Accelerating Discrete Cosine Transform for faster Image compression

Kirthi Vignan Reddy
CVEST
IIIT Hyderabad
Hyderabad, India
kirthi.vignan@research.iiit.ac.in

E. Siddharth Pavan
CVEST
IIIT Hyderabad
Hyderabad, India
siddharth.pavan@students.iiit.ac.in

***Abstract*—Here we are implementing a 2D-DCT algorithm which is very useful in image compression. This can be utilised to accelerate image compression for a full HD video streaming at 60fps. Each transformation needs many multiplication and accumulate operations to be computed, hence there is a need to accelerate this procedure on a device for better performance [1]. Various defects for conventional operation of the transform and methods to overcome them, there by optimising the algorithm are discussed. The program is accelerated on Virtex Ultrascale FPGA.**

***Index Terms*—Discrete Cosine Transform, FPGA acceleration, Parallelization**

I. Introduction

A discrete cosine transform (DCT) expresses a finite sequence of data points in terms of a sum of cosine functions oscillating at different frequencies. It is one of the important image compression algorithms used in image processing applications. It involves a lot of multiplications and additions. The general use of this transformation is in the field of lossy information, where the image is stored in less memory. For image compression, we use a 2D-DCT implementation which involves breaking the image into 8x8 pixel blocks and perform the transform. The output is just the coefficients of the corresponding frequency component of the image block. Later the coefficient matrix is then encoded in the entire JPEG compression module.

Here, for the naive implementation of the transform, we must apply a numerous amount of multiplication and accumulate operations for just a single coefficient. And applying parallelization for better performance is not possible here. Hence we move to a different approach in solving both the problems.

II. Related Work

DCT transform has many configurations and approaches based on the domain it is used in [2]. A 1D-DCT is used in signal processing where we try to encode the incoming signal. A 2D-DCT is mainly used in the image and video processing or compressing domain. This is a very useful application where in the input image or video frame is taken and broken down into 8x8 pixelled images and cosine transformed to get the transformed output.

$$DCT(u) = \frac{c(u)}{2}\Sigma_{x=0}^{7} f(x) cos(\frac{\pi(2x+1)u}{16}) \quad (1)$$

III. Proposed Design

$$DCT = \begin{bmatrix} c_{00} & c_{01} & \dots & c_{07} \\ c_{10} & c_{11} & \dots & c_{17} \\ \vdots & \vdots & & \vdots \\ c_{70} & c_{71} & \dots & c_{77} \end{bmatrix} \begin{bmatrix} X_{00} & X_{01} & \dots & X_{07} \\ X_{10} & X_{11} & \dots & X_{17} \\ \vdots & \vdots & & \vdots \\ X_{70} & X_{71} & \dots & X_{77} \end{bmatrix} \begin{bmatrix} c_{00} & c_{10} & \dots & c_{70} \\ c_{01} & c_{11} & \dots & c_{71} \\ \vdots & \vdots & & \vdots \\ c_{07} & c_{17} & \dots & c_{77} \end{bmatrix}$$

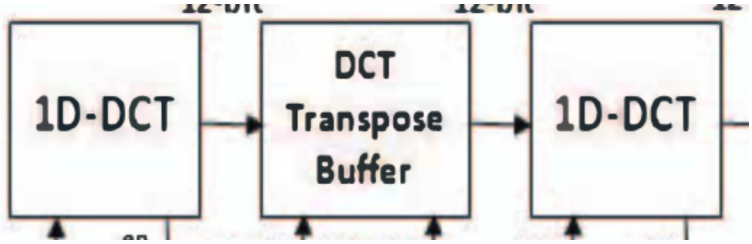
Here we use a method that is best described in [3] . Instead of using a 2D-DCT matrix, we use the 1D-DCT matrix and its transpose. Then we multiply it individually with the pixel block matrix to get the required DCT coefficient matrix. The normal 2D-DCT matrix has to be computed at each different frequency along the axes and then need to be multiplied and accumulated at each step in order to get corresponding coefficient. This approach has many drawbacks as it needs additional computation and its very conventional method makes it unsuitable for parallelizing the process. Hence the new method solves both the problems, delivering legitimate outputs.

$$DCT(u,v) = \frac{c(u)c(v)}{4}\Sigma_{x=0}^{7}\Sigma_{y=0}^{7} f(x,y) cos(\frac{\pi(2x+1)u}{16}) cos(\frac{\pi(2y+1)}{16}$$

(2)

The coefficient matrix obtained has a unique property, where we try to extract values corresponding to lesser frequencies. These are present in the top left of the coefficient matrix. The resultant matrix is then encoded and stored in an appropriate manner. This is the entire flow of the JPEG conversion, where the main focus lies in the cosine transformation of the image.

graphicx



A. Roof-line Analysis

No.of DSPs =6800 Each DSP can do 1 MAC operation The Clock Frequency of FPGA is 250MHz

The image taken for DFT is HD image so its dimensions are 1280*720=921600 pixels In our algorithm we take 8x8 pixels and apple 2d DFT transform that is 2 matrix multiplications with 2 8x8 matrices.

In Matrix multiplication generation of each element requires 8MAC operations .So total 8x8 matrix multiplication takes about 64x8 MAC operations =512 MAC operations for 64 pixels so it takes about =73,72,800 MAC operations in total for 1 matrix operation for 2 we need 1,47,45,600MAC operations

The FPGA can perform 6800*250 MAC operations =17,00,000/second

The 921600 pixels require 1280*720 bytes to transfer =0.99216 Megabyte The input bandwidth is 16GB/sec

By this we can conclude that it is compute bound

IV. Result

Vector size (image taken as 1 d)	Computation time
2048	58014.2ms
2048*2	65016.1ms
2048*4	66015.3ms

References

[1] A. Lotfi, A. Rahimi, A. Yazdanbakhsh, H. Esmaeilzadeh, and R. K. Gupta, "Grater: An approximation workflow for exploiting data-level parallelism in fpga acceleration," in *2016 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2016, pp. 1279–1284.

[2] M. Jridi and A. Alfalou, "A low-power, high-speed dct architecture for image compression: Principle and implementation," in *2010 18th IEEE/IFIP International Conference on VLSI and System-on-Chip*, 2010, pp. 304–309.

[3] E. Aggrawal and N. Kumar, "High throughput pipelined 2d discrete cosine transform for video compression," in *2014 International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT)*, 2014, pp. 702–705.