EFFICIENT IMPLEMENTATION OF DISCRETE COSINE TRANSFORM USING ADDER-BASED DISTRIBUTED ARITHMETIC

Ajit Kumar Patro[1], Dr. Sudhansu Sekhar Nayak[2]
[1]Research Scholar, Centurion University of Technology and Management, Paralakhemundi, Odisha, India
[2]Professor, Centurion University of Technology and Management, Paralakhemundi, Odisha, India

Abstract *- Distributed Arithmetic (DA) has been widely used in inner product calculation with one fixed vector. In ROM-based DA, the variable input is decomposed into bit level but in adder-based DA the fixed vector is decomposed into bit level instead of the variable input. The adder-based DA algorithm takes advantage of shared partial sum-of-products in the inner product calculation and nonzero bits in the fixed input to achieve high speed and low area design. 8-point Discrete Cosine Transform (DCT) architecture using adder-based DA needs 30% of ROM area and has faster speed compared with conventional ROM-based DA.*

Keywords *- VLSI, Distributed Arithmetic, Discrete Cosine Transform, Look-up Table.*

I. INTRODUCTION

In digital signal processing the computation of the sum-of-products or the inner product of two vectors involves more hardware cost. The inner product is calculated fast by multiplier and accumulator based designs but these designs have high cost when size of the inner product is large. Use of distributed arithmetic (DA) [1, 2] efficiently reduces the number of operations. DA uses pre-computed sum-of-products rather than computing them at the run time. So DA is popular in various digital signal processing applications [2]. The conventional ROM-based DA [2] decomposes the variable input of the inner products into bit level to efficiently generate pre-computed date required. The ROM-based DA uses ROM table to store the pre-computed data which makes it regular and attractive. However, when the size of inner product increases, the ROM area in ROM-based DA increases exponentially and becomes very large [8]. M. T. Sun et al. [5] and S. Uramoto et al. [6] have presented regular and efficient IC realisation for discrete cosine transform (DCT) by concurrent architectures using DA and memory oriented structure. The ROM size of these architectures increases rapidly with the order of DCT so that the architectures may be useful for implementing DCT of lower order only.

In this paper, we present an adder-based DA [7] for computation of the inner product. Adder-based DA decomposes the fixed coefficients into bit level instead of decomposing the variable input and distributes the multiplication operation. It exploits the distribution of binary value pattern and maximises the hardware sharing possibility in the implementation. So, adder-based DA requires less area and smaller delay than ROM-based DA.

In Section 2, we have presented the algorithms of the distributed arithmetic. The algorithm and architecture bit level implementation of DCT is presented in Section 3. Section 4 deals with the conclusion.

II. ALGORITHMS OF THE DISTRIBUTED ARITHMETIC

Let us consider the inner product of size L

$$X = \sum_{i=1}^{L} C_i x_i \qquad (1)$$

where C_i is fixed coefficient and x_i is variable input data.

C_i and x_i are expressed as:

$$C_i = \sum_{j=1}^{M} C_{i,j} 2^{-j} \qquad (2)$$

and

$$x_i = \sum_{k=1}^{N} x_{i,k} 2^{-k} \qquad (3)$$

where *M* is the word length of *C*$_i$ and *N* is the word length of *x*$_i$.

1. ROM-based Distributed Arithmetic

ROM-based DA decomposes the variable input x_i into bit level. Substituting equation (3) in equation (1), we get

$$X = \sum_{i=1}^{L} C_i x_i = \sum_{i=1}^{L} C_i \left(\sum_{k=1}^{N} x_{i,k} 2^{-k} \right) = \sum_{k=1}^{N} \left(\sum_{i=1}^{L} C_i \, x_{i,k} \right) 2^{-k} \qquad (4)$$

ROM-based DA pre-computes these partial products $\sum_{i=1}^{L} C_i \, x_{i,k}$ and stores them in a look-up table ROM for all possible $x_{i,k}$ values. The pre-computed values are addressed by the input data and accumulated by shift-and-add. The inner product operation is replaced by a ROM table and shift-adder, shown in Fig. 1.

