1D Discrete Cosine Transform using Distributed Algorithm

Trinadh Balaga, B.Bhaskar Rao

Abstract: Discrete Cosine Transform (DCT), which is an important component of image and video compression, is adopted in various standardized coding schemes, such as JPEG. As the ongoing demand increases, for better compression performance of the latest video coding standard, the H.264/AVC (Advanced Video Coding) is formulated. The H.264/AVC is also known as MPEG-4. An advantage of the H.264/AVC is the simplicity of its transform. Distributed Arithmetic (DA) is an effective method for computing inner products when one of the input vectors is fixed. It uses precomputed look-up tables and accumulators instead of multipliers for calculating inner products and has been widely used in many DSP applications such as DFT, DCT, convolution, and digital filters. In particular, there has been great interest in implementing DCT with parallel distributed arithmetic and in reducing the ROM size required in the implementations since the DA-based DCT architectures are known to have very regular structures suitable for VLSI implementations. Low hardware circuit cost as well as low power consumption. Low hardware cost is achieved by exploiting redundant computational units and a technique to reduce error introduced by sign extension is also presented. The results indicate the considerable power as well as hardware savings in presented architecture.

Index Terms: Distributed Arithmetic (DA), JPEG, Discrete Cosine Transform (DCT), MPEG.

I. INTRODUCTION

The arrival of portable devices processing audio and video data endorses the need for solutions providing high-speed and low-power consumption for implementing the compute-intensive multimedia calculations. Hardwired implementations of such algorithms are not suitable, as a margin of flexibility is required due to the constantly changing algorithms and DSPs provide low-throughput and high power-consumption. In the past years reconfigurable hardware has emerged as a low-cost and flexible solution for high-throughput custom hardware at the cost of increased power consumption and area. As reported earlier in [1], a reconfigurable array specific to one type of calculation provides a good compromise between flexibility, power-consumption, area and performance when compared to DSPs, FPGAs and hardwired solutions. This paper presents a reconfigurable architecture specific to computations that can be implemented in Distributed Algorithms [2]; this includes computations such as DCT and FIR filtering used in video and audio systems. Previous domain-specific and coarse-grain reconfigurable architectures are more processor based; e.g. [3] provides simple programmable processors interconnected together for the execution of complex algorithms. In [4] the data path of a processor can be reconfigured during run-time to adapt to calculations. The architecture proposed in this paper is based on a heterogeneous array with a mesh of interconnects that is able to provide more parallel computations and a higher throughput at a lower frequency. Previous programmable and configurable architectures for DCT such as the one presented in [5] provide limited flexibility in the wide range of possible implementations that could be suitable. By using FPGA-style interconnects and elements we can provide greater flexibility at a lower-level.

The paper is organized as follows: In section 2 the algorithms to be supported are overviewed. Section 3 describes the area and power efficient architecture for 1D DCT computation, and in section 4 the implementation and comparison results. Conclusions are drawn in section V.

II. ERROR REDUCTION TECHNIQUE FOR 8X1 1D DCT

A. Computation of 2D-DCT

For a 2-D data X(p,q), 0 ≤ p ≤ 7 and 0 ≤ q ≤ 7, 8x8 2-D DCT is given by

$$F(a,b) = \frac{2}{8} C(a) C(b) \sum_{p=0}^{7} \sum_{q=0}^{7} X(p,q) \cos \left( \frac{(2+1)\pi}{16}(a+0.5) \right) \cos \left( \frac{(2+1)\pi}{16}(b+0.5) \right)$$

where 0 ≤ a ≤ 7 and 0 ≤ b ≤ 7 and $c(a), c(b) = 1/2$ for a,b=0, c(a),c(b)=1 otherwise. Implementation computation is reduced by decomposing (1) in two 8x1 1-D DCT given by,

$$F(a,b) = \frac{1}{2} C(a) \sum_{p=0}^{7} X(p) \cos \left( \frac{(2+1)\pi}{16}(a+0.5) \right)$$

For 2-D DCT computation of a 8x8 2-D data, first row-wise 8x1 1-D DCT is taken for all rows followed by column-wise 8x1 1-D DCT to all columns. Intermediate results of 1-D DCT are stored in transposition memory.

