Image Processing IP on FPGA

Contents

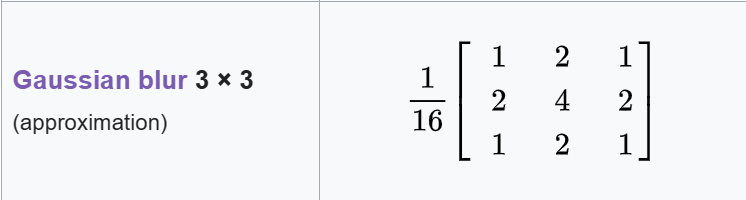
[The Convolution Entity: 3](#_Toc153396837)

# The Convolution Entity:

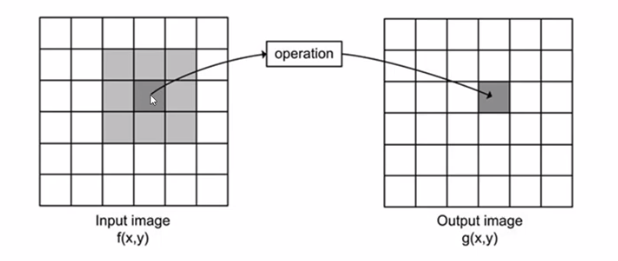
In our project we implement the neighborhood image processing in which the output pixel value does not depend only on the value of the corresponding input pixel, but also on it’s neighbors.

The output pixel value is the result of multiplying the corresponding input pixel and a number of it’s neighbors by a mask or a kernel which is the filter that will implement the required processing on the image.

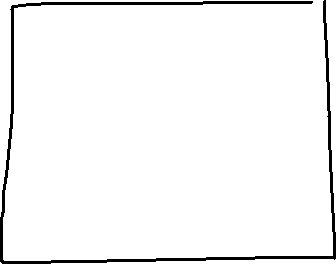
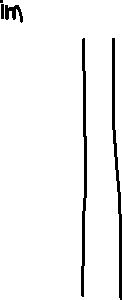
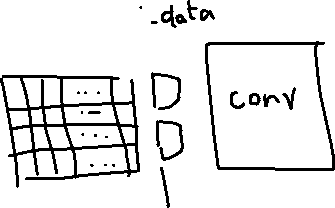
In the project, we chose to implement the Gaussian Blur kernel which blurs the input image, and it is represented by the following 3x3 matrix:



So, this entity is required to do multiply and accumulate operation(2-D convolution), as explained by the following image:



The imageControl entity:



Assume the input image is **512 \* 512** pixel in grey scale, so each pixel can be represented by 8 bits.

Control Logic:

* 4 line buffers are used to store read pixels from the image. If you wonder why 4 line buffers are used, well the answer is this will accelerate the operation so when the convolution unit is processing on 3 line buffers’ pixels the fourth line buffer is being filled with new pixels.
* The multiplexers’ select signal depends on the number of pixels that have been read from the line buffer (When it reaches 512).
* Furthermore, additional MUXes before the line buffers are a must to determine to which line buffer the new pixel data will be written.

Writing to line buffers:

* To determine to which line buffer the new pixel data will be written, pixelCounter is used to count the read pixels. Once it reaches 512 (since the width of the input image is 512), we assert the write enable signal (i.e., i\_data\_valid) of the subsequent line buffer and de-assert that of the already full line buffer.
* Note that i\_pixel\_data\_valid has to be set to enable writing to the line buffers. It’s an AXI control signal.

Reading from line buffers:

* We start processing the pixels from line buffers only when 3 line buffers have been filled so when totalPixelCounter reaches 1536 the rd\_line\_buffer signal is asserted.
* rdCounter needs to increment by 1 every cycle we read from the line buffers since it will be used to determine from which line buffer new pixels should be read. In other words, when it reaches 512, the read enable signal (i\_rd\_data) of the fourth line buffer will be asserted and de-assert the read enable signal of the appropriate line buffer according to the scheme mentioned in the videos.
* Each 3 pixels read from every line buffer are concatenated together to form the new 9 pixels to be processed in an output named o\_pixel\_data.

