Area and Delay Optimized Arithmetic Architectures of VLSI for ACT Design

Aithi Vamsi Krishna¹, Niddana Prabha²

¹,² Electronics and Communication Engineering Department,
¹² PYDAH College Of Engineering and Technology, Visakhapatnam, A.P., India

Abstract:
The discrete cosine transform (DCT) is a widely-used and important signal processing tool employed in a plethora of applications. Typical fast algorithms for nearly-exact computation of DCT require floating point arithmetic, are multiplier intensive, and accumulate round-off errors. Recently proposed fast algorithm arithmetic cosine transform (ACT) calculates the DCT exactly using only additions and integer constant multiplications, with very low area complexity, for null mean input sequences. The ACT can also be computed non-exactly for any input sequence, with low area complexity and low power consumption, utilizing the novel architecture described. However, as a trade-off, the ACT algorithm requires 10 non-uniformly sampled data points to calculate the eight-point DCT. This requirement can easily be satisfied for applications dealing with spatial signals such as image sensors and biomedical sensor arrays, by placing sensor elements in a non-uniform grid. In this work, a hardware architecture for the computation of the null mean ACT is proposed, followed by a novel architecture that extend the ACT for non-null mean signals. All circuits are implemented and tested and synthesized using the Xilinx ISE14.7.

Keywords: DCT, ACT, EIGHT-POINT DCT, AFT,

I. INTRODUCTION

The discrete cosine transform (DCT) was first proposed by Ahmed et al. in 1974 and published in IEEE Transactions on Computers [1]. It has since attracted much attention in the computer engineering community [2], [3], [4], [5]. In particular, the eight-point DCT and its variants, in the form of fast algorithms, has been widely adopted in several image and video coding standards [6] such as JPEG, MPEG 1/2, and H.261-5 [7]. Some applications which use image and video compression include automatic surveillance [8], geospatial remote sensing [9], traffic cameras [10], homeland security [11], satellite based imaging [12], unmanned aerial vehicles [13], automotive [14], multimedia wireless sensor networks [15], the solution of partial differential equations [16] etc. A particular class of fast algorithms is constituted by the arithmetic transforms. An arithmetic transform is an algorithm for low-complexity computation of a given trigonometric transform, based on number-theoretic results. A prominent example is the arithmetic Fourier transform (AFT) proposed by Reed et al. [17], [18]. The AFT allows multiplication-free calculation of Fourier coefficients using number-theoretic methods and non-uniformly sampled inputs. A feature of the AFT is its suitability for parallel implementation [17], [18]. Recently, an arithmetic transform method for the computation of the DCT, called the arithmetic cosine transform (ACT) was proposed in [19]. The ACT can provide a multiplication-free framework and leads to the exact computation of the DCT—provided that the input signal has null-mean and is non-uniformly sampled [19]. The computational gains of the ACT are only possible when its prescribed nonuniformly sampled data is available. Classically the required non-uniform samples are derived by means of interpolation over uniformly sampled data [19]. Such interpolation implies a computational overhead. Another aspect of the ACT is that, for arbitrary input signal, it requires the computation of the input signal mean value [19]. Usually, the mean value is computed from uniformly sampled data [19]. In fact, this dependence on uniformly sampled data has been precluding the implementation of the ACT based exclusively on non-uniformly sampled data. On the other hand, the requirement for non-uniform samples can be satisfied when spatial input signals are considered. In spatial signal processing, non-uniformly sampled signals can be directly obtained, without interpolation using a non-uniform placement of sensors [7], [20]. This motivates the search for architectures which could solely rely on non-uniformly sampled inputs. In this paper we address two main problems: (i) the proposition of a method to obtain the mean value of a given input signal from its non-uniform samples as prescribed by the ACT and (ii) the introduction of efficient architectures for calculation of the eight-point DCT based on the ACT, operating on non-uniformly sampled data only. This leads to designs with...
low computational complexity. Having ACT architectures that compute 1-D DCT can be utilized as a building block to implement such 2-D DCT architectures that take inputs from sensors placed on a non uniform grid. Two architectures based on the ACT are sought, being referred to as Architectures I and II. Architecture I provides the hardware implementation of the ACT algorithm proposed in [19], and calculates the DCT with exact precision for null mean eight-point sequences. The proposed Architecture I is designed to require only additions and multiplications by integers. Thus, no source of intrinsic computation error is present, such as rounding-off and truncation. Therefore, area consuming hardware multipliers are not necessary. We propose Architecture II that implements the novel modified ACT algorithm for DCT calculation of arbitrary, non-null-mean input signals, using 11 hardware multiplications. Both architectures require only non-uniformly sampled inputs.