In [9], (2) can be simplified as,

$$F(0) = [X(0) + X(1) + X(2) + X(3) + X(4) + X(5) + X(6)] \cos \left( \frac{(2+1)\pi}{16}(a+0.5) \right)$$

$$F(1) = [X(0) - X(7)] A + [X(1) - X(6)] B + X(2) - X(5) + X(6)$$

$$F(2) = [X(0) - X(3) - X(4) + X(7)] M + X(1) - X(5)$$

$$F(3) = [X(0) - X(7)] B + [X(1) - X(6)] C + [X(2) - X(5)]$$

$$F(4) = [X(0) - X(1) + X(3) + X(4) - X(5) - X(6) + X(7)]$$

$$F(5) = [X(0) - X(7)] C + [X(1) - X(6)] D + [X(2) - X(5)] D + [X(3) - X(4) - X(4) + X(7)]$$

$$F(6) = [X(0) - X(3) - X(6)]$$
1D Discrete Cosine Transform Using Distributed Algorithm

\[ X(4) + X(7) \] \[ + \] \[ X(1) - X(2) - X(5) + X(6) \] \[ (-M) \]

\[ F(7) = [X(0) - X(7)]D + [X(1) - X(6)](-C) + \]
\[ [X(2) - X(5)]B + [X(3) - X(4)](-A) \] \[ (3h) \]

Where

\[ M = \frac{1}{2} \cos \frac{\pi}{8}, \]
\[ N = \frac{1}{2} \cos \frac{3\pi}{16}, \]
\[ I = \frac{1}{2} \cos \frac{\pi}{4}, \]
\[ A = \frac{1}{2} \cos \frac{\pi}{8}, \]
\[ B = \frac{1}{2} \cos \frac{3\pi}{16}, \]
\[ C = \frac{1}{2} \cos \frac{5\pi}{16}, D = \frac{1}{2} \cos \frac{7\pi}{16}. \]

Distributed arithmetic is used to compute the 8 equations above where cosine terms are expressed in DA form. Implementation is realized by using shift and adds operations. Shifting operation is performed by wiring. Each value of \( F \) is computed in parallel and hence faster speed is achieved \([11]\). Shifted data are represented by less number of bits and hence adder bit-width is reduced resulting in less hardware cost. For DCT computation image data is represented in signed 2’s complement form range -128 to 127. Bit width of shifted data is determined by number of times shift operation is done. So different bit-width intermediate data are present which are to be added. For 2-input adder both input data width has to be equal and hence sign extension is done in smaller bit-width data. Shifting and addition with sign extension creates error. For example if initial value is -2, in 8-bit 2’s complement representation this can be written 11111110. If we take 4 shifted sample of this value \( a = 1111111 (\text{shifting 2 times}), b = 111111 (\text{shifting 4 times}), c = 111111 (\text{shifting 5 times}), d = 111111 (\text{shifting 6 times}) \) all are -1. But all these data should be zero. If we add these values in cascade, result we will have -4 (which should be zero). To overcome this problem we have realized adder as shown in \( \text{fig}.1 \). If one of the input is -1

\[ a0 \]
\[ a1 \]
\[ a2 \]
\[ a3 \]
\[ a4 \]
\[ a5 \]
\[ a6 \]
\[ a7 \]
\[ b0 \]
\[ b1 \]
\[ b2 \]
\[ b3 \]
\[ b4 \]
\[ b5 \]
\[ b6 \]
\[ b7 \]

\[ Y(1\text{n+1 bit}) \]
\[ Y(2\text{n+1 bit}) \]

(c) Final sum is from MUX1 or MUX2 output.

then output is other one. To verify this VHDL implementation of 1-D DCT and simulation is done using Xilinx ISE simulator for 8x1 data matrix of \( X \) given by, \( X = \{60, 40, 25, 55, 40, 42, 82, 84\} \)

Matlab simulation result for 1-D DCT gives, \( Y = \{151.3209 -32.4895 33.1588 -1.7108 17.6777 18.5074 -16.0309 -5.0975\} \)

Fig.2 shows the result of Xilinx ISE simulator using simple adder and proposed adder scheme. It is evident with matlab comparison that error due to sign extension is less in proposed adder scheme.

