966
Indata(一次推一筆值，並一次輸出為紅框部分)
※〈注意〉：綠框部分是會需要被跳過的

Two Implementations of Shift Registers

```
x[0] <= d
x[1] <= x[0]
x[2] <= x[1]
…
x[N-2] <= x[N-3]
x[N-1] <= x[N-2]
out <= x[N-1]
```

```
// a total of N shift registers, each bw bits
reg [bw-1:0] x[0:N-1];
reg [bw-1:0] d, out;
always @ (posedge clk) begin
   x[0] <= d;
   for (i=1; i<=N-1; i=i+1)
      x[i] <= x[i-1];
   out <= x[N-1];  // extra output register out
end
```

```
// a total of N shift registers
reg [bw-1:0] x[0:N-1];
reg [bw-1:0] d, out;
always @ (posedge clk) begin
   {x[0:N-1], out} <= {d, x[0:N-1]};
end
```