Image-Processing-on-FPGA

Project Report

Eklavya Mentorship Programme

At

SOCIETY OF ROBOTICS AND AUTOMATION,
VEERMATA JIJABAI TECHNOLOGICAL
INSTITUTE, MUMBAI

AUGUST - SEPTEMBER 2022

ACKNOWLEDGEMENT

We are incredibly grateful to our mentors Siddesh Patil ,Premraj Jadhav and Om Sheladia for their patience, motivation, enthusiasm, and immense knowledge they shared with us throughout the duration of the project.
We were able to explore a new domain of FPGA and successfully work on it only because of the expert guidance provided to us
We would also like to thank all mentors of SRA VJTI for their constant support and motivation and for giving us the opportunity to be a part of Eklavya 2022.

Our Team :

Harsh Bhosale
lt.harshbhosale@gmail.com

Hardik Bhortakke
hardikbhortakke@gmail.com

TABLE OF CONTENTS

PROJECT OVERVIEW

Description of Project :

In this project we take YCbCr color space image and apply JPEG compression algorithm to it. We apply algorithms in two stages, the first stage is the encoding stage. In this stage algorithms like Discrete cosine transform and quantization are applied and then we save it in memory in an encoded format. In the second stage we take the encoded string from memory and apply the decoding algorithms like dequantization and inverse DCT to it. This achieves 70 percent compression with little perceptible loss in image quality.

Tech Stack :

- Verilog HDL
- FPGA
- Tang Dynasty(TD4.3.815-64bit)

- Modelsim
- Tang Primer FPGA Dev Board

Brief Idea:

Through Verilog HDL, gate level digital circuits can be created with the help of Tang Dynasty IDE on Tang Primer FPGA Dev Board which can perform the image compression using JPEG encoding algorithm and is then stored . Another digital circuit is used to decode the compressed image to get the desired output.

2. THEORY

What is JPEG image compression?

"JPEG" stands for Joint Photographic Experts Group, the name of the committee that created the JPEG standard and also other still picture coding standards.

JPEG is designed for compressing either full-color or gray-scale images of natural, real-world scenes. It works well on photographs, naturalistic artwork, and similar material; not so well on lettering, simple cartoons, or line drawings.

JPEG is "lossy," meaning that the decompressed image isn't quite the same as the one you started with. (There are lossless image compression algorithms, but JPEG achieves much greater compression than is possible with lossless methods.) JPEG is designed to exploit known limitations of the human eye, notably the fact that small color changes are perceived less accurately than small changes in brightness. Thus, JPEG is intended for compressing images that will be looked at by humans. If you plan to machine-analyze your images, the small errors introduced by JPEG may be a problem for you, even if they are invisible to the eye.

Compression rate decreasing and hence quality increasing, from left to right

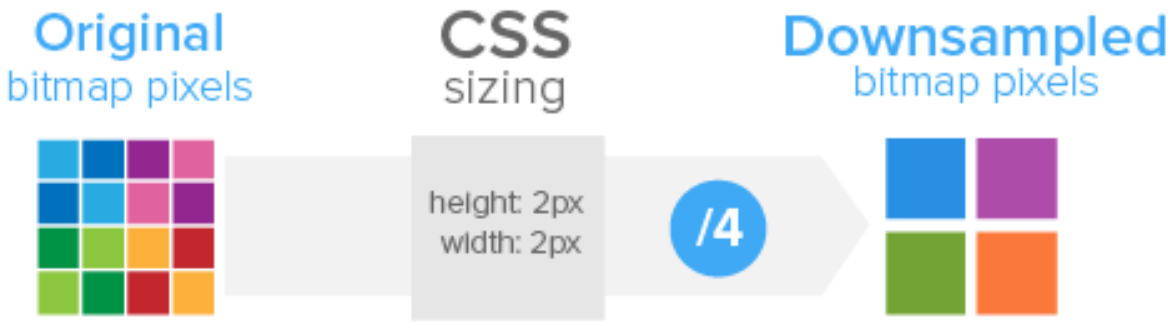
Encoding Algorithms :