**III. PROPOSED ARCHITECTURE OF 8X1 1D DCT**

Instead of computing \( F(0) \) to \( F(7) \) in parallel process, they can be computed in pipeline process and there are only seven osine terms which can be seen from equation 3a to 3h. Hence we can implement a Distributed Arithmetic based module which gives the multiplication and accumulation results by taking the inputs.

Let us consider,

\[ a1 = X(0) + X(1) + X(2) + X(3) + X(4) + X(5) + X(6) + X(7), \]
\[ a2 = X(0) - X(1) - X(2) + X(3) + X(4) + X(5) - X(6) + X(7), \]
\[ b1 = X(0) - X(7), b2 = X(1) - X(6), b3 = X(2) - X(5), \]
\[ b4 = X(3) - X(4), c1 = X(0) - X(3) - X(4) + X(7) \] and \( c2 = X(1) - X(2) - X(5) + X(6). \)

| a0 | 60 |
| a1 | 40 |
| a2 | 25 |
| a3 | 55 |
| a4 | 40 |
| a5 | 42 |
| a6 | 82 |
| a7 | 84 |
| b0[10:0] | 149 |
| b1[10:0] | -36 |
| b2[10:0] | 31 |
| b3[10:0] | -5 |
| b4[10:0] | 16 |
| b5[10:0] | 14 |
| b6[10:0] | -18 |
| b7[10:0] | -11 |

**Fig 2:** Simulation result of 1D DCT architecture using Xilinx ISE simulator (a) Addition Operation (b) Proposed adder.
Then from (3a) to (3h), \( F(0) = a_1xP \), \( F(4) = a_2xP \),
\( F(1) = b_1xA + b_2xB + b_3xC + b_4xD \),
\( F(3) = b_1xB - b_2xD - b_3xA - b_4xC \),
\( F(5) = b_1xC - b_2xA + b_3xD + b_4xB \),
\( F(7) = b_1xD - b_2xC + b_3xB - b_4xA \),
\( F(2) = c_1xM + c_2xN \), and \( F(6) = c_1xN - c_2Xm \).

Expressed in DA form with four inputs and for Module3
\((1/2)\cos(\pi/8)\) and \((1/2)\cos(3\pi/8)\) is expressed in DA form
with two inputs. Table I, II and III show the inputs given to
these module on clock cycle basis provided by timing and
control unit. Outputs obtained are stored in 8 registers.

**Table I: Pipeline Computation Of DCT Coefficients F(0) And F(4)**

<table>
<thead>
<tr>
<th>MODULE 1</th>
<th>CLOCK CYCLE 1</th>
<th>CLOCK CYCLE 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>a1</td>
<td>a2</td>
</tr>
<tr>
<td>Output</td>
<td>F(0)</td>
<td>F(4)</td>
</tr>
</tbody>
</table>

**Table II: Pipeline Computation Of DCT Coefficients F(1), F(3), F(5), F(7)**

<table>
<thead>
<tr>
<th>MODULE 2</th>
<th>CLOCK CYCLE 1</th>
<th>CLOCK CYCLE 2</th>
<th>CLOCK CYCLE 3</th>
<th>CLOCK CYCLE 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input 1</td>
<td>b1</td>
<td>-b3</td>
<td>-b2</td>
<td>-b4</td>
</tr>
<tr>
<td>Input 2</td>
<td>b2</td>
<td>b1</td>
<td>b4</td>
<td>b3</td>
</tr>
<tr>
<td>Input 3</td>
<td>b3</td>
<td>-b4</td>
<td>b1</td>
<td>-b2</td>
</tr>
<tr>
<td>Input 4</td>
<td>b4</td>
<td>-b2</td>
<td>b3</td>
<td>b1</td>
</tr>
<tr>
<td>Output</td>
<td>F(1)</td>
<td>F(3)</td>
<td>F(5)</td>
<td>F(7)</td>
</tr>
</tbody>
</table>

