Image Filtering in VHDL

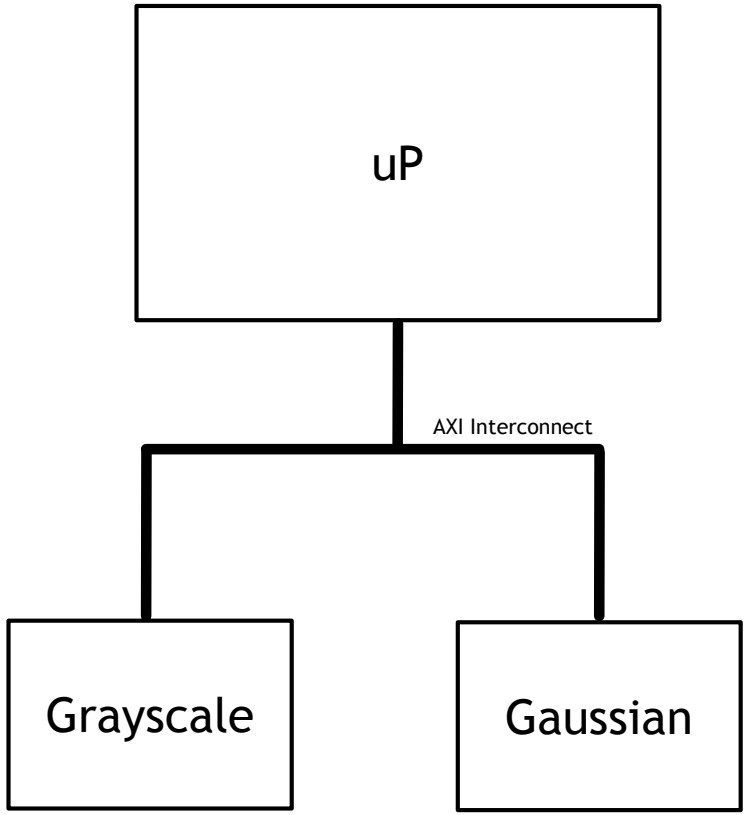
By: Austin Copeman

Azam Tayyebi

Overview

- Filters and Implementation
  - Filter Overview
  - Gaussian
  - Grayscale
- Software
- Results
  - Gaussian
  - Grayscale
- Questions

Filter Overview



Gaussian Filter Statement

- read image data (100x100x8 bits) from PS part of zynq and save it into memory.
- Perform the convolution of the image with a 3x3 kernel.
- Return the result to the convolution into PS part and also display with VGA.

Gaussian

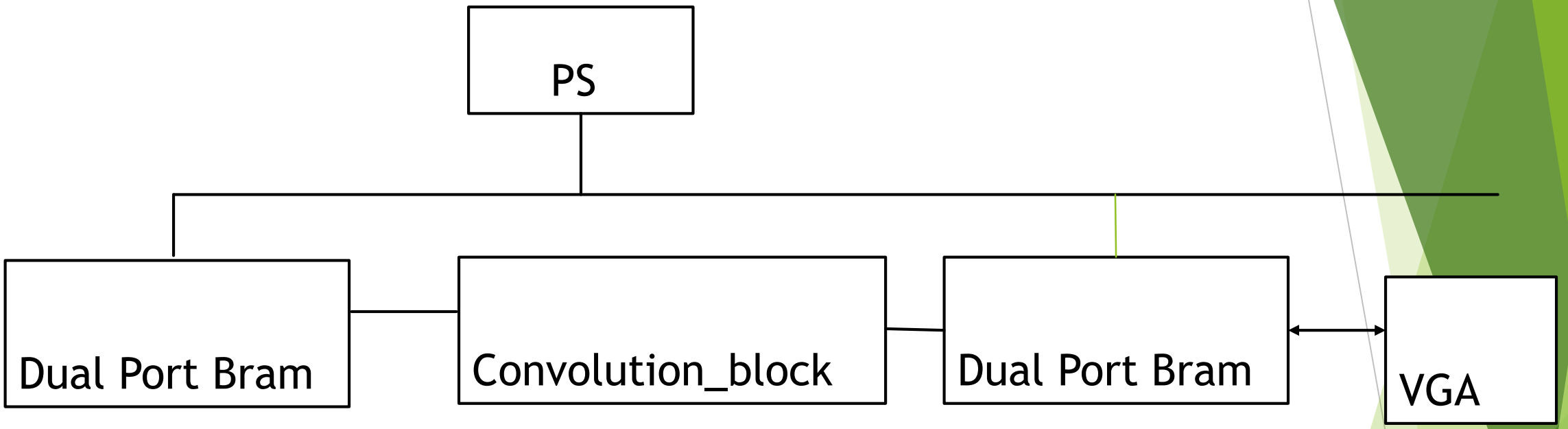
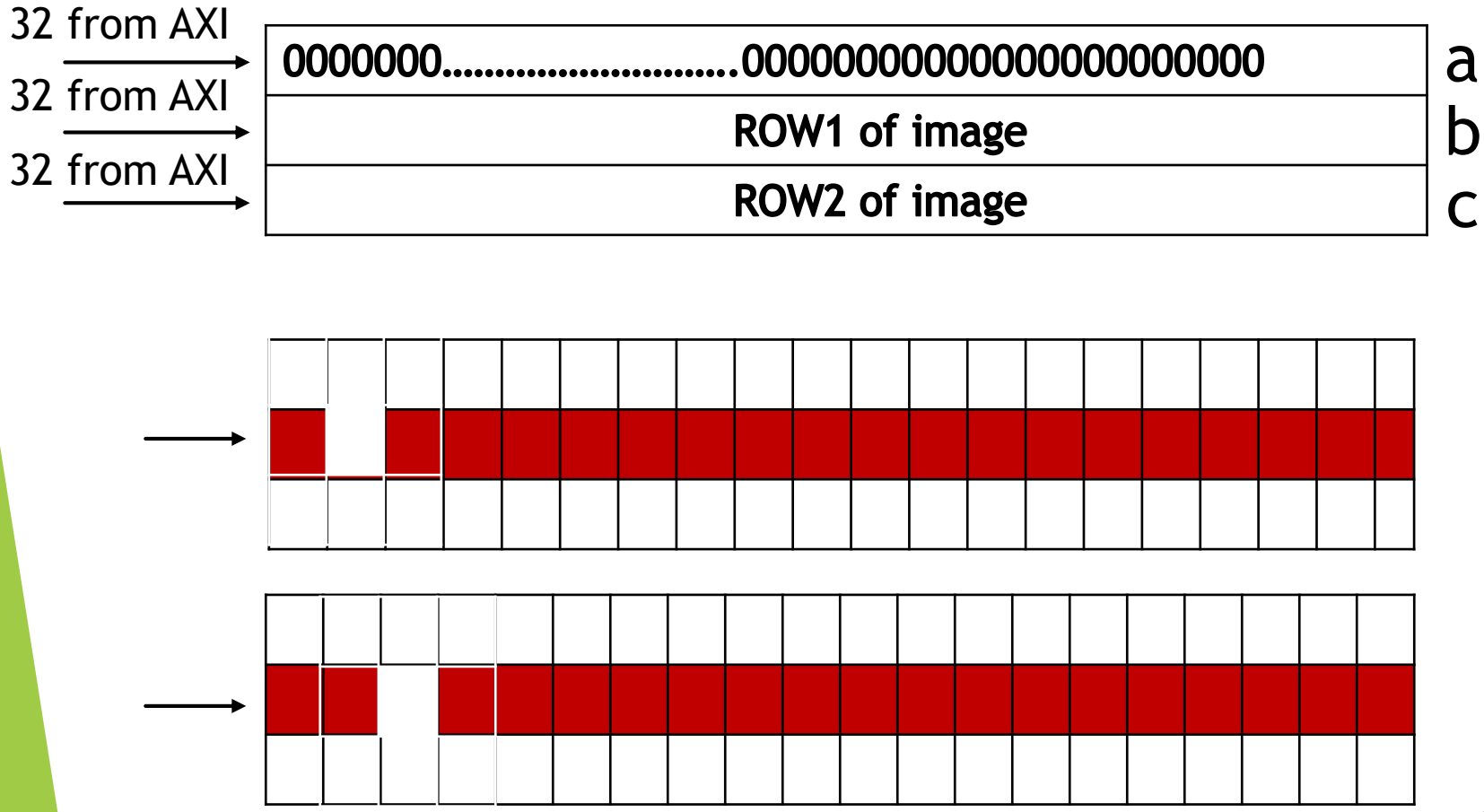
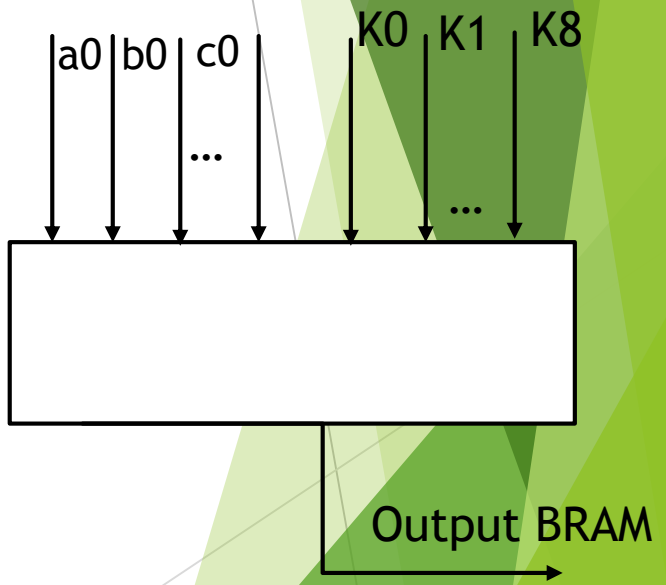