II. THE ARITHMETIC COSINE TRANSFORM

The usual input sequence to the DCT can be considered as uniform samples of a continuous input signal \( v(t) \). This results in an N-point column vector \( \mathbf{v} = \{v_n\}_{n=0}^{N-1} \) which has its DCT denoted by the N-point column vector \( \mathbf{V} = \{V_k\}_{k=0}^{N-1} \). To calculate \( \mathbf{V} \), the ACT algorithm requires non-uniformly sampled points of the continuous input signal \( v(t) \) [19]. These points are given by

\[
r = \frac{2mN}{k} - \frac{1}{2},
\]

where \( k \) runs from 1 to \( N-1 \), and \( m \) runs from 0 to \( \frac{N}{2} \). We can define the set \( R \) as

\[
R = \{ \text{Set of all values of } r \}
\]

It is important to notice that the values of \( r \) are not necessarily integer. In fact, they are expected to be fractional. If the signal of interest has zero mean, then the ACT algorithm can be used to calculate the DCT coefficients as follows. First, let the ACT averages \( S_k \), \( k = 1/2; \ldots; N-1 \), of the non-uniform sampled inputs be defined as [19]:

\[
S_k = \frac{1}{k} \sum_{m=0}^{k-1} v_{2m+k}, \quad k = 1, 2, \ldots, N-1.
\]

The ACT averages can be employed to compute DCT coefficients according to [19]:

\[
V_k = \sqrt{\frac{N}{2}} \sum_{l=1}^{\frac{N}{2}} \mu(l) \cdot S_{kl}, \quad k = 1, 2, \ldots, N-1.
\]

where \( \mu \) is the Möbius function. The derivation of the ACT [19] utilizes the Möbius inversion formula. Because the Möbius function values are limited to 1, 0, 1, 3 results in no additional multiplicative complexity. In actice, input sequences are not always null mean, therefore a correction term is necessary to (3). In [19] an expression suitable for the non-null mean signals is given as:

\[
V_k = \sqrt{\frac{N}{2}} \sum_{l=1}^{\frac{N}{2}} \mu(l) \cdot S_{kl} - \sqrt{\frac{N}{2}} \cdot M\left(\frac{N-1}{k}\right),
\]

where \( M(n) \) is the Möbius function [19] and \( v \) is the mean value of the uniformly sampled input sequence.

III. EXISTING ALGORITHM

3.1 Mean Value Calculation

Although (4) leads to the DCT coefficients of non-null mean input signals, it requires the knowledge of quantity \( v \), which could be calculated straightforwardly from the N uniform samples in \( v \). Since uniform samples are not available, \( v \) should be directly calculated from non-uniform samples. The non-uniform samples are related to the uniform samples according to the interpolation scheme given by [19]:

\[
v_r = \sum_{n=0}^{N-1} w_n(r) \cdot v_n, \quad r \in R,
\]

where \( w_n(r) \) is the interpolation weight function expressed by:

\[
w_n(r) = \frac{1}{2N} \left[ D_{N-1} \left( \frac{\pi}{N} (n + r + 1) \right) + D_{N-1} \left( \frac{\pi}{N} (n - r) \right) \right], \quad n = 0, 1, \ldots, N-1,
\]

