# A Real Time Correlator Architecture Using Distributed Arithmetic Principles

A. Benjamin Premkumar and T. Srikanthan
Nanyang Technological University
School of Applied Science
Singapore - 2263

*Abstract-* **A real time correlator design based on the principles of Distributed Arithmetic (DA) is described. This design is shown to be more efficient in terms of memory requirement than the direct DA implementation, specially when the number of coefficients is large. Since, the proposed architecture implements the sum of product evaluation, it can be easily extended to finite and infinite response filters. Methods to further reduce the memory requirements are also discussed. A brief comparison is made between the proposed method and different DA implementations**

## 1 Introduction

The increasing flexibility and decreasing cost of computing technology have made real time Digital Signal Processing (DSP) both possible and cost effective. In many areas such as radar and sonar detection, speech processing etc., real time signal processing is best performed by special purpose hardware. An operation that is common to many signal processing applications is the sum of products evaluation. Some of the most commonly used DSP implementations using the sum of products operation are: Finite Impulse Response filters (FIR), Infinite Impulse response filters (IIR) and Fast Fourier Transforms (FFT). This type of computation is executed most efficiently by DA principles. The advantage of DA is its efficiency of mechanization of the sum of products operation. For example, the DFT can be evaluated using DA, since the expression for the DFT is given by: $X(k) = \frac{1}{N}\sum_{n=0}^{N-1} x(n)e^{-j2\pi\frac{n}{N}k}$, and contains the sum of products. Generally, the FIR filter can be described by the equation $y_m = \sum_{k=0}^{K-1} a_k x_{m-k}$, where the infinite sequences $\{x_j\}$, $\{y_i\}$ are the input and the output sequences, respectively, and $a_k$ is the set of $K$ coefficients. This expression is very general, for a trivial modification to the above expression, converts it into a cross correlation operation, $y_m = \sum_{k=0}^{K-1} a_k xm + k$. Here, $\{x_i\}$ is an infinite sequence of samples of data which is cross correlated with $K$ sample reference function to yield an infinite sequence of cross correlation samples, $\{y_i\}$. Since, both FIR and IIR filters can be described by similar equations, any architecture implementing one could well be used to implement the other.

In this paper, we propose an architecture which evaluates the sum of products efficiently using DA principles. In the implementation of the architecture, modifications have been made to the direct DA implementation resulting in reduced memory and faster throughput with minimum initial latency. Although, the implementation described is based on real numbers, the principle can be extended to complex numbers easily.

# 2 The Correlator

Several hardware implementations of the correlator architecture are available in literature [1, 2, 3]. In all these implementations, emphasis is laid on the multipliers to speed up product evaluation. However, in the implementation of correlator using the DA principle, emphasis is laid on the reduction in memory and hardware, since no multiplication is performed in the evaluation of the product of two numbers. The principle underlying the design of the proposed correlator is the somewhat different implementation of the DA concept.

It is well known that DA is a computational operation that forms the inner product of a pair of vectors [4]. DA is considered slow because of its bit serial nature of computation. However, this seemingly slow bit serial nature of DA is not real if the number of bits in each element of the vector is nearly equal to the number of elements in the vector. As an example of DA implementation of the sum of products, consider the expression:

$$y = \sum_{k=1}^{K} a_k x_k \tag{1}$$

where, $a_k$ are the coefficients and the $x_k$ are the input data. This expression for the sum of products can be rewritten as:

$$y = \sum_{k=1}^{K} a_k \sum_{n=0}^{N-1} b_{kn} 2^n \tag{2}$$

where each $x_k$ has $N$ bits. Interchanging the order of summation results in:

$$y_n = \sum_{n=0}^{N-1} \{ \sum_{k=1}^{K} a_k b_{kn} 2^n \} \tag{3}$$

Since $b_{kn}$ can take values of either 0 or 1, it is easier to precompute the values for $\sum_{k=1}^{K} a_k b_{kn}$ and store them rather than compute them as and when $b_{kn}$ arrive. This would mean that a memory of $2^K$ is needed for storing $\sum_{k=1}^{K} a_k b_{kn}$ for different combinations of $b_{kn}$. Stanley White, in his paper [4], has proposed a way by which the memory can be reduced by a factor of 2 by recasting the input data as offset binary code instead of straight binary code.

Modifications to the principles described above, have been made and implemented as architectures [5, 6, 7]. A correlator based on DA is described in [8] in which the sum of products expression is converted into three sums. By changing the order of summation of the three sums and using negabinary representation for the 2's complement input data, sum of products is evaluated. In the design to be discussed, the sum of products is evaluated using look up tables and high speed adders. In this design, the size of the look up tables used is smaller than that used in the direct DA implementation. The following sections describe the correlator implementation and also discuss the modifications to the proposed method which result in further reduction in memory. A comparison is made with the direct DA design of the correlator using 2 bits at a time (2BAAT) and 4 bits at a time (4BAAT).