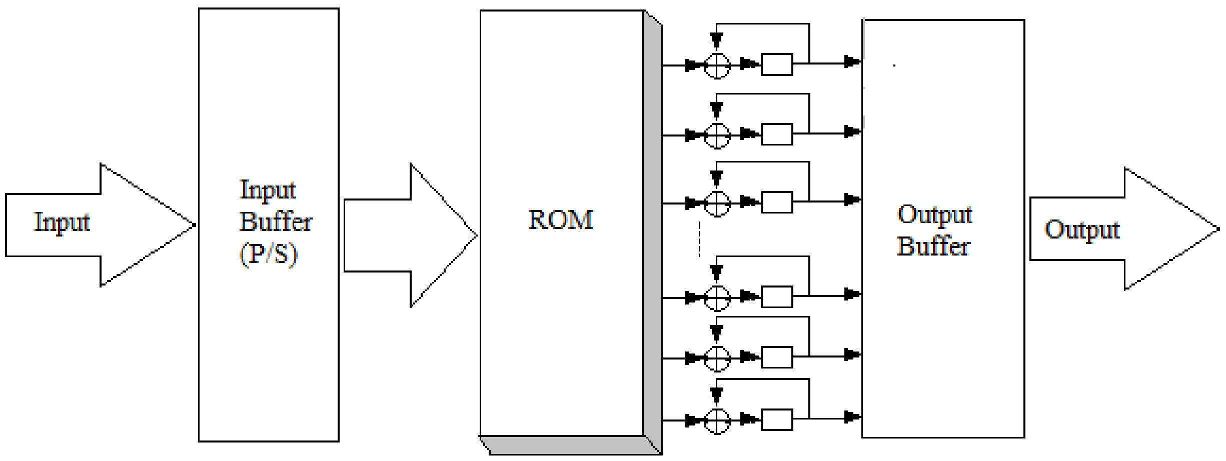


Fig. 1: ROM-based DA Architecture

2. Adder-based Distributed Arithmetic

Adder-based DA decomposes the fixed coefficients *C*$_i$ instead of variable input *x*$_i$ into bit level. Substituting equation (2) in equation (1), we obtain

$$X = \sum_{i=1}^{L} C_i x_i = \sum_{i=1}^{L} \left(\sum_{j=1}^{M} C_{i,j} 2^{-j} \right) x_i = \sum_{j=1}^{M} \left(\sum_{i=1}^{L} C_{i,j} \, x_i \right) 2^{-j} \qquad (5)$$

Multiplication operations are distributed into $\sum_{i=1}^{L} C_{i,j} \, x_i$.

Since $C_{i,j}$ is either 0 or 1, $\sum_{i=1}^{L} C_{i,j} \, x_i = S_j$ is a combination of *x*$_i$.

If *x*$_i$ is serially input, we can obtain *S*$_j$ bit by bit via serial adder's i.e.

$$S_j = \sum_{t=1}^{P} S_{j,t} 2^{-t} \qquad (6)$$

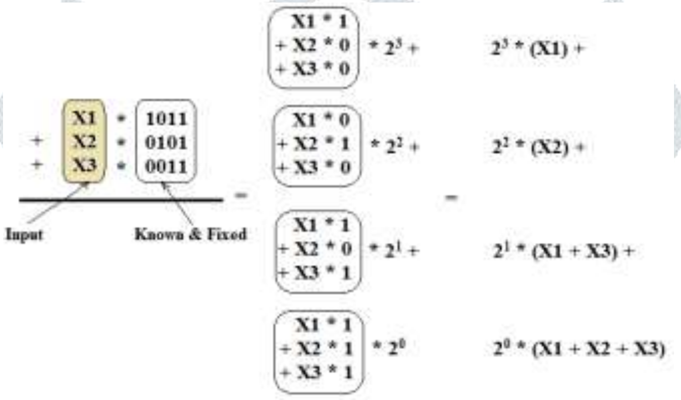
So equation (5) can be written as:

$$X = \sum_{j=1}^{M} S_j 2^{-j} = \sum_{j=1}^{M}\left(\sum_{t=1}^{P} S_{j,t}2^{-t}\right)2^{-j} = \sum_{t=1}^{P}\left(\sum_{j=1}^{M} S_{j,t}2^{-j}\right)2^{-t} \qquad (7)$$