**Table III: Pipeline Computation Of DCT Coefficients F(2) And F(6)**

<table>
<thead>
<tr>
<th>MODULE 3</th>
<th>CLOCK CYCLE 1</th>
<th>CLOCK CYCLE 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input 1</td>
<td>c1</td>
<td>c2</td>
</tr>
<tr>
<td>Input 2</td>
<td>c2</td>
<td>c1</td>
</tr>
<tr>
<td>Output</td>
<td>F(2)</td>
<td>F(6)</td>
</tr>
</tbody>
</table>

**IV. IMPLEMENTATION RESULTS AND COMPARISONS**

VHDL code is written for the implementation of 1-D
DCT architecture in (9) and proposed architecture. Adders
used in both the implementations are proposed scheme (as
explained in section II). Registers are added at the output
of 1-D DCT architecture in (9). Code is synthesized in Xilinx
xc2vp30 FPGA device. Table IV shows the device utilization
summary for the FPGA implementation. From the FPGA
implementation comparison results, it is evident that the
proposed architecture is efficient in terms of area (FPGA
resources in case of FPGA implementation) and power.

**Table IV: Device Utilization for the FPGA IMPLEMENTATION**

<table>
<thead>
<tr>
<th>FPGA CHIP XILINX XC2VP30</th>
<th>1-DDCT architecture in (9)</th>
<th>Proposed 1-D DCT architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td># of 4 input LUTs</td>
<td>1268</td>
<td>696</td>
</tr>
<tr>
<td># of slices</td>
<td>694</td>
<td>370</td>
</tr>
<tr>
<td># of slice Flip Flops</td>
<td>0</td>
<td>97</td>
</tr>
<tr>
<td># of IOB Flip Flops</td>
<td>88</td>
<td>0</td>
</tr>
<tr>
<td>Min. Period (ns)</td>
<td>32.6</td>
<td>16.29</td>
</tr>
<tr>
<td>Power (W)</td>
<td>13.1</td>
<td>2.06</td>
</tr>
</tbody>
</table>
8x8 2-D DCT is implemented using row column decomposition technique. Table V shows FPGA implementation device utilization summary.

### Table V Device Utilization Summary for 2-D DCT Implementation Using Row-Column Decomposition Technique of Proposed 1-D DCT Architecture

<table>
<thead>
<tr>
<th>FPGA-chip</th>
<th>Xilinx XC2VP30</th>
</tr>
</thead>
<tbody>
<tr>
<td># of input LUTs</td>
<td>2522</td>
</tr>
<tr>
<td># of slices</td>
<td>1701</td>
</tr>
<tr>
<td># of slice Flip Flops</td>
<td>1025</td>
</tr>
<tr>
<td>Max. Freq.(MHz)</td>
<td>45.173</td>
</tr>
<tr>
<td>Power (W)</td>
<td>0.751</td>
</tr>
</tbody>
</table>

**V. CONCLUSION**

We have proposed an area and power efficient architecture for the computation of 1-D DCT using ROM free DA is implemented in Xilinx FPGA. Comparison with recently reported DA based 1-D DCT, a significant area reduction as well as low power consumption is achieved in proposed architecture. With proposed 1-D DCT architecture, 2-D DCT is implemented using row column decomposition technique and area and power results are tabulated. An adder for the error reduction introduced due to shift and adds with less number of bit representation is also proposed and comparative HDL simulation is done for the accuracy. Proposed adder shows less error as compared to conventional adder.

**REFERENCES**


**AUTHOR’S PROFILE:**

Trinadh Balaga is pursuing his M.Tech in St. Theresa Institute of Engineering and Technology under the guidance of Mr. B. Bhaskar Rao, Assoc. Professor, HOD, and Department of ECE. My research interests include VLSI design and digital signal processing.

Sri. B. Bhaskar Rao working as Assoc. Professor and Head, Department of ECE at St. Theresa Institute of Engineering and Technology, Vizianagaram. His research interests include VLSI design and digital signal processing.