2-D Separable Discrete Hartley Transform Architecture for Efficient FPGA

Vijay Kumar Sharma, Richa Agrawal, U. C. Pati, K. K. Mahapatra
National Institute of Technology, Rourkela, India-769008
#kmaha2@gmail.com

Abstract — Discrete cosine transform (DCT) is usually used in JPEG based image transform coding. This paper presents separable 2-D discrete Hartley transform (SDHT) and its Distributed Arithmetic (DA) based hardware architecture as an alternate to DCT in transform based coding of image compression. The proposed DA architecture for 1-D DHT has very less computations as compared to existing 1-D DCT. The proposed DHT architecture implemented in FPGA indicates a significant hardware savings as compared to FPGA resources used in an efficient memory based DA approach. The additional advantage of SDHT is that its inverse transform is same as forward transform with a constant division. This is demonstrated through a Xilinx FPGA XC2VP30 device.

Keywords — Distributed Arithmetic, Discrete Hartley Transform, Discrete Cosine Transform, JPEG, Offset Binary Coding.

I. INTRODUCTION

Digital image compression is the most important part in the multimedia applications which aims to reduce the number of bits in an image data for its efficient storage (less storage area). JPEG based still image compression follows three steps i.e. transform, quantization and coding to compress an image [1]. Reverse process comprising decoding, dequantization and inverse transform is used for image de-compression [2,3]. Discrete cosine transform (DCT) is used to transform the image from spatial domain to frequency domain. During quantization less important frequencies are discarded and is termed as lossy image compression. Bracewell has drawn attention to the discrete Fourier transform (DFT) as a substitute for the discrete fourier transform (DFT) [4,5]. Many applications of DHT in signal processing and communications have been presented in the literatures [6-8]. DHT is used in JPEG based image compression with DHT replacing the DCT in [9]. PSNR comparable to DCT based image transform has been obtained. The advantage of using DHT over DCT is that a dequantizer is not required at the decoder side and hence facilitates saving of hardware resources.

Hardware implementation of DHT requires a large number of multipliers. Since multiplier requires much more hardware as compared to adder and additionally consumes more power, it is not recommended in a battery running portable device [10,11]. Distributed Arithmetic (DA) is a digital signal processing technique that implements multiplication without the use of multiplier. Two approaches have been emerged for DA. One is look-up table based [12] and other is without look-up table [13]. Field programmable gate array (FPGA) is an emerging technology in VLSI design because of its many advantages like reprogramability, low cost, fast design cycle, in-circuit programmability etc [14-18].

In this paper separable 2-D DHT (SDHT) is proposed in place of DCT for image transform. The image transformed by SDHT is reconstructed after taking inverse SDHT (same as forward with a constant division). No error in the reconstructed image indicates that SDHT can be used for image transform [19-23]. SDHT is implemented in Xilinx FPGA XC2VP30 device using memory based DA and proposed DA for DHT. Proposed DA for DHT requires very less hardware for its implementation as compared to memory based DA [24-30]. It also requires less computation as compared to DCT in [13].

The organization of this paper is as follows. Section II contains the introduction of separable 2-D DHT. Use of SDHT in place of DCT and image reconstruction by taking inverse SDHT is explored in section III. Look-up table based DA for 1-D DHT and its hardware implementation is explained in section IV. Section V describes the proposed DA for 1-D DHT. Section VI discusses the FPGA implementation results and comparisons in terms of hardware utilization. Finally conclusions are drawn.

II. SEPARABLE 2-D DHT

Separable 2-D DHT is given by the equation,

\[ Y(u, v) = \sum_{x=0}^{N-1} \sum_{y=0}^{M-1} f(x, y) \cos \left( \frac{2\pi ux}{N} \right) \cos \left( \frac{2\pi vy}{M} \right) \] (1)

and the 1-D DHT is given by the equation,

\[ Y(k) = \sum_{n=0}^{N-1} X(n) \cos \left( \frac{2\pi nk}{N} \right) \quad k = 0, 1, \ldots, N - 1 \] (2)

where \( H_{nk} = \cos \left( \frac{2\pi nk}{N} \right) \), is the transform’s Kernel \( \cos(x) = \cos(x) + \sin(x) \) and \( X(n) \) is input data.

The advantage of using separable 2-D DHT equation (1) is that it can be implemented with 1-D DHT given by equation (2) by row-column decomposition method. Non
separable 2-D DHT equation requires further calculations after taking row-column decomposition [4].

III. IMAGE TRANSFORM AND RECONSTRUCTION FROM SDHT

Fig. 1 shows the JPEG baseline encoder which uses DCT based transform for image compression. Fig. 2 shows the DHT based encoder. Fig. 3a shows the image used for DHT transform by 8x8 block using SDHT. The reconstructed image after taking inverse SDHT as shown in Fig. 3 (b) with almost no distortion proves that SDHT can be used in place of DCT for image transform. Energy calculation and thresholding is used in DHT based encoder instead of quantization table in DCT based encoder.

