INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY

26/C, Electronic City, Hosur Road, Bengaluru - 560100

PROJECT ELECTIVE REPORT

POWER EFFICIENT VLSI ARCHITECTURE OF A NEW CLASS OF DYADIC GABOR WAVELETS FOR MEDICAL IMAGE RETRIEVAL

by

SUSHMA R MT2023533
G LASYA MT2023539

Under the guidance of : Dr MADHAV RAO

Aim :

1. To implement the image convolution for an image with kernels derived from the Gabor filter bank and realize the hardware architecture for the same.
2. There are 4 different kernels for the modified Gabor filter with slightly different orientation parameters.
3. The algorithm implements the multiplication of the elements using shift and add operations where kernel values are expressed in powers of 2 (dyadicity).
4. All the 4 kernels need to be implemented parallelly.

Overview of the Progress

1. The openCV python function for Gabor wavelets were used on the circle test image in the paper to study how the gabor kernel works and how the output generated images will look like.

   Inference from this step:

a) The opencv function is unable to obtain kernel values as mentioned in the paper
b) The function was applied on several kinds of images (circle, square etc) and on images of different sizes (128x128, 250x650 , 256x256, 512x512).
c) The expected results i.e edge detection for 1 particular orientation was seen at the output image only for some cases.

2. Implemented the basic framework of the convolution in RTL (Verilog) without shift add but with given kernel values in the paper and generated the results.

   Inference from this step:

a) Again different images have different kinds of results.
b) Speculated with the mentor that this might be due to the inherent construction of images and their corresponding pixel values and alignment.
c) The python image and RTL image was a fair match.

3. Implemented the multiplication using shift add instead of * operator in RTL. Considerations were 8 bit input and output values to be shifted by a maximum of 9 bit values and the associated sign bit manipulation.

   Inference from this step:

a) The parallel multiplication and reuse of multiplied elements each time for multiplication of the subsequent elements in the kernel was a challenge to implement.

4. The parallel shifting and adding of all 4 kernels with reusing elements was designed in verilog.
   This step has the design of 2 modules: Convolution and Sorting.

   Sorting module sorts the 4 elements from the 4 kernels at a position (i,j) in increasing order such that the pixel element is shifted by its least required amount, for the next kernel value in the sorted array the shift is performed only by the difference amount between the two.

   Convolution computes the multiplication of the image with kernel values using shifting and adding and finally assigns the 4 generated results to the 4 different output buffers as is required by each orientation.

   Features:

   a) This code shifts based on the difference amounts and not completely reuses every shift value, thus rendering it more reusable and not hard coded as mentioned in the paper. This conclusion was drawn along with a PE update with the mentor (Ms Priyanka).
   b) At this stage convolution is sort of 1D as in, we are trying to perform element wise multiplication of image pixels with each value from the 4(6) corresponding kernels at a time.

5. Generating the exact coefficients in the paper for the slightly altered orientation parameters by modifying the python source code as customized to the sigma x, sigma y, frequency as per the paper. With this we tried to merge all the 4 kernel outputs and visualize the output as 1 final image.

   Inference at this step:
   a) The exact coefficients can be generated.
   b) This result was obtained by discussing with the mentor.
   c) All the 4 output images being merged into 1 gave desired result for the circle image.
   d) Also figured out resizing an image to a different value caused some sort of a pixel arrangement which gave incorrect output. Hence use 512x512 image is followed next.

6. Designed a 2D convolution code for the same since element wise multiplication did not yield desirable results. From the hardware perspective, we had to store a 512x512 image in a block ram and similarly write the 4 results also into the block rams.

A suitable design for accessing the 25 elements for each iteration from block ram and furthermore for signed addition in 2 D convolution was formulated.

With this, we hope to have achieved the architectural implementation of the proposed design in the paper.

The GITHUB repository has the following files:

All the python files for image to coe and output text file to image conversions.
The python code using gabor function from open cv and its source code.

RTL designs of initial convolution, sorting and convolution for element wise multiplication in 2 approaches.

RTL final design for 2D convolution for an FPGA board.