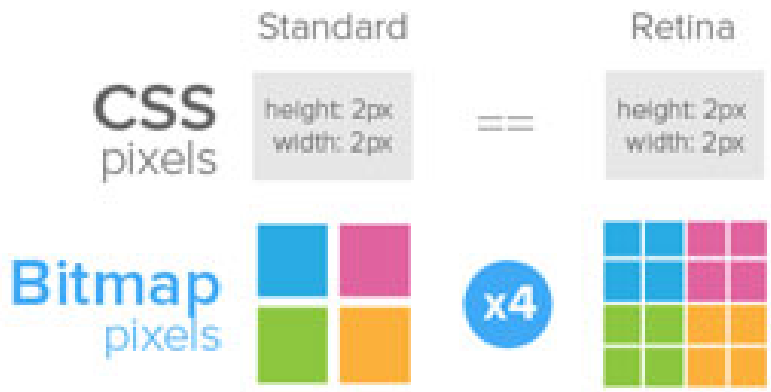
- ## Color Space Conversion

  Firstly the input image is in the form of RGB format representing the amount of red,green and blue color through the variation of these colors from 0 to 255 rest all the colors can be obtained where if all the 3 are 255 represent white color.
  This RGB image is converted into YCbCr format for image compression where Y represents Luminance or brightness and Cb and Cr represents Chrominance i.e the colors of the image.
  For the sake of this project we will be taking the input directly in the YCbCr format. For the conversion of RGB to YCbCr click here.

- ## Chrominance downsampling

  In chrominance downsampling, the pixel values of two blocks in horizontal and two blocks in vertical are taken i.e. 2x2 matrix and their values are replaced by their average values. Thus the amount of data reduces by a factor of 4.

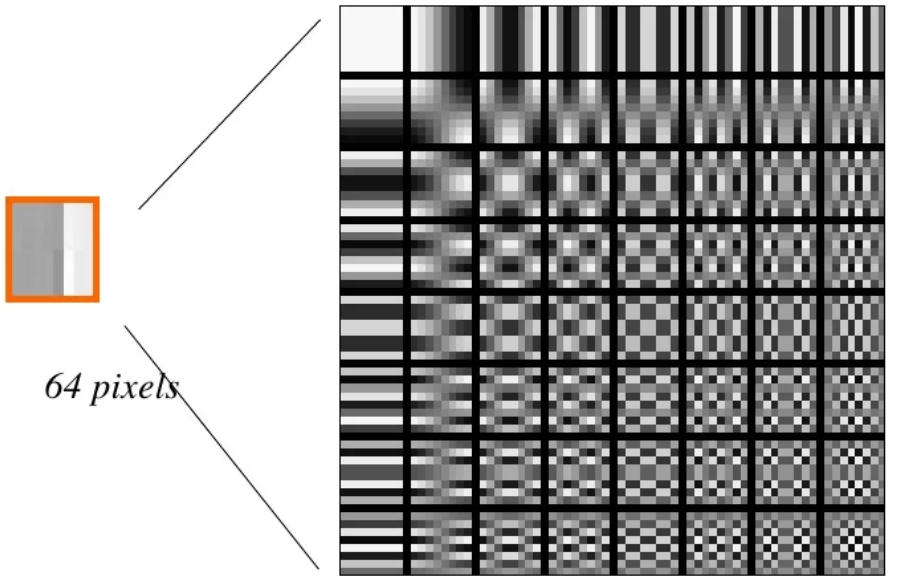




- **Discrete cosine transform**

  While performing Discrete Cosine Transformation(DCT), we subtract 128 from each pixel value so that the range of the pixel values change from 0 - 255 to (-128) - 128 creating negative pixel values as well. Now this converted matrix is manipulated through matrix multiplication. This manipulation is done by multiplying a cosine coefficient orthogonal matrix with the above converted matrix which is further multiplied by the transpose of the cosine coefficient matrix.

Discrete Cosine Transform (DCT) (2)



Cosine patterns/DCT basis functions

This coefficient matrix is fixed for a particular image and calculated by calculating the contribution of each pixel as mentioned in the above images where the frequency of image increases diagonally from top left to bottom right. This contribution is calculated as follows,

$$T_{i,j} = \begin{cases} \frac{1}{\sqrt{N}} & \text{if } i = 0 \\ \sqrt{\frac{2}{N}} \cos\left[\frac{(2j+1)i\pi}{2N}\right] & \text{if } i > 0 \end{cases}$$

This contribution is calculated for each pixel and a coefficient matrix is created called cosine coefficient matrix.
The matrix multiplication is done as :

$$D = TMT'$$

Where, T is the coefficient matrix , M is the converted matrix and T' is the transpose of the coefficient matrix.
Here D is the output DCT matrix.