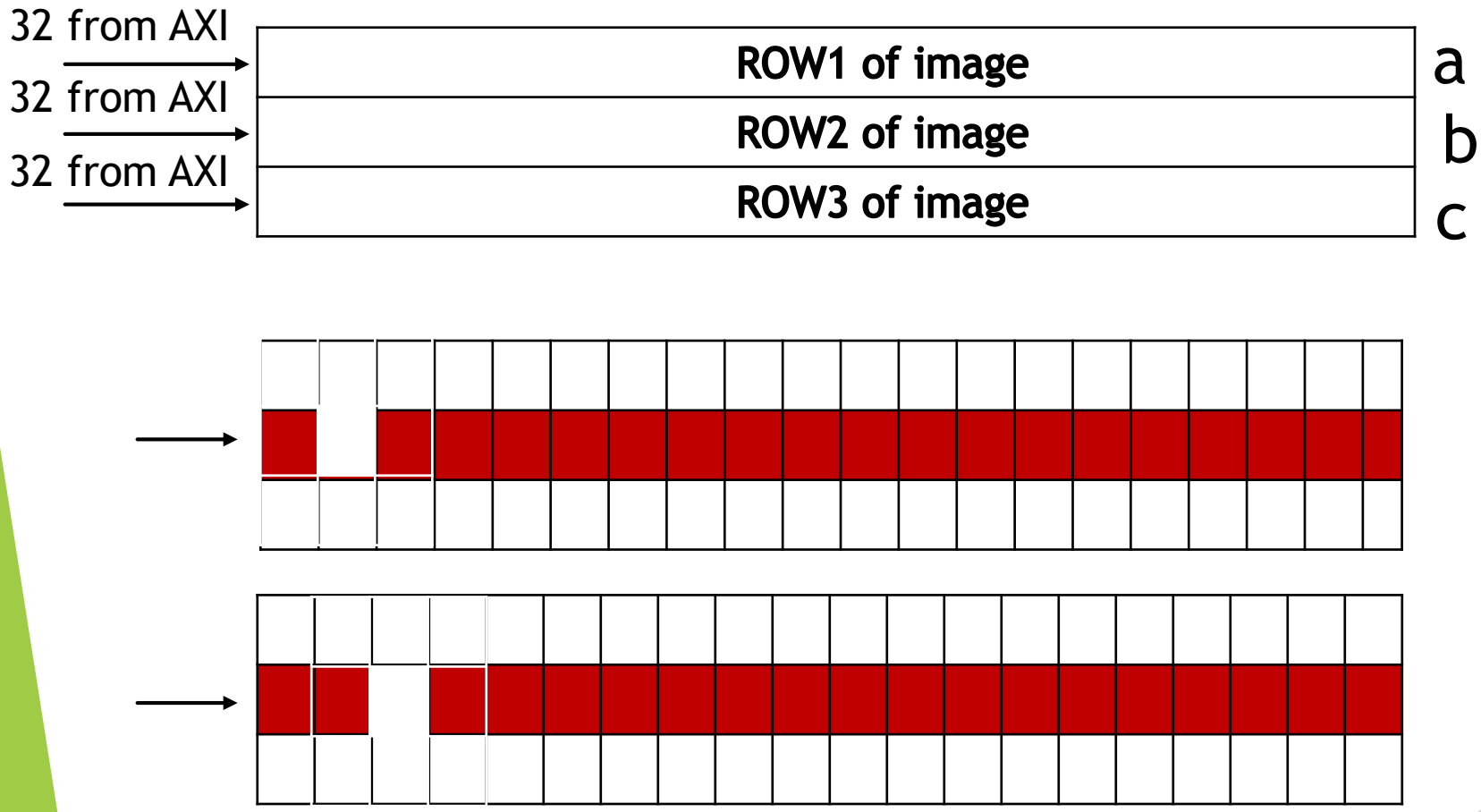
Image is sent by AXI peripheral and stored in the BRAM.
100*102*8 bit = 10200bytes
In each step a portion of the image in the BRAM is sent to the convolution block.
The filtered result is saved in BRAM 100*100*16bit=20000 byte and read by PS part.
The previous process is repeated for the whole image.