And

\[
D_N(x) = \frac{\sin ((N + 1/2)x)}{\sin (x/2)}
\]

denotes the Dirichlet kernel [21, p. 312]. Here, the set \( R \) is defined in (1). More compactly, (5) can be put in matrix form. Indeed, we can write \( v_r \) as \( Wv \), where \( W \) is a column vector containing the required non-uniform samples,

\[
W = \{w_n(r)\}, \quad n = 0, 1, \ldots, N-1, \quad r \in R,
\]

is the implied interpolation matrix. For the particular case of the eight-point ACT, the following 10 non-uniform sampling instants are required.

\[
r \in R = \left\{ \frac{1}{2}, \frac{1}{4}, \frac{3}{2}, \frac{5}{4}, \frac{7}{2}, \frac{9}{4}, \frac{11}{2}, \frac{13}{4}, \frac{15}{2} \right\}.
\]

Moreover, matrix \( W \) is found to possess full column rank. Thus, its Moore-Penrose pseudo-inverse \( W^+ \) is the left inverse of \( W[22, p. 93] \). Therefore, we obtain

\[
\mathbf{v} = W^+ \cdot \mathbf{v}_r.
\]
Consequently, the mean value of \( v \) can be determined exclusively from the non-uniform samples, according to

\[
\bar{v} = \frac{1}{8} W \cdot v_r,
\]

where \( w \) is the eight-point vector of the sums of each column of \( W \). Scaled vector \( w = 8 \) has constant elements given by

\[
\frac{1}{8} w^T = \begin{bmatrix}
0.131763492716950 \\
0.498388117552161 \\
-0.313306526814540 \\
0.018837637958148 \\
0.389746948996966 \\
-0.178465262210960 \\
0.166302458810496 \\
0.269801852271683 \\
-0.131541981375149 \\
0.148473262094246
\end{bmatrix},
\]

were the superscript \( ^T \) denotes the transposition operation.

**Matrix Factorization of ACT**

In view of (2) and (7), (4) can be interpreted as the sought relation between \( V \) and \( v_r \). Thus, we can consider a Transformation matrix \( T \) relating these two vectors. Notice that \( T \) is not a square matrix. Since \( k = 1, 2, \ldots N - 1 \), the size of \( T \) is \( 8N - 1 \times 1 \), where \( jR \) is the number of elements of \( R \). This transformation matrix returns all the DCT components, except the zeroth one, according to

\[
\begin{bmatrix}
V_1 \\
V_2 \\
\vdots \\
V_{N-1}
\end{bmatrix}^T = T \cdot v_r.
\]

Notice that \( V_0 = \sqrt{N} \cdot \bar{v} \).

For \( N = 8 \), matrix \( T \) has size \( 7 \times 10 \) and admits the following matrix factorization:

\[
T = 2 \cdot M_0 \cdot D_1 \cdot S + M_0 \cdot W^+,
\]

where \( D_1 = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \frac{1}{3} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{1}{4} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{1}{5} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \frac{1}{6} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{7}
\end{bmatrix},
\]

\[
S = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 2 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \\
1 & 2 & 0 & 0 & 2 & 0 & 0 & 2 & 0 & 0
\end{bmatrix},
\]

\[
M_0 = \begin{bmatrix}
1 & -1 & -1 & 0 & -1 & 1 & -1 \\
0 & 1 & 0 & -1 & 0 & -1 & 0 \\
0 & 0 & 1 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix},
\]

Existing architectures

In this section, above discussed methods are employed to furnish two novel low complexity architectures, which take only non-uniformly sampled inputs. Integer multiplications, which are exact in nature, are realized using simple shift-add structures. The designs are fully pipelined by judicious sertion of registers at internal nodes, leading to low critical path delay at the cost of latency.

