Overview

In many image processing algorithms, the first step is to obtain the image as a grayscale. Further analysis is possible only on such presented data. The video data collected by the research equipment near the telecommunications station is collected in the form of color (RGB) pictures. The test scenario assumes that the module converting RGB to grayscale has been damaged and must be recreated on the FPGA.

Theoretical background

The conversion from RGB to grayscale is carried out according to the formula:

$$gray = round(0.299R + 0.587G + 0.114B) \qquad (1)$$

Where:

gray – the intensity of the grayscale image
R – the intensity of the red component of the image
G – the intensity of the green component of the image
B – the intensity of the blue component of the image
round – output must be rounded to the nearest integer (for example round(234.20) = 234, round(234.49) = 234, round(234.50) = 235, round(234.51) = 235)

The listed intensities, for a single pixel, are numbers in the range of 0-255 (8 bits).

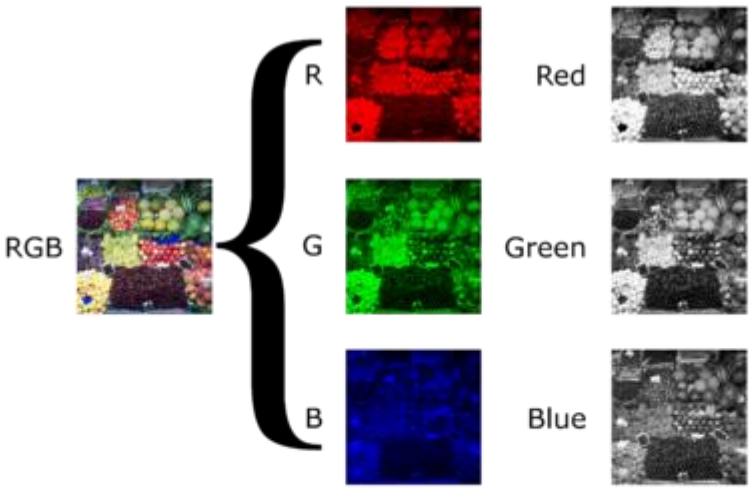

Figure 1 Composition of RGB from three grayscale images.

This algorithm is based on human color perception. The human eye is most sensitive to green, followed by red and then blue. Based on empirical data, one has decided to use the above coefficients in many applications.

Goal

Create an RTL module that converts given RGB data to a grayscale form according to equation 1.

Provided module

You can edit only the *rgb2gray.sv* file.

You only get the outline of the ***rgb2gray*** module. Inputs and outputs are declared as:

Table 1 List of inputs and outputs of the rgb2gray module.

Signal name	Signal type	Bit length
i_clk	Input	1 bit
i_rst	Input	1 bit
i_enb	Input	1 bit
i_RGB	Input	8-bit
o_gray	Output	8-bit
o_valid	Output	1 bit

Input and output interfaces

In one packet task receives **243 bytes** of data. Given that calculating one byte of output takes three bytes of input data (RGB to one byte of grayscale) task should send $\frac{243}{3} = \mathbf{81}$ **bytes** of output data.

Task receives data in the way shown in the figure below:

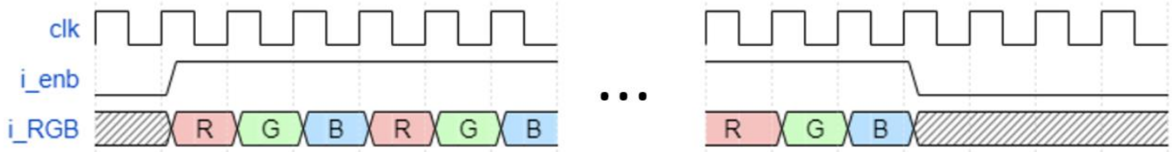

Figure 2 Input signal waves.

The task needs to compute one *o_gray* value from three adequate *i_RGB* values (R, G, B).

To send data from the task properly, one needs to provide the *o_valid* signal synchronized with data on the *o_gray* bus.

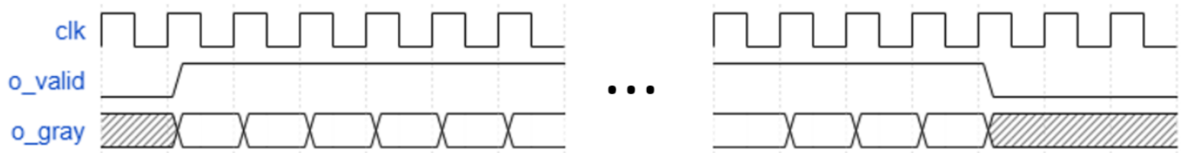

Figure 3 Output signal waves.

The output data can be with or without gaps. To transfer data correctly, the only requirement is to set the *o_valid* signal to high when on the *o_gray* bus are the data that you want to send.

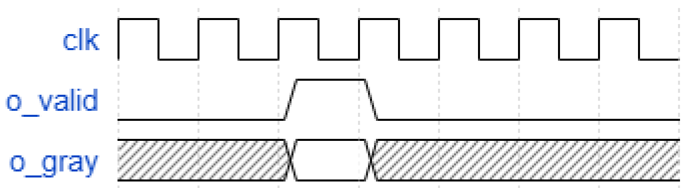

Figure 4 Output signal waves – another configuration. Only one o_gray data sample was sent.

Evaluating the task

For the task to be considered properly complete, all received pixels must be converted to grayscale according to Equation 1.

Example of correct answer:

For inputs:

R = 134

G = 88

B = 23

The expected (ex) output value is calculated as follows:

$$gray_{ex} = round(0.299R + 0.587G + 0.114B)$$

$$\begin{aligned} gray_{ex} &= round(0.299 \cdot 134 + 0.587 \cdot 88 + 0.114 \cdot 23) \\ &= round(40.066 + 51.656 + 2.622) \\ &= round(94.344) = 94 \end{aligned}$$

round – output must be rounded to the nearest integer (for example round(234.20) = 234, round(234.49) = 234, round(234.50) = 235, round(234.51) = 235)

Answer is marked as correct if the value sent from task (s) is equal to value expected:

$$gray_s = gray_{ex}$$

If answers for all received pixels are marked as correct, then the task is done correctly.