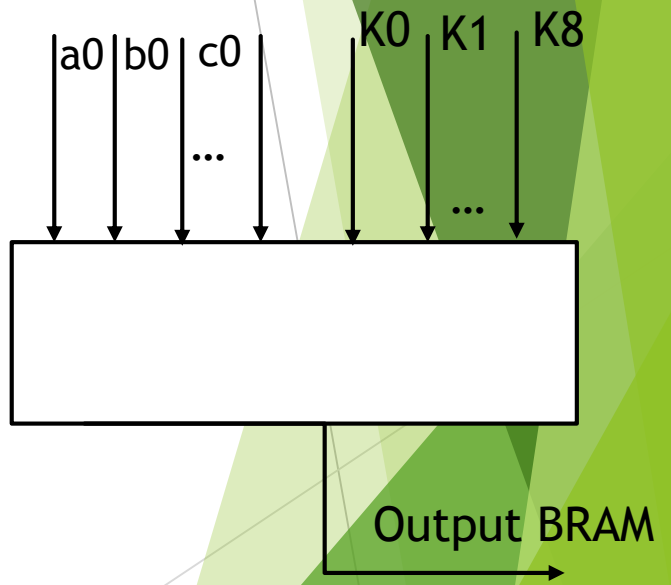
Gaussian - Operation



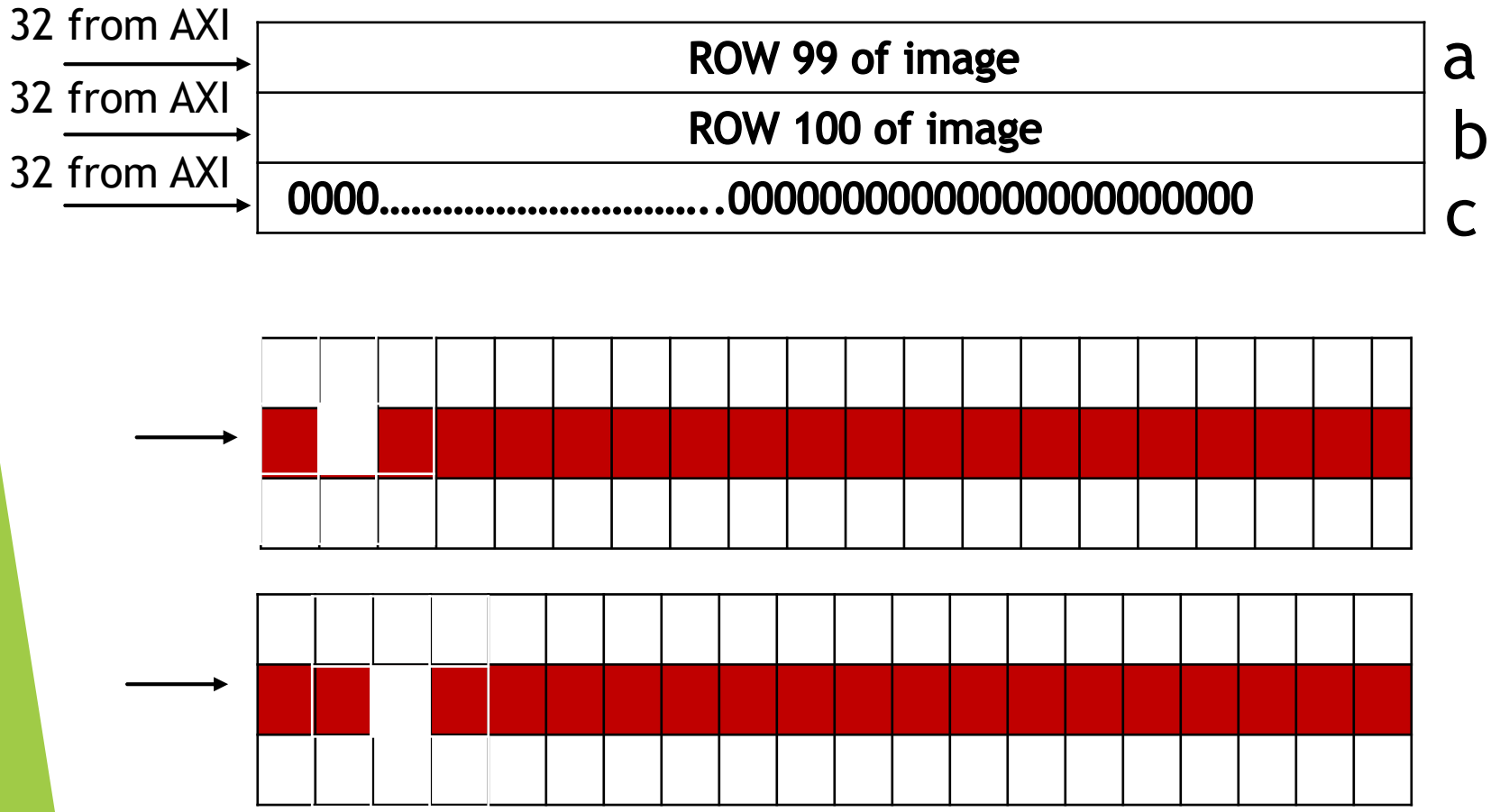


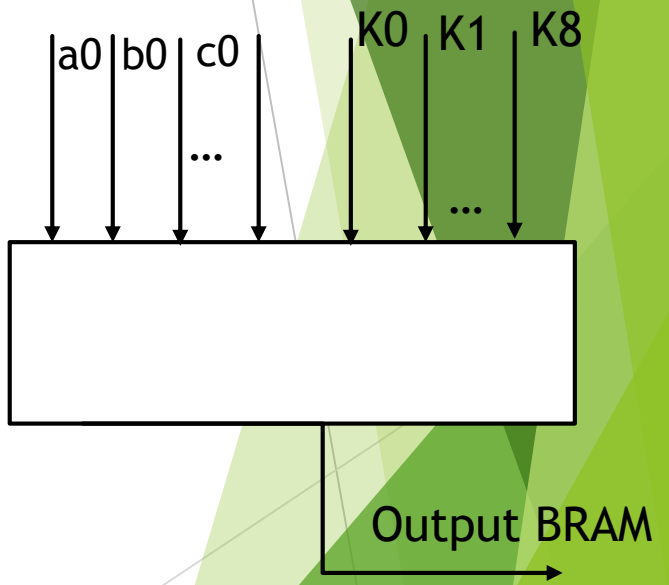
Gaussian - Operation

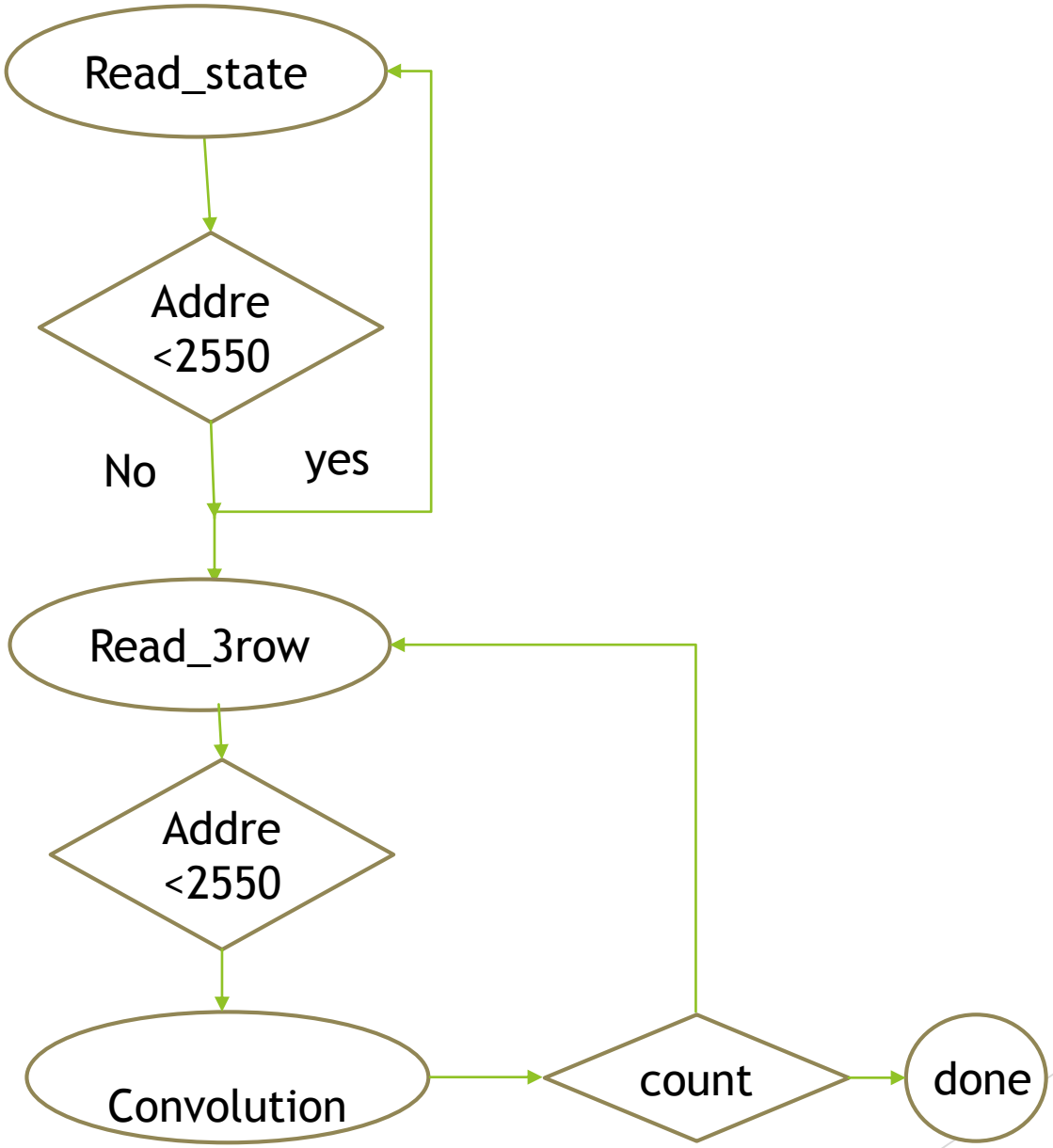




Gaussian - Operation

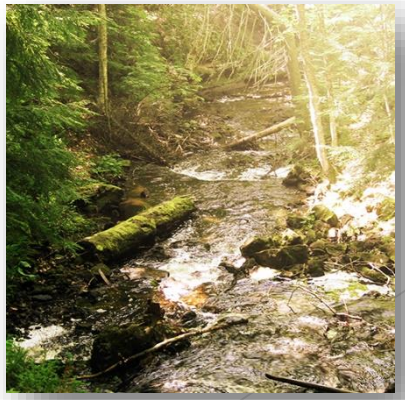





Read_state
Addre
<2550
No
yes
Read_3row
Addre
<2550
Convolution
count
done

Grayscale - Intro

- What is Grayscale?
  - A range of gray shades from white to black
- How is color represented?
  - Each color(Red, Green, Blue) is represented on a gray scale
  - The gray scale range: [0,255]
    - Where 0 is black and 255 is white
- Conversion from RGB to Grayscale
  - “Percentage” Method
    - Grayscale = R * R% + G * G% + B * B%; where R% + G% + B% = 100%
    - Example
      - Luminosity Grayscale = R * 0.2126 + G * 0.7152 + B * 0.0722