Thus, we can accumulate and shift the term $\sum_{j=1}^{M} S_{j,t}2^{-j}$ at each cycle t to obtain the inner product. Since S_j is obtained by adders, the DA algorithm is called adder-based DA. The adder-based DA has two representations given by equations (5) and (7). Equation (5) uses bit parallel input x_i and equation (7) uses bit serial input of x_i.

3. An Example of Adder-based DA

Fig. 2 illustrates how adder-based DA works. Fixed coefficients are first decomposed into bit level. We find that addition occurs only when the bits of fixed coefficients are not zero. This is called zero-one pattern property. In this example, we only need to perform ($X_1 + X_2$) and ($X_1 + X_2 + X_3$). In addition, summation term ($X_1 + X_2$) can be shared between bit weights 2^1 and 2^0. This is called common term sharing property. We can save area using these two properties. The summation network of the example is shown in Fig. 3. The adder in Fig. 3 is a bit-serial adder that follows equation (7). We only connect the required term to the respective bit to reduce unnecessary routing.



X1 * 1011
+ X2 * 0101
+ X3 * 0011
Input
Known & Fixed
X1 * 1 + X2 * 0 + X3 * 0 * 2³ +
X1 * 0 + X2 * 1 + X3 * 0 * 2² +
X1 * 1 + X2 * 0 + X3 * 1 * 2¹ +
X1 * 1 + X2 * 1 + X3 * 1 * 2⁰
2³ * (X1) +
2² * (X2) +
2¹ * (X1 + X3) +
2⁰ * (X1 + X2 + X3)