**VLSI Architectures:**

**Architecture I**

**Multiple by 2 Design using Shifter:**

Arithmetic Cosine Transform require multiple by two architectures, for this multiplier designed using shift register by 1 shift. By one left shift of shift register we can get the multiple by 2 design.

**Fig 1:** Vedic multiplier architecture
Architecture 2: The radix-8 booth technique:

![Fig 2: The radix-8 Booth encoded modulo multiplier](image)

To ensure that the radix-8 Booth encoded modulo multiplier does not constitute the system critical path of a high-DR moduli set based RNS multiplier, the carry propagation length in the hard multiple generation should not exceed n bits. To this end, the carry propagation through the HAs in Fig. 2 can be eliminated by making the end-around-carry bit $c_7$ a partial product bit to be accumulated in the CSA tree. This technique reduces the carry propagation length to n bits by representing the hard multiple as a sum and a redundant end-around-carry bit pair. The resultant $|n/3| + 1$ end-around-carry bits in the partial product matrix may lead to a marginal increase in the CSA tree depth and consequently, may aggravate the delay of the CSA tree. In which case, it is not sufficient to reduce the carry propagation length to merely bits using the above technique.

Since the absolute difference between the noncritical modulo $2^n-1$ multiplier delay and the system critical path delay depends on the degree of imbalance in the moduli word-length of a RNS, the delays cannot be equalized by arbitrarily fixing the carry propagation length to n bits. Instead, we propose to accomplish the adaptive delay equalization by representing the hard multiple in a partially-redundant

Architecture 3: Vedic Technique:

Multiplier is the important block in almost all the arithmetic logic units. These multipliers are mostly used in the fields of the Digital Signal Processing (DSP), Fast Fourier Transform, convolution, filtering and microprocessor applications. Since multiplier is the main component and hence a high speed and area efficient multiplier is needed to achieve this one of the finest technique is by using Vedic mathematics. The mathematical techniques like Vedic mathematics used to reduce the time for the processor such that it can increases speed and also takes only few hardware elements. Vedic is a word obtained from the word “Veda” and its meaning is “store house of all knowledge” Vedic mathematics consisting of the 16 sutras which it can be related to the different branches of mathematics like algebra, arithmetic geometry.

Architecture 4: The ACT:

The ACT expressions for null mean signals in (2) and (3) can be implemented for N = 8 as shown in . We refer to this design as Architecture I. The eight-point null mean ACT block admits the 10 non-uniformly sampled inputs corresponding to (6). Constant multiplications by two are implemented as left-shift operations; and the fractional constant multipliers $1/2, 1/3, 1/4, \ldots$ are converted to integers by scaling them by the least common multiple of their denominators: 420. The integer constant multipliers can be implemented as Booth encoded shift-and-add structures making the architecture multiplier free, and the outputs of the block are scaled by 420 \( \sqrt{2/N} = 211 \). This architecture is useful in applications that have null mean input sequences, and can be implemented with very low computational and area complexity.
Architecture II

The proposed method in Section 3 for the computation of \( v \) from the non-uniformly sampled 10-point signal can be implemented as shown in Fig. 6. We will refer to it as the mean calculation block, which computes (7). The correction term associated to the Mertens function required in (4) is shown in Fig. 2a. A combination of (i) this particular block, (ii) the Architecture I block, and (iii) the mean calculation block yields the proposed Architecture. Note that calculation of the DCT coefficients using the null mean ACT block can also be achieved by subtracting the mean \( v \) from its inputs. However, Architecture II has a lower computational complexity when compared to such alternative. Computation complexity of both Architecture I and Architecture II are listed in Table 1 in terms of constant multipliers and two-input adders. Integer constant multiplications are implemented as shift-and-add structures, therefore are not counted as multipliers. Note that the adder count also include the adders required for the Booth encoded structures.