IV. LOOK-UP TABLE BASED DA FOR 1-D DHT

Representing X(n) in 2’s complement fractional form

\[ X_n = -x_{n,i-1} + \sum_{m=1}^{i-1} x_{n,i-1-m} 2^{-m} \]  \hspace{1cm} (3)

where \( x_{n,m} \) is \( m\)th bit from LSB and \( m=i-1 \) is sign bit

Negative of \( X_n \) is represented by

\[ -X_n = -\bar{x}_{n,i-1} + \sum_{m=1}^{i-1} \bar{x}_{n,i-1-m} 2^{-m} + 2^{-(i-1)} \]  \hspace{1cm} (4)

Representing \( X_n \) in Offset binary Code (OBC) we have

\[ X_n = \frac{1}{2} [X_n - (-X_n)] = 1 \left[ - (x_{n,i-1} - \bar{x}_{n,i-1}) + \sum_{m=1}^{i-1} (x_{n,i-1-m} - \bar{x}_{n,i-1-m}) 2^{-m} - 2^{-(i-1)} \right] \]  \hspace{1cm} (5)

Define, \( d_{n,m} = \begin{cases} x_{n,m} - \bar{x}_{n,m}, & \text{for } m \neq i - 1 \\ - (x_{n,i-1} - \bar{x}_{n,i-1}), & \text{for } m = i - 1 \\ \end{cases} \) \hspace{1cm} (6)

and \( d_{n,m} \in (-1,1) \)

Now from Equations (5) and (6),

\[ X_n = \frac{1}{2} \left[ \sum_{m=0}^{i-1} d_{n,i-1-m} 2^{-m} - 2^{-(i-1)} \right] \]  \hspace{1cm} (7)

From Equations (2) and (7),

\[ Y(k) = \sum_{n=0}^{N-1} \frac{1}{2} H_{nk} \left[ \sum_{m=0}^{i-1} d_{n,i-1-m} 2^{-m} - 2^{-(i-1)} \right] \]

\[ = \sum_{m=0}^{i-1} \left( \sum_{n=0}^{N-1} \frac{1}{2} H_{nk} d_{n,i-1-m} \right) 2^{-m} - \left( \sum_{n=0}^{N-1} \frac{1}{2} H_{nk} \right) 2^{-(i-1)} \]  \hspace{1cm} (8)

From equation (8), Y(k) can be computed for different values of k, let for example k=1 the summation \( \sum_{n=0}^{N-1} \frac{1}{2} H_{nk} d_{n,i-1-m} \) can be stored in look-up table (ROM) after pre-additions as there can be a maximum of \( 2^n \) possible combinations as shown in Table 1 for \( n=3 \). Inputs are transmitted in serial, 1 bit at a time. One bit of each input forms the address of memory location and they are accumulated after shifting left the previous result. Second accumulation term is a constant value and it is similar to the first row in table and hence there is no need of additional memory for it. Final output is obtained after \( i+1 \) clock cycle where i is number of bits in input (1 clock cycle for adding constant value). From Table 1 it can be seen that upper half is same as lower half except sign is opposite. Hence look-up table can be halved as only upper half is stored and bits of \( X_i \) is X-ORed with all other inputs so that when it is 0 upper half is addressed and accumulated with no sign change but when it is 1 again upper half is addressed and accumulated with sign reversed.
1-D DHT is implemented in Xilinx XC2VP30 FPGA device.

Both memory based and proposed architecture for 1-D DHT are done by shifting. Table 4 and 5 shows the hardware utilization summary which indicate considerable hardware savings in proposed DA for DHT as compared to memory based DA. 2-D DHT is implemented using row column decomposition method.

V. PROPOSED DA FOR 1-D DHT

Considering the periodicity and symmetry of trigonometric functions equation (1) can be written as,

\[ y(0) = [x(0) + x(4) + x(1) + x(5) + x(2) + x(6) + x(3) + x(7)]A \]
\[ y(1) = [x(1) - x(5)]C + [x(2) - x(6)]B + [x(0) - x(4)]A \]
\[ y(2) = [x(2) + x(6)](-A) + [x(1) + x(5)]B + [x(3) + x(7)](-B) \]
\[ + [x(0) + x(4)]A \]
\[ y(3) = [x(2) - x(6)](-B) + [x(3) - x(7)]C + [x(0) - x(4)]A \]
\[ y(4) = [x(1) + x(5)](-A) + [x(2) + x(6)]A + [x(3) + x(7)](-A) \]
\[ + [x(0) + x(4)]A \]
\[ y(5) = [x(1) + x(5)](-C) + [x(2) - x(6)]B + [x(0) - x(4)]A \]
\[ y(6) = [x(0) + x(4)]A + [x(1) + x(5)](-B) + [x(2) + x(6)](-A) + [x(3) + x(7)]B \]
\[ y(7) = [x(2) - x(6)](-B) + [x(3) - x(7)](-C) + [x(0) - x(4)]A \]

where,
\[ A = \cos(0), \; B = \cos(\frac{\pi}{8}), \; \text{and} \; C = \cos(\frac{\pi}{4}) \]