# 3 Design of the Correlator

The proposed correlator is based on the design explained in reference [9]. In the evaluation of the expression $y = \sum_{k=0}^{K-1} a_k x_{m-k}$, it is seen that $y$ at any instant contains the sum of products of all the input samples until that instant with all coefficients. Hence, it would be efficient to multiply all coefficients with each sample as it is input to the correlator and store them temporarily. When a new sample arrives at the input, it is also multiplied with all coefficients and then added to the delayed sums generated until the previous sample. However, in the proposed design, no multiplication of the input sample with the coefficients takes place. Look up tables in which the products of the coefficients and input data (for different combinations of the bits in the input data) are stored are used. A block diagram of the architecture is shown in Fig. 1.

The correlator is capable of accepting data serially and producing correlated output with a latency of one clock period. As can be seen from the figure, the number of tables required is equal to the number of coefficients used in the correlation. The input data sequence is fed serially into all coefficient tables. The bit pattern of each input data is used to access the memory location in which the product of the coefficient with that particular bit pattern is stored. The accessed data enters the summer where it is added with the delayed data corresponding to the previous input. Delay units in the circuit enable the previous sum to be added to the product of the current input and the coefficients. The output is available at the last adder unit with a delay corresponding to one clock cycle.

Storing the products of the coefficients with the input data requires a memory of size $M = n \times 2^b$, where, $M$ is the total memory required, $n$ is the number of coefficients and $b$ is the number of bits in the input data. However, it is possible to partition the input data into two fields of $\frac{b}{2}$ bits each and store data with suitable weights assigned to the partitioned bits. This way, it is possible to reduce the total memory required. For any input, the products corresponding to the two partitioned groups of data bits are accessed and then added. This extra addition is not an overhead, since the existing adders can be clocked at twice the input data rate. Consider, for example, the following case: Supposing the number of coefficients is 8 and the input data width is 8 bits, direct implementation of the look up tables will require a total memory, $M = 8 \times 256$. However, partitioning the input data into two groups of 4 bits each results in a total memory requirement of $M = 2 \times 8 \times 15$, a considerable saving of memory. The memory can be further reduced by a factor of 2 by partitioning input data into smaller groups of 2 bits each. However, an additional set of adders will be required whenever data is accessed from the table. Fig.2 illustrates the architecture using this principle. With single partitioning, the total memory requirement is given by the expression: $2n(2^{\frac{b}{2}} - 1)$ for $b$ even. For $b$ odd, the memory requirement is given by the expression $M = n(2^{\frac{b+1}{2}} + 2^{\frac{b-1}{2}} - 2)$. The adders in the architecture can be speeded up using carry save techniques. In the correlation operation, the result, $y$, at the $n^{th}$ sample instant is not expected at the output until $n$ samples arrive at the correlator input. Hence, the summing operation can be performed in the carry save domain, where the propagation and the addition of the carry is deferred until the very last adder where the

| Parameter | 1BAAT | | 2BAAT | | 4BAAT | | Proposed | |
|---|---|---|---|---|---|---|---|---|
| No. of Bits (input) | 8 | 16 | 8 | 16 | 8 | 16 | 8 | 16 |
| No. of Coeffts (K) | 4 | 12 | 4 | 12 | 4 | 12 | 4 | 12 |
| No. of Add. lines | 3 | 11 | 7 | 23 | 15 | 47 | 7 | 13 |
| Output Data/Clock | 0.5 | 0.75 | 1 | 1.5 | 2 | 3 | 1 | 1 |
| Memory Required | $m$ | $m$ | $m^2$ | $m^2$ | $m^4$ | $m^4$ | $m_1$ | $m_2$ |

Table 1: Comparison between the DA and Proposed Method

output is available. This eliminates the delay due to carry propagation in the intermediate adders, thus speeding up the overall sum of product operation.

# 4 Comparison with Conventional DA Approach

In the direct DA implementation, with input data at the rate of 1 bit at a time (1BAAT) computation of the sum of products is slow. If $b$ be the number of bits in the input data, $b$ clock cycles are needed to form the sum of products. However, equivalent of $K$ separate products are being formed during $b$ clock cycles. Thus, it is seen, that if $K$ is greater than $b$, then the DA processor is faster than the conventional multiplier in the evaluation of the sum of products. However, the number of inputs to the memory may restrict the number of coefficients. The computation can be speeded up at the expense of linearly increased or exponentially increased memory [4]. In the case of linearly increased memory, the input word is partitioned into $L$ subwords. The memory requirement is determined by the number of bits in each subword. The data is input as subwords and this increases the speed by a factor of $L$. However, high speed and large capacity accumulators are needed. Because of the bit serial nature of the input data, shift operations become mandatory. By inputing data, $p$ bits at a time, the memory increases exponentially and is given by the expression $M = \frac{1}{2}\{2^{Kp}\}$. Since the memory increases exponentially with the number of coefficients, this approach to DA implementation becomes impracticable in applications where $K$ is large. Also, in this approach, the number of input lines to the memory is directly related to the number of coefficients. Thus, VLSI implementation of this method to applications where $K$ is large, becomes exceedingly difficult. In the correlator described, the memory requirement is governed by the number of bits in the input data. Since, in most of the signal processing applications, either 16 or 8 bit data resolutions