- **Quantization**

This is one of the important steps which decides the compression quality and quantity. Higher the compression percentage lesser the quality.
Quantization matrix is obtained by multiplying each individual pixel value of a standard quantization matrix of 50% quality factor by (100 - quality level)/50 for quality level greater than 50 and by 50/(quality level) for quality level less than 50.
The matrix with 50% quality factor is as follows:

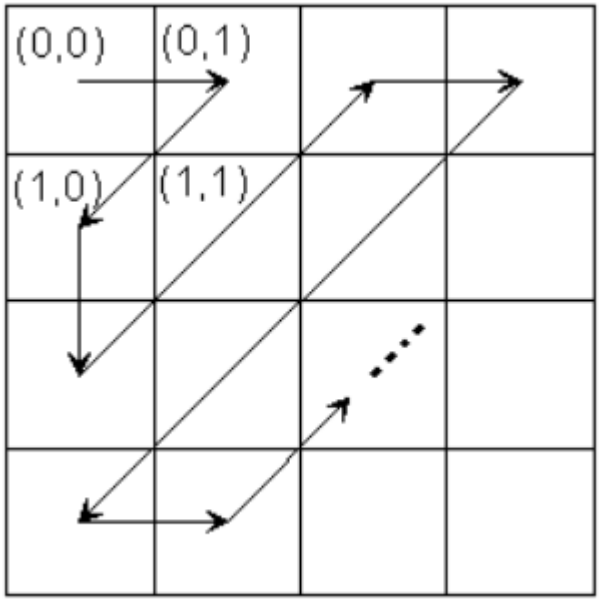
$$Q_{50} = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

After getting this Quantization matrix, each value of the DCT matrix is divided by the corresponding values of the Quantization matrix and rounded off to the nearest integer. Thus bottom left values of the matrix are zero and thus high frequency data is removed as it is not that sensitive to our

eyes.Thus the output matrix contains a lot of zeros and this data is further stored at a single place thus reducing its size.

- **Huffman Encoding**

Once the data is reduced, the reduced data needs to be removed or discarded and the data needs to be stored in the storage.
Huffman encoding is the step where actual data compression takes place and is the final step for JPEG image compression.



In this step the 2D matrix is converted into a 1D array or more precisely bitstream to be stored in the storage. The values of the 2D matrix are stored as 8 bit integers in one dimension wherein the first bit represents the flag where if the number is positive the flag is 0 and if the number is 0 then the flag is 1. If the flag is 1 i.e. the number is 0 then the next 7 bits store the number of continuous zeroes and if the flag is 0 then the next 7 bits store the positive number encountered. The conversion takes place in the order as shown above to get maximum compression of data.

Decoding algorithms:

The stored data is now compressed to a smaller size but this compressed data cannot be viewed directly as we have encoded it in order to reduce its size. Thus we need a decoder which will perform a decoding algorithm every time we wish to see the image and we can see the desired output but as this is lossy compression some of the data is lost and if we zoom in to see each pixel value it won't be a 100% match.

- **Decoding**

  In this Step we start retracing the steps we performed while encoding but in reverse order. Thus the last step in encoding is the first step in decoding that is Huffman decoding where we create a 2D matrix and start storing the values as we did during encoding but instead of 2D to 1D, we are now converting the 1D array to 2D array by recognizing the first bit as flag and the next 7 bits as counter or the number depending upon the value of the flag.

- **Dequantization**

  Dequantization is the exact reverse of the quantization process where instead of division, multiplication takes place. Here each individual element of the matrix is multiplied by the respective element of the quantization or dequantization matrix which will be fixed and known. Thus, the output will be similar to that of the DCT matrix output wherein the bottom right values or lesser values will be replaced by zero.

- **Inverse Discrete Cosine Transform**

In inverse DCT the DCT matrix or the output of dequantization say matrix R is taken. The transpose of the coefficient matrix used during encoding is used and multiplied with the matrix R which is further multiplied by the coefficient matrix. To get the required output in the range of 0 to 255 which is the pixel range, 128 is added to each element of the matrix to get the desired output which can be shown by its pixel values.
This calculation is represented as follows:

$$N = round(T' \; R \; T) + 128$$

Here, N is the desired output for Y while Cb and Cr needs to go through one more process of Chrominance resampling.

3.References

1. What is an FPGA? Programming and FPGA Basics - INTEL® FPGAS
2. Introduction to JPEG Compression (tutorialspoint.com)
3. https://www.youtube.com/watch?v=Kv1Hiv3ox8I
4. https://www.youtube.com/watch?v=n_uNPbdenRs