Fig. 2: An Example of Adder-based DA

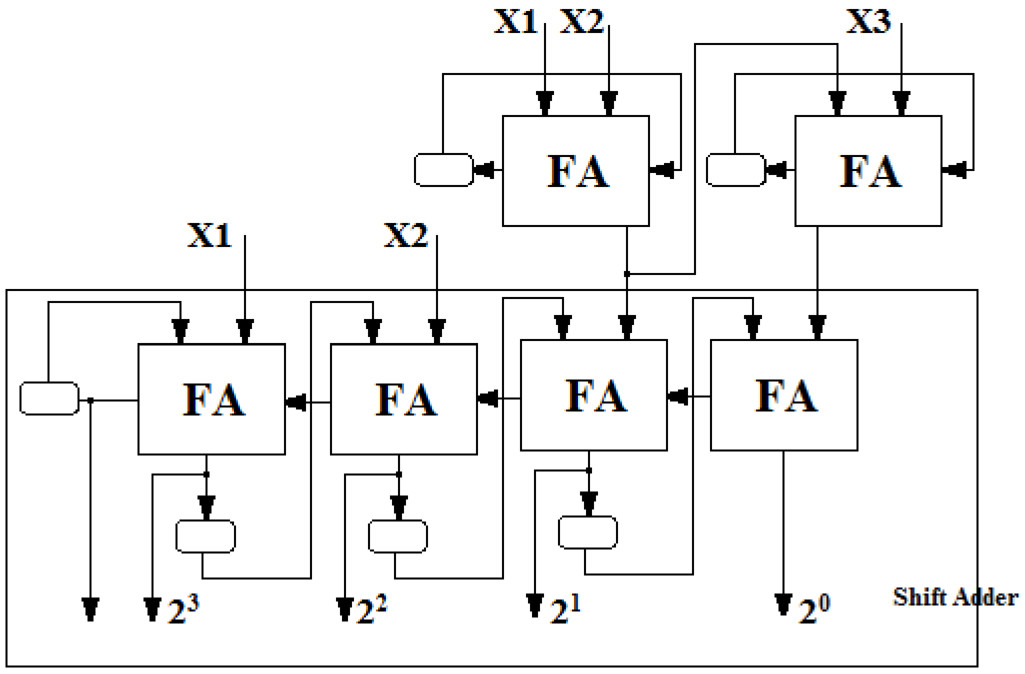


Fig. 3: The Summation Network Design of the Example

4. Architecture Design of Adder-based DA

Adder-based DA architecture is shown in Fig. 4. It consists of parallel to serial converter, summation network, and shift-adds part. The parallel to serial converter accepts parallel input and serially output bits of each input. The summation network is a tree structure that connects required input and generates their summation terms. The shift-adds part shifts and accumulates the output of the summation network to form the result.

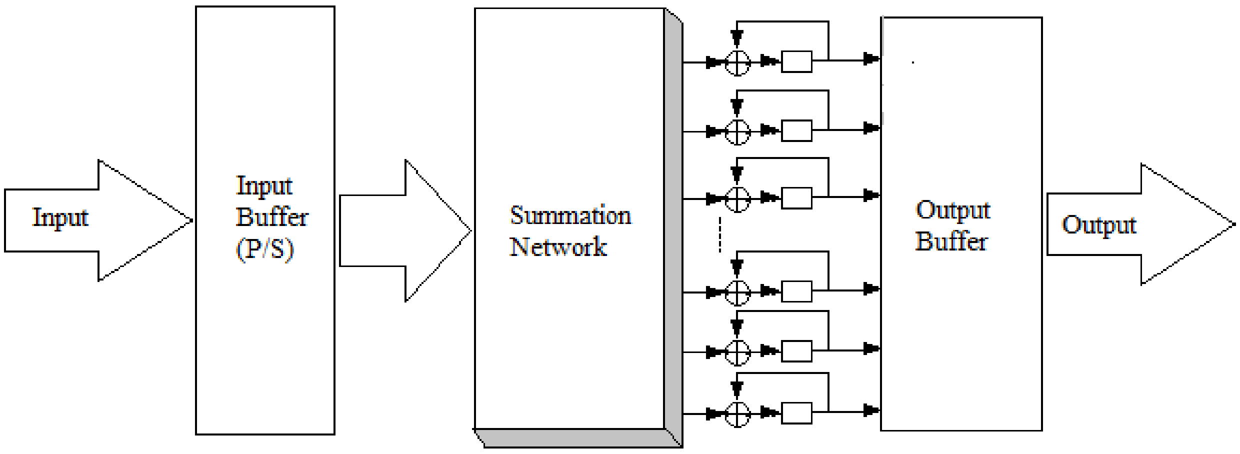


Fig. 4: Adder-based Architecture

III. ALGORITHM FOR BIT LEVEL IMPLEMENTATION OF DCT

The DCT of a sequence {*x* (*n*), *n* = 0, 1, 2, ……, *N*-1} is defined as

$$X(k) = \frac{2}{N}\mathcal{E}(k)\sum_{n=0}^{N-1} x(n)\cos\left[\frac{\pi(2n+1)k}{2N}\right] \qquad (8)$$

for *k* = 0, 1, 2, ……, *N*-1

where

$$\mathcal{E}(k) = \begin{cases} (2)^{-\frac{1}{2}} & for\ k = 0 \\ 1 & for\ 1 \le k \le N-1 \end{cases} \qquad (9)$$

Neglecting the scale factor and denoting cos $\left[\frac{\pi(2n+1)k}{2N}\right]$by C_{kn}, equation (1) is expressed a:

$$X(k) = \sum_{n=0}^{N-1} x(n)C_{kn} \qquad (10)$$

Each row transform given by equation (10), can be written as:

$$X(k) = \sum_{n=0}^{\frac{N}{2}-1}[x(i) + (-1)^k x(N-1-i)]\,C_{ki} \qquad (11)$$