Representing in DA form as in [13] for DCT we get Adder/Subtractor matrix for DHT for all data as in Fig.4. Table 2 shows the explanation of Fig.3. Divide and multiply operations are done by shifting. Y^2 implies shifting n bits. Negative sign in n implies left shift where as positive sign implies right shift. Positive sign in table 2 implies ALUs perform addition and negative sign implies subtraction operation. As an example, let’s take the fourth column for calculating Y(2). Y(2) is the sum of Y(1)^2, Y(0)^2, Y(1)^2, Y(2)^2, Y(3)^2, Y(4)^2, Y(5)^2, Y(6)^2, and Y(7)^2. The values of Y(2), Y(0)^2, Y(1)^2, Y(2)^2, Y(3)^2, Y(4)^2, Y(5)^2, Y(6)^2 can be got from R3, R5, 0, 0, 0, 0, 0, 0. So Y(2) can be calculated as,

\[ Y(2) = R3 \ast 2 + R5 \]

Compared to DCT adder/subtractor of [13], DHT adder/subtractor requires less no. of ALU + adders (only 4 ALUs+7 adders in proposed DHT but 9 ALUs +6 adders in DCT).

VI. FPGA IMPLEMENTATION RESULTS AND COMPARISONS

Both memory based and proposed architecture for 1-D DHT is implemented in Xilinx XC2VP30 FPGA device and table 4 and 5 shows the hardware utilization summary and table 6 shows the hardware utilization summary.

8x8 image data of table 7 is implemented and table 8 shows the matlab simulation result. Fig.5 shows the hardware implementation result using memory based DA method and fig.6 shows the hardware implementation result.
of proposed DA for DHT method. It is evident that proposed DA method implementation has less error as compared to memory based DA. Floating point representation is required to store pre-computed values in ROM for accuracy and hence floating point circuitry has to be used which will further increase the hardware resource in memory based DA. Also memory based DA is slow as it requires \(i+1\) clock cycles \((i \text{ is number of bits in input image data of one pixel})\) for 1-D DHT and subsequently \(8 \times ((i+1)+(i+1+2))\) clock cycles \((\text{generally input image pixel is represented by 8-bit, i.e } i=8, \text{ and hence total clock latency will be } 160 \text{ clock cycle})\) for 2-D DHT implementation. Although memory based DA work with faster clock as it has lesser calculations per clock cycle to perform than proposed DA for DHT.

### Table 4

<table>
<thead>
<tr>
<th>Logic Utilization</th>
<th>Used</th>
<th>Available</th>
<th>Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Slices</td>
<td>561</td>
<td>13696</td>
<td>4%</td>
</tr>
<tr>
<td>Number of 4 input LUTs</td>
<td>998</td>
<td>27392</td>
<td>3%</td>
</tr>
</tbody>
</table>

### Table 5

<table>
<thead>
<tr>
<th>Logic Utilization</th>
<th>Used</th>
<th>Available</th>
<th>Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Slices</td>
<td>309</td>
<td>13696</td>
<td>2%</td>
</tr>
</tbody>
</table>

### Table 6

<table>
<thead>
<tr>
<th>Logic Utilization</th>
<th>Used</th>
<th>Available</th>
<th>Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Slices</td>
<td>1295</td>
<td>13696</td>
<td>9%</td>
</tr>
<tr>
<td>Number of Slice Flip Flops</td>
<td>899</td>
<td>27392</td>
<td>3%</td>
</tr>
<tr>
<td>Number of 4 input LUTs</td>
<td>2319</td>
<td>27392</td>
<td>8%</td>
</tr>
</tbody>
</table>

### Table 7

<table>
<thead>
<tr>
<th>8x8 block image data to be implemented</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
</tr>
<tr>
<td>33</td>
</tr>
<tr>
<td>37</td>
</tr>
<tr>
<td>40</td>
</tr>
<tr>
<td>40</td>
</tr>
<tr>
<td>40</td>
</tr>
<tr>
<td>41</td>
</tr>
</tbody>
</table>

### Table 8

<table>
<thead>
<tr>
<th>Matlab simulation result for 8x8 image data in table 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>2934</td>
</tr>
<tr>
<td>-397</td>
</tr>
<tr>
<td>-214</td>
</tr>
<tr>
<td>-147</td>
</tr>
<tr>
<td>-104</td>
</tr>
<tr>
<td>-64</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>336</td>
</tr>
</tbody>
</table>

### Fig.5

Hardware implementation result of 8x8 image data in table 7 by ROM DA method.
VII. CONCLUSIONS

SDHT is used for image transform and image is reconstructed by taking inverse of SDHT. Reconstructed image shows that SDHT can be used in place of DCT in transform based coding without errors. Two DA approaches namely (1) memory based (2) proposed without use of memory have been used to implement both 1-D and 2-D DHT in Xilinx XC2VP30 FPGA device. Comparison results indicate the significance hardware savings in the proposed DA based implementation. Proposed technique additionally demonstrates that there is less error and faster speed when compared to memory based DA implementation without use of floating point adder. As compared to existing DCT, less computation is required for DHT.

VIII. ACKNOWLEDGMENT

The authors acknowledge to DIT (Ministry of Information & Communication Technology) for the financial support for carrying out this research work.

IX. REFERENCES