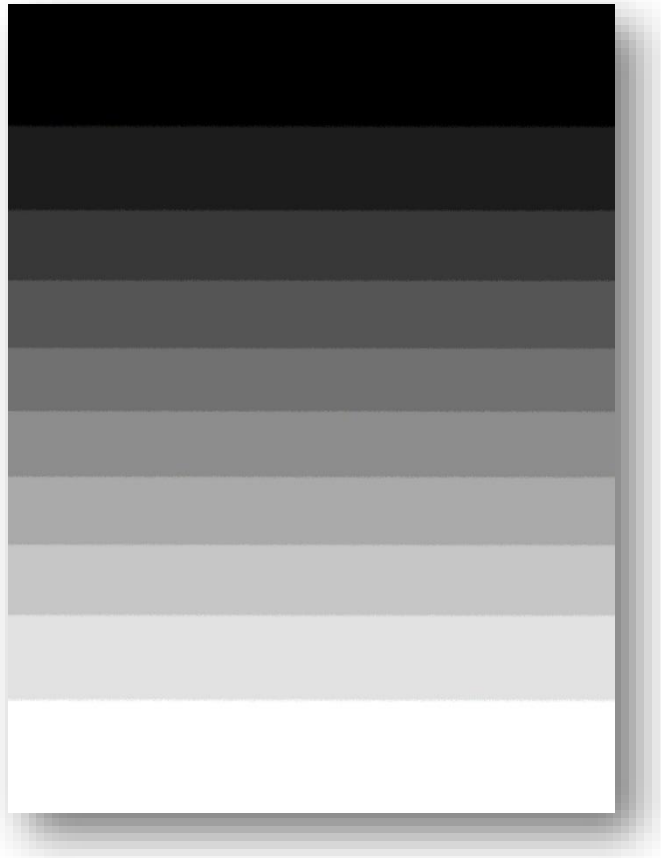


Original Image

Grayscale

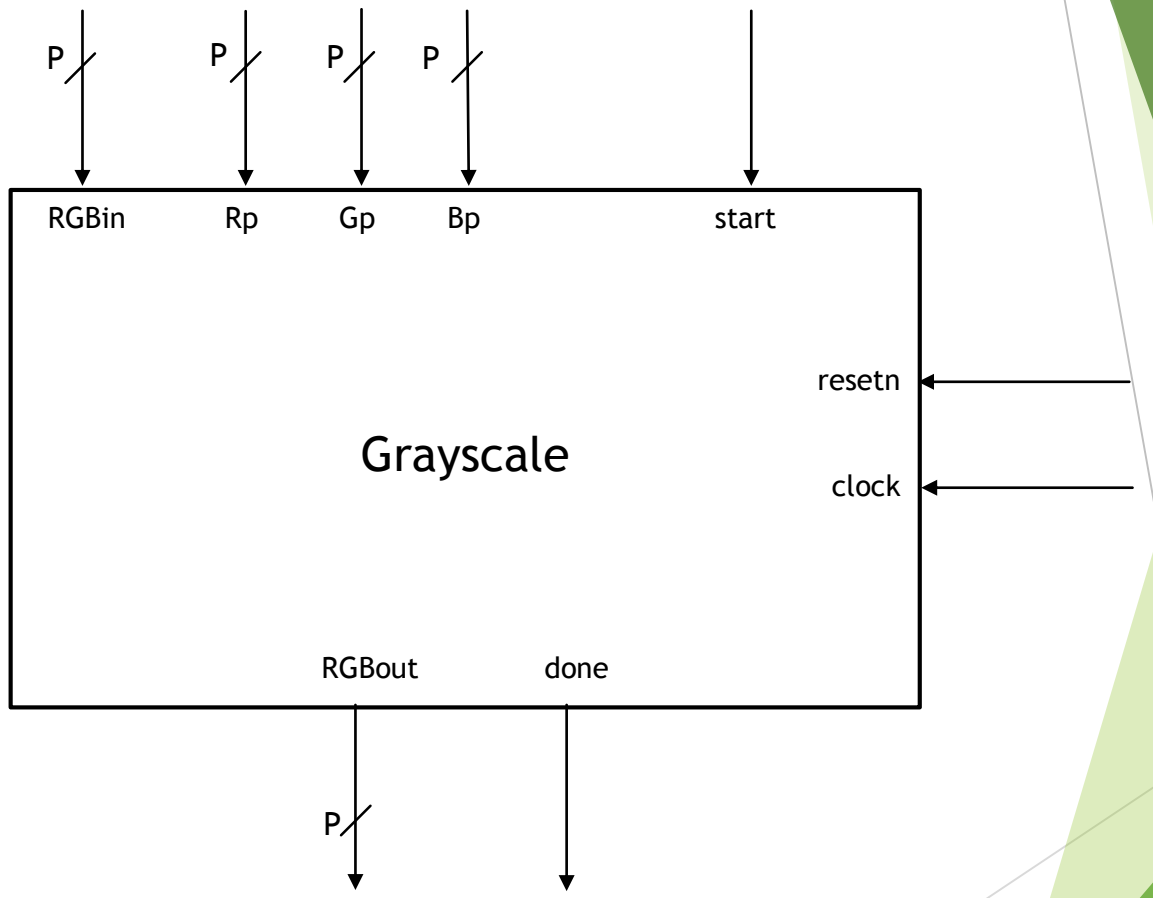
Grayscale - Intro

- Why grayscale?
  - Signal to noise
    - Many applications don’t require color
    - These cases color is considered a “noise”
  - Simplifies code
    - Simplifies finding an image’s edge
  - Less complex
    - Color is complex
    - Color has 3 channels
    - Grayscale has 1 channel
  - Speed
    - Color requires 3 channels to process