IV. PROPOSED ALGORITHM

Ladner-Fischer adder:

Ladner-Fischer parallel prefix adder was developed by R. Ladner and M. Fischer in 1980. Ladner-Fischer prefix tree is a structure that sits between Brent-Kung and Sklansky prefix tree. The LF adder has minimum logic depth but it has large fan-out. Ladner-Fischer adder has carry operator nodes. The delay for the type of Ladner-Fischer prefix tree is \( \log_2 n + I \). Fig. shows the 16-bit LF adder.

Parallel-Prefix adders:

Parallel-Prefix adders perform parallel addition i.e. most important in microprocessors, DSPs, mobile devices and other high speed applications. Parallel-Prefix adder reduces logic complexity and delay thereby enhancing performance with factors like area and power. Therefore the Parallel-Prefix adders are requisite element in the high speed arithmetic circuits and popular since twenty years. Parallel prefix computation carries out three necessary or vital steps:

1) Computation of carry generation & carry propagation signals by using no. of input bits.
2) Calculating all the carry signals in parallel that is called prefix computation.
3) Evaluating total sum of given inputs.

These steps are given in fig. that is given above.
Previous carry is calculated to the next bit is called propagate signal and generate is to generate the carry bit below are the signals:

\[ G_i = A_i \cdot B_i \] ........................ (1)  

\[ P_i = A_i \oplus B_i \] ........................ (2)  

2. Calculation of all carry signals:  
\[ G_{i:j} = G_{i:k} + P_{i:k} \cdot G_{k-1:j} \] ........................ (3)  

\[ P_{i:j} = P_{i:k} \cdot P_{k-1:j} \] ........................ (4)  

3. Calculation of Final Sum:  
\[ S_i = P_i \oplus G_{i-1:0} \] ........................ (5)  

Fig. 6. (a) Null mean ACT and (b) mean calculation block

Fig 7: Black, Grey cells, Buffer and there Schematics

V. IMPLEMENTATION AND RESULTS

We implemented Ladner-Fischer parallel prefix adder architectures described in the previous section. Ladner-Fischer prefix tree is a structure that sits between Brent-
Kung and Sklansky prefix tree. The LF adder has minimum logic depth but it has large fan-out. Ladner-Fischer adder has carry operator nodes. The delay for the type of Ladner-Fischer prefix tree is $log_{2}n+1$. Fig. shows the 16-bit LF adder.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_{0}$</td>
<td>000a</td>
</tr>
<tr>
<td>$a_{1}$</td>
<td>001a</td>
</tr>
<tr>
<td>$a_{2}$</td>
<td>010a</td>
</tr>
<tr>
<td>$a_{3}$</td>
<td>011a</td>
</tr>
<tr>
<td>$a_{4}$</td>
<td>100a</td>
</tr>
<tr>
<td>$a_{5}$</td>
<td>101a</td>
</tr>
<tr>
<td>$a_{6}$</td>
<td>110a</td>
</tr>
<tr>
<td>$a_{7}$</td>
<td>111a</td>
</tr>
</tbody>
</table>

Fig 8. Simulation results of ACT

Parallel-Prefix adders perform parallel addition i.e. most important in microprocessors, DSPs, mobile devices and other high speed applications. Parallel-Prefix adder reduces logic complexity and delay thereby enhancing performance with factors like area and power. Therefore the Parallel-Prefix adders are requisite element in the high speed arithmetic circuits.

VI. CONCLUSION

The ACT estimation is fitting for figuring the eight point DCT coefficients decisively using just adders and entire number consistent increments, moreover with low computational capriciousness. ACT models for invalid mean commitments and furthermore for non-invalid mean data sources are proposed, executed and attempted on XilinxISE14.7. The ordinary rate botch and PSNR were gotten as figures of authenticity to review the think results. Comes to fruition show that despite for lower settled point word-lengths, the executions provoke acceptable squirm room. The advantage use for various settled point executions exhibits a trade off among precision and device resources (chip area, speed, and power). It is the underlying move towards new research on low power and low multifaceted nature computation of the DCT by strategies for the starting late proposed ACT.

REFERENCES


