Implementation of 2-D Discrete Cosine Transform in JPEG Compression

Fengying Deng                                                                   Ming Dai

I. INTRODUCTION

DCT transform is widely used in audio and image compression and is the basis for compression JPEG and MPEG standards. In this report, we present implementations of 8x8 2-D DCT algorithm used in JPEG standard. In section II a fast algorithm using only eleven multipliers for 1-D DCT is discussed. Then the algorithm of 2-D DCT based on 1-D DCT is derived. Section IV will demonstrate different architectural optimizations on 2-D DCT. Section V provides performance comparison among those architectural optimization techniques.

II. 1-D DCT OPTIMIZATION

The basic equation for 1-D DCT is shown. Direct implementation of equation would require 64 multiplications (in our case N=8). In our fast algorithm the number of multipliers is reduced to eleven.

\[ y(k) = \alpha(k) \sum_{n=0}^{N-1} x(n) \cos \left( \frac{\pi(2n+1)k}{2N} \right) \]

\[ x(n) = \sum_{k=0}^{N-1} \alpha(k)y(k) \cos \left( \frac{\pi(2n+1)k}{2N} \right) \]

Where, \( \alpha(0) = \frac{1}{\sqrt{N}} \), \( \alpha(k) = \frac{1}{\sqrt{2}} \) \( k \neq 0 \).

Our fast 1D-DCT algorithm utilizes two modules: butterfly and rotation. A butterfly module can be implemented with two adders, while with some tricks the rotation is realized with three multipliers and three adders. A full structure of 8-point DCT computation is shown.

III. 2-D DCT ALGORITHM

Rewrite the 1-D DCT in matrix form we have

\[ [Y] = [C][X] \]

where, \([Y]\) and \([X]\) are N-point output and input columns, and \([C]\) is N-dimension square transform matrix. Then the 2-D DCT transform can be represented in the form of

\[ [Y] = [C][X][C]^T \]

where, \([Y]\) and \([X]\) are N-dimension output and input square matrices, and \([C]\) is N-dimension square transform matrix. Rewriting the 2-D DCT expression we can see that 2-D DCT can be computed via two-step 1-D DCT and transposition.

\[ [Y] = ([C][X][C]^T)^T \]

IV. ARCHITECTURAL DESIGN AND IMPLEMENTATION

In this section we will first introduce the building blocks in our 2-D DCT system, including butterfly and rotation and shifters.

1-D DCT Block

Because pixel value of digital image is integer in the range of (0,255) (8-bits), and 8-point DCT (3-bits) is applied for JPEG compression. Output word-length of all multiplier and adders are set to sign (16, 4).

a) Pipelined Butterfly:

b) Pipelined Rotation:

Direct form (4 multis, 2 adds)
Optimal form (3 mults, 3 adds)

Equations of direct form:

\[ O_0 = k_1 \sin \left( \frac{n}{16} \right) + k_1 \cos \left( \frac{n}{16} \right) \]
\[ O_1 = k_2 \sin \left( \frac{n}{16} \right) - k_1 \cos \left( \frac{n}{16} \right) \]

Equations of revised form: (one less multiplier at the expense of one extra adder and one additional latency)

\[ O_0 = k_1 (t_1 + t_2) \sin \left( \frac{n}{16} \right) + k_1 \cos \left( \frac{n}{16} \right) - \sin \left( \frac{n}{16} \right) \]
\[ O_1 = k_1 (t_1 + t_2) \sin \left( \frac{n}{16} \right) - k_1 \cos \left( \frac{n}{16} \right) + \sin \left( \frac{n}{16} \right) \]

c) Pipelined 1-D DCT using Butterfly and Rotation:


d) Shifters: The normalized 2-D DCT has 1/N in front, which is performed by 3 bit right shift.

c) Folding and Interleaving
The register array is filled up from left to right after performing 1-D DCT on columns of new input image after 8 cycles. Then the intermediate matrix is pushed out from top to bottom for transposition, while the 1D-DCT result of rows of intermediate matrix fill the register array up from bottom to top. After another 8 cycle result image of 2-D DCT is done, and is pushed out from right to left for normalization.

Because the logic depth of all the architecture is the same (applying the same pipelined 1-D DCT module, in which 16-bit multiplier is the worst-case logic delay), the maximum achievable clock frequency is also the same. According to the result, pipelining improves the performance significantly by decreasing logic depth and thus boosting clock frequency. In contrast, Folding and Interleaving reduces the area and power (however, energy stays the same) dramatically while scarifies performance (factor of folding is 2, factor of interleaving is 8). To achieve the same performance, more power (energy) is needed to speed up the logic and hence enable those blocks to operate at a higher frequency by the factor of folding and interleaving. By combining pipelining, folding and interleaving, the performance-area-power efficiency can be optimized (e.g. 2D-DCT (F, P) is much more efficient than direct 2D-DCT without pipelining).

### V. PERFORMANCE COMPARASION

**Power, Area and Timing**

P: pipelining, F: folding, I: interleaving

### REFERENCES