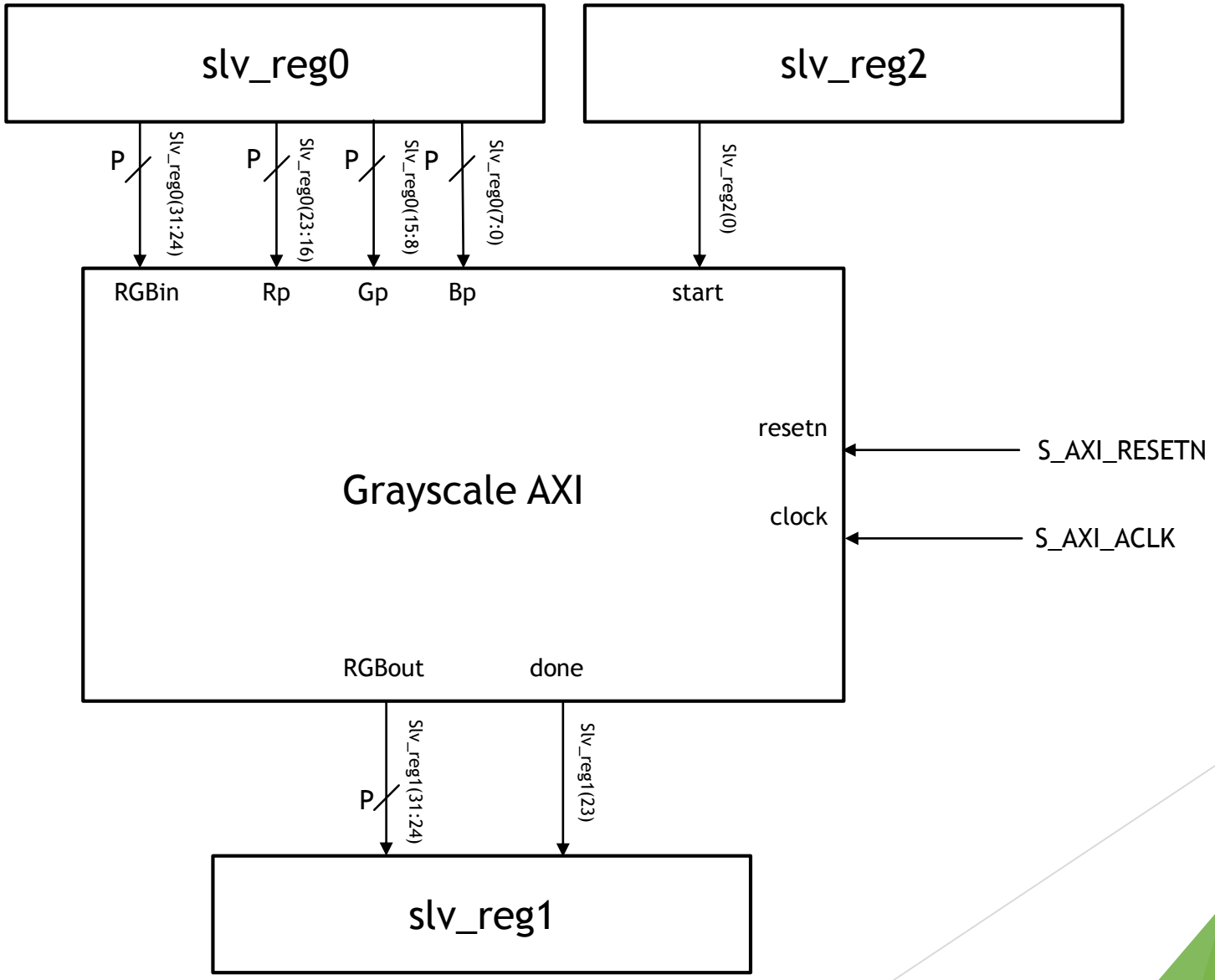
Grayscale - Implementation

- Input
  - 8bit RGB pixel
    - $b_7b_6b_5b_4b_3b_2b_1b_0$
  - RGB Percent
    - $R_p+G_p+B_p=100\%$
  - Start
- Output
  - 8bit RGB Pixel
    - $b_7b_6b_5b_4b_3b_2b_1b_0$
  - Done