Using matrix notation, for *N* = 8, equation (10), can be written as

$$\begin{bmatrix} X(0) \\ X(2) \\ X(4) \\ X(6) \end{bmatrix} = \begin{bmatrix} C_{00} & C_{01} & C_{02} & C_{03} \\ C_{20} & C_{21} & C_{22} & C_{23} \\ C_{40} & C_{41} & C_{42} & C_{43} \\ C_{60} & C_{61} & C_{62} & C_{63} \end{bmatrix} \begin{bmatrix} x(0)+x(7) \\ x(1)+x(6) \\ x(2)+x(5) \\ x(3)+x(4) \end{bmatrix} \qquad (12)$$

$$\begin{bmatrix} X(1) \\ X(3) \\ X(5) \\ X(7) \end{bmatrix} = \begin{bmatrix} C_{10} & C_{11} & C_{12} & C_{13} \\ C_{30} & C_{31} & C_{32} & C_{33} \\ C_{50} & C_{51} & C_{52} & C_{53} \\ C_{70} & C_{71} & C_{72} & C_{73} \end{bmatrix} \begin{bmatrix} x(0)-x(7) \\ x(1)-x(6) \\ x(2)-x(5) \\ x(3)-x(4) \end{bmatrix} \qquad (13)$$

Equations (12) and (13) describe the computation of even and odd coefficients for *N* = 8, respectively.

1. Architecture for Implementation of 8-point DCT

The architecture for implementation of 8-point DCT is shown in Fig. 5. The parallel to serial (P/S) converter converts input to serial order. Two summation networks accept these serial inputs to evaluate the summation terms. The speed of summation network is 2-bit summations per cycle. The accumulator uses shift-adders to accumulate 16-bit words per cycle to perform shift addition. Each shift-adder consists of a carry save adder and a binary look-ahead carry (BLC) adder. The carry save adder sums the two input numbers and the previous result into two 16-bit numbers carry and partial sum. Then the BLC adder sums the two numbers to the accumulated result of the inputs.

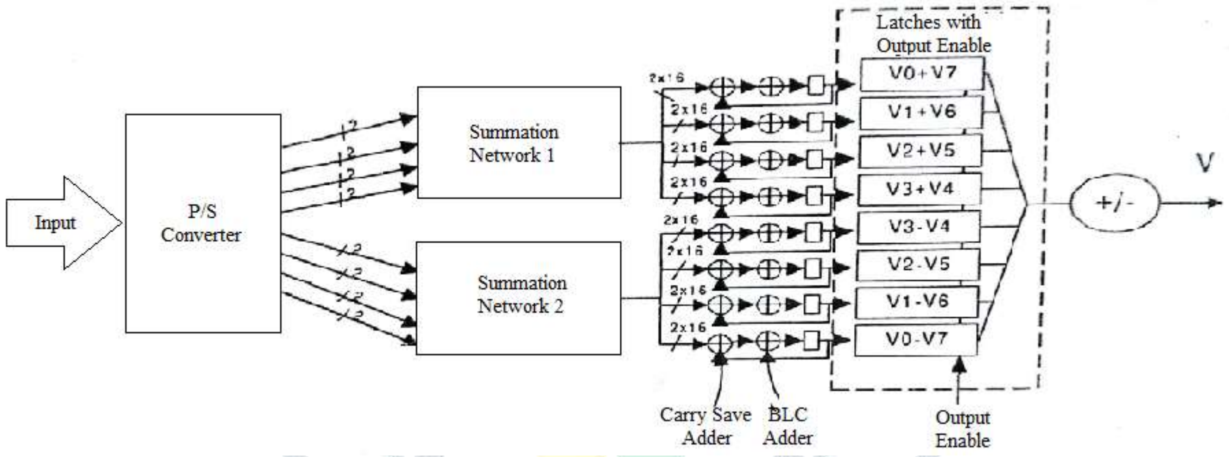


Fig. 5: Block Diagram of 8-point DCT

2. Hardware Consideration

The serial adder in summation network is composed of two full-adders and a *D* flip-flop (*D*-FF) with reset, shown in Fig. 6. The final network contains 22 full-adders, 11 *D*-FFs and 30 output latches. The gate count of the network is 481, while 40% of the gate count is output latches. The adder-based DA needs only 30% of ROM area compared with ROM-based DA.

