EVALUATION-II

IMPLEMENTATION OF PCA ON FPGA

Image processing is one of the most in-demand procedures and its implementation on hardware has proven to be highly-beneficial due to the flexibility it provides when hardware is to be changed or altered.

For simulation, we must convert test images to PGM format and they can be easily read using VHDL file access. The next step is to process the image as a stream as it comes in from the source.

The instructions or program written scripted in MATLAB prove to fall useless when it comes to implementation on hardware. Some of the problems faced are:

- The use of while loops is not allowed in Verilog coding
- Jump statements like break, continue and return cannot be used in Verilog
- Verilog reads a grayscale image and converts it to a hexadecimal format before performing any operations on it

LOOKUP TABLE

The LUT is actually implemented using a combination of the SRAM bits and a MUX: A LUT consists of a block of SRAM that is indexed by the LUT's inputs. The output of the LUT is whatever value is in the indexed location in it's SRAM.

In other words, whatever behavior you get by interconnecting any number of gates (like AND, NOR, etc.), without feedback paths (to ensure it is state-less), can be implemented by a LUT.

The way FPGAs typically implement combinatorial logic is with LUTs, and when the FPGA gets configured, it just fills in the table output values, which are called the "LUT-Mask", and is physically composed of SRAM bits.

CORDIC ALGORITHM(Volder's algorithm) -COordinate Rotation DIgital Computer

- Used when no hardware multiplier is available (e.g. in simple microcontrollers and FPGAs), as the only operations it requires are addition, subtraction, bitshift and table lookup. It is a type of shift-and-add algorithm
- It is a very efficient algorithm for hardwares where a hardware multiplier is not available, making it a very complex method to actually implement
- We will try this approach as we have to use Givens matrix(has sines and cosines) on FPGA. CORDIC will convert sine and cosine functions to a hardware usable form.

HDL Code Generation- MATLAB:

An in-built application in MATLAB, HDL Coder helps to generate VHDL or Verilog code from appropriate MATLAB script. The script has to be a function, without any floating point variables or data types.

Errors encountered:

Undefined dimensions of variables: Reduced the number of variables (optimized) and made sure that the sizes are appropriately assigned.
-some error in final matrix of pca.

2D Matrix not supported as input argument:
Reason: HDL coder assumes all external inputs (from external input ports) and hence does not allow for large matrices which require nxn number of pins. But our image is read from the memory, hence this error can be resolved.
-reading image from RAM is explored but not implemented yet.

Unsupported type String, cannot find function of java.lang.string input arguments:
Unresolved.

Found unsupported dimensions on matrix type at output port 0:
It is possible that MATLAB assumes it to be a dynamic matrix (which we rectified) and HDL does not support dynamic arrays.
If the input is a double, fixed point conversion fixes that.

Undefined input parameter in Fixed Point Converted Function:
The input to the newly generated code is undefined, although it is a parameter passed through a function. <- requires more study to resolve.

The conversion from floating point to fixed point has been done in MATLAB, but we do not know how to further it into HDL compatible code.

SIMULINK: We will need to develop the logic and design for generating a FPGA compatible code.

Vivado HLS simulator is also an implementation tool which allows the high level programming languages such as C/C++ instead of hardware description language such as Verilog or VHDL. Due to its iterative nature, the Jacobi method can easily be expressed as a series of loops, which is the main element used in Vivado HLS to implement the algorithms.