*Note: P=8

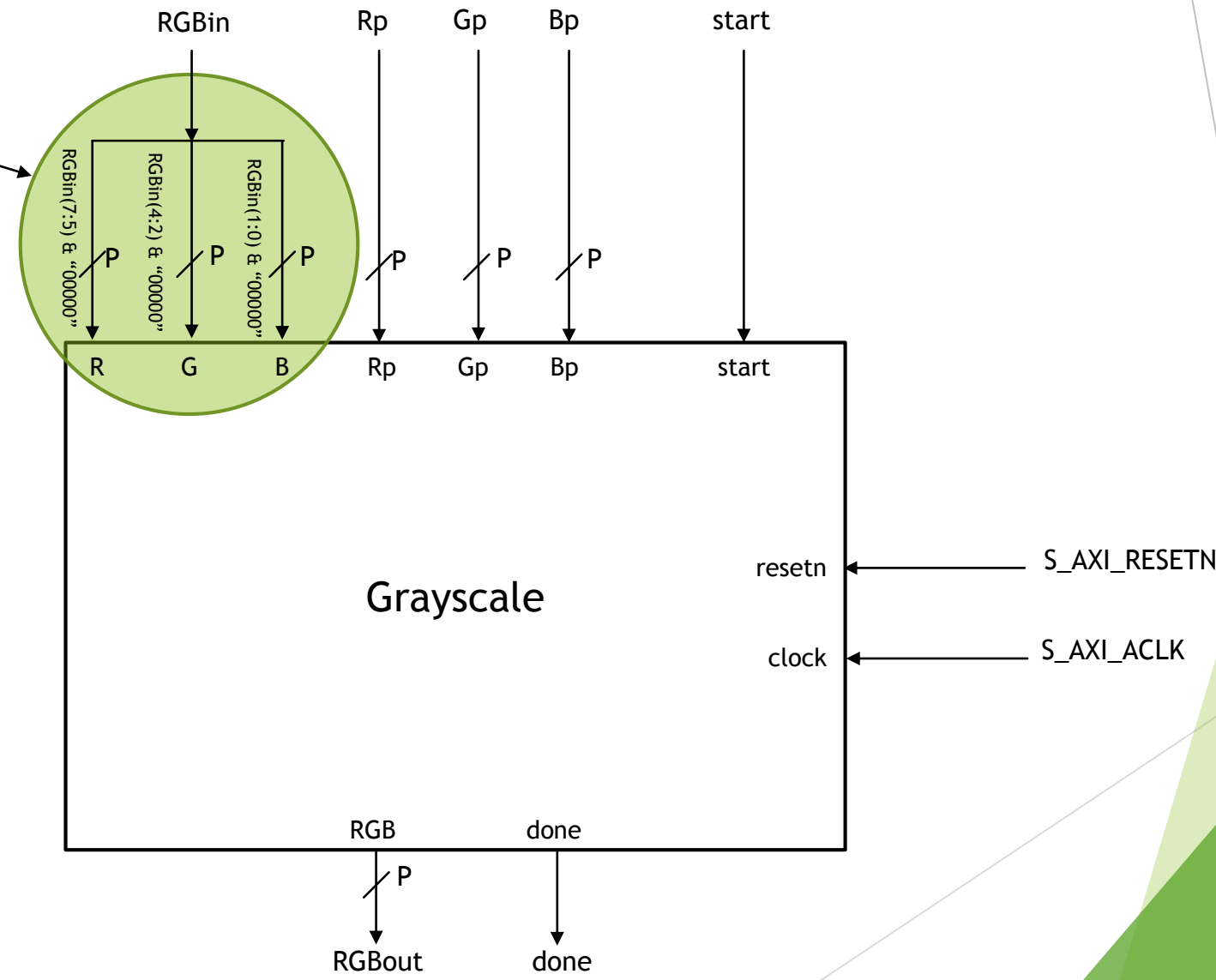
Grayscale – AXI Top



*Note: P=8

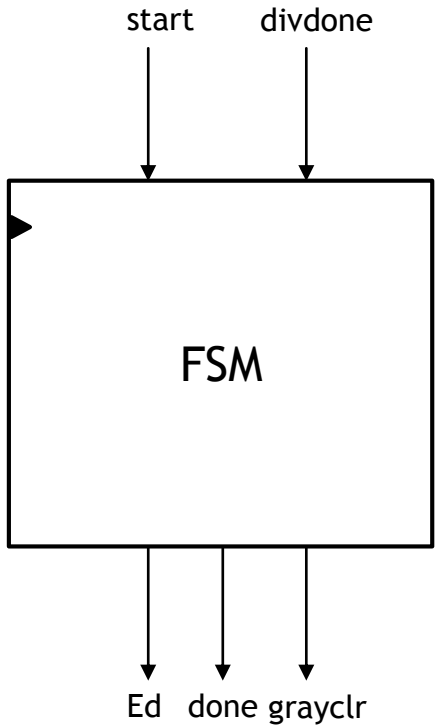
Grayscale – Top

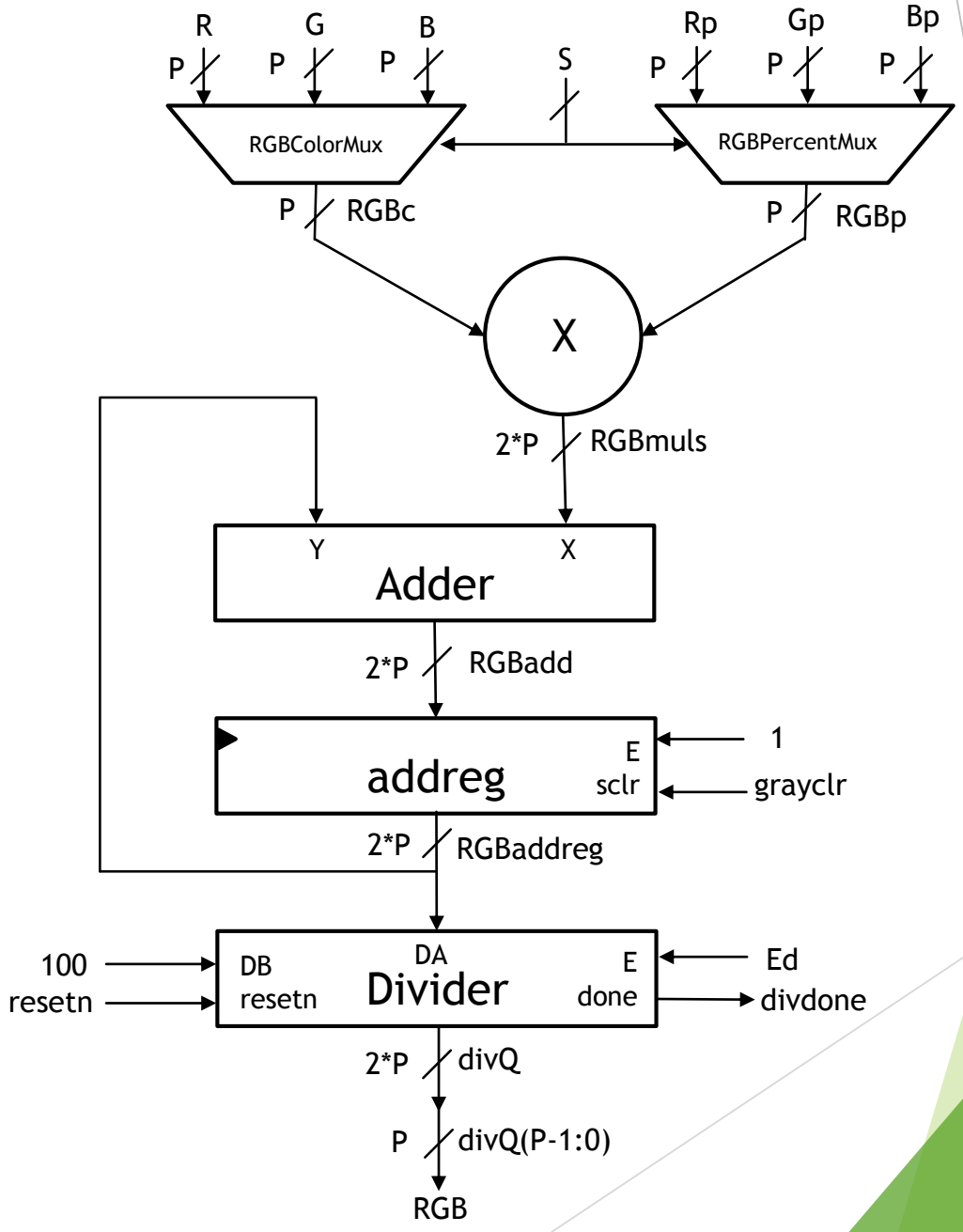
Decoding 8bit RGB
Red = (RGBin >> 5) * 32
Green = ((RGB & 0x1C) >> 2) *32
Blue = (RGBin & 0x03) * 64