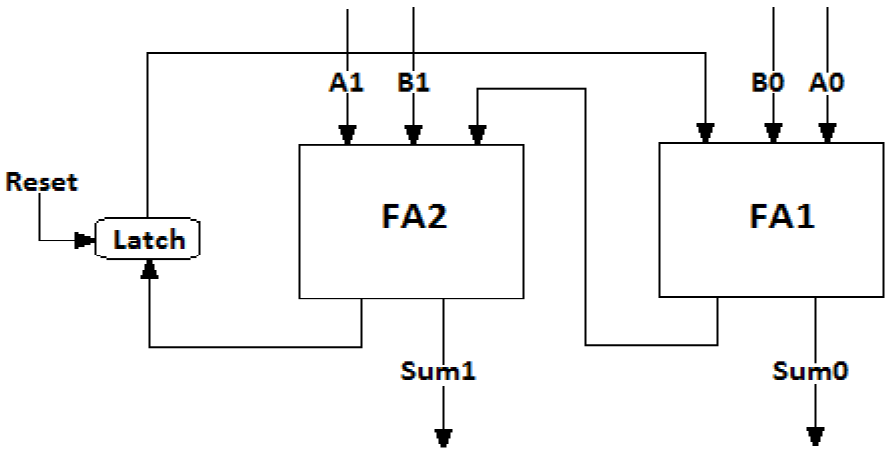


Fig. 6: A 2-bits Shift-adder used in the Summation Network

IV. CONCLUSION

In this paper, we have presented the computation of inner product of two vectors using adder-based DA. This algorithm decomposes the fixed coefficients into bit level instead of decomposing variable input data into bit level. The adder-based DA requires less area and smaller delay than ROM-based DA. So adder-based DA is a superior design choice over ROM- based DA in current DA applications. It needs only 30% ROM area compared with ROM-based DA.

REFERENCES:

[1] Peled, and Bede Liu, "A New Hardware Realization of Digital Filters", *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 22, no. 6, pp. 456-462, December 1974.

[2] S. A. White, "Applications of Distributed Arithmetic to Digital Signal Processing: A Tutorial Review", *IEEE ASSP Magazine*, vol. 6, no. 3, pp. 4-19, July 1989.

[3] S. Zohar, "New Hardware Realizations of Nonrecursive Digital Filters", *IEEE Transactions on Computers*, vol. C-22, no. 4, pp. 328-338, April 1973.

[4] Burrus, "Digital Filter Structures Described by Distributed Arithmetic", *IEEE Transactions on Circuits and Systems*, vol. 24, no. 12, pp. 674-680, December 1977.

[5] M. T. Sun, T. C. Che, and A. M. Gottlieb, "VLSI Implementation of a 16 × 16 Discrete Cosine Transform", *IEEE Transactions on Circuits and Systems*, vol. 36, no. 4, pp. 610-617, April 1989.

[6] S. Uramoto et al., "A 100 MHz 2-D Discrete Cosine Transform Core Processor", *IEEE Journal of Solid-State Circuits*, vol. 27, no. 4, pp. 492-499, April 1992.

[7] Chen, T. Chang, and C. Jen, "The IDCT Processor on the Adder-based Distributed Arithmetic", *Proc. Symp. VLSI Circuits*, pp. 36-37, 1996.

[8] H. C. Karathanasis, "A Low ROM Distributed Arithmetic Implementation of the Forward/Inverse DCT/DST using Rotations", *IEEE Transactions on Consumer Electronics*, vol. 41, no. 2, pp. 263-272, May 199.