*Note: P=8

Grayscale – DP

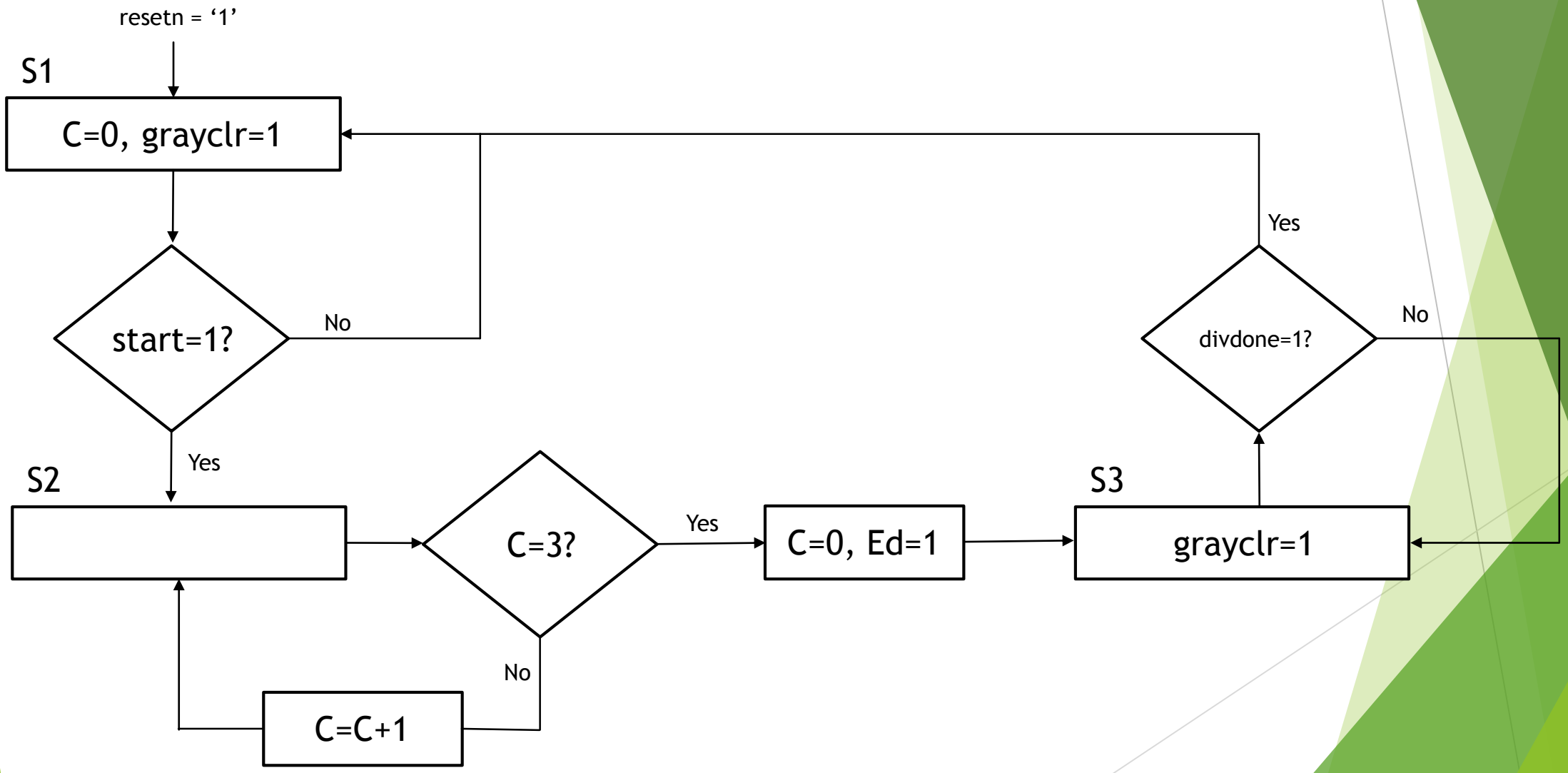




*Note: P=8

Grayscale – FSM

*Note: Unless otherwise noted, all values are initially set to 0

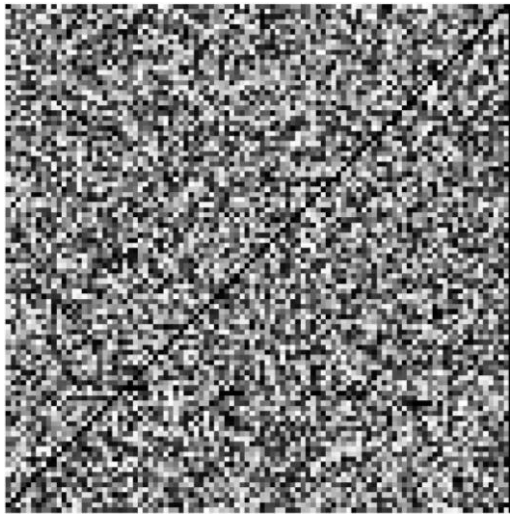


Software

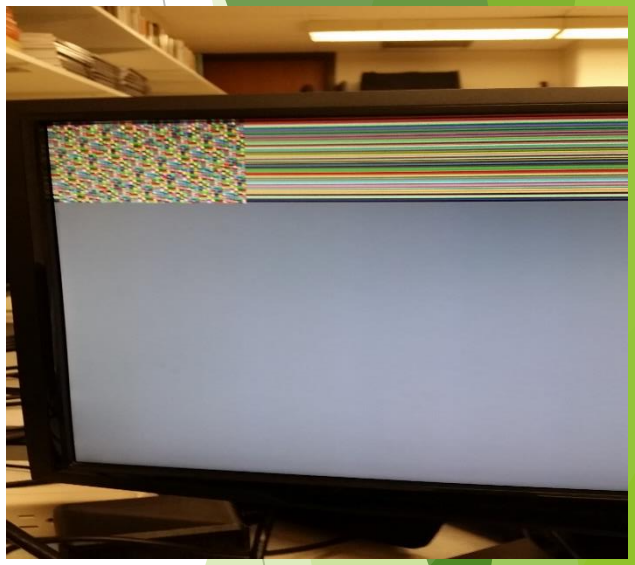
- Written in C
  - Image hard coded into header file
  - Converted output shown on terminal
- Image
  - Converted to text file using MATLAB script
  - Converted back to image file using MATLAB script

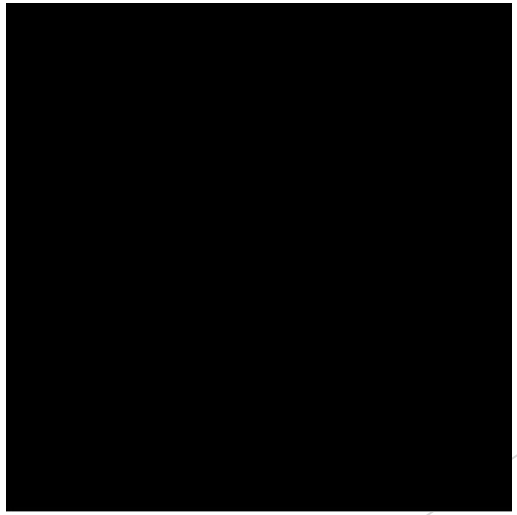
Results - Gaussian

Grayscale Filter Output
R=30%, G=59%, B=11%



Gaussian Filter 8bit Output





Gaussian Filter 16bit Output

Results - Grayscale

Original Image

Expected: Octave Output
R=29.89%, G=58.70%, B=11.40%

Result: Grayscale Filter Output
R=30%, G=59%, B=11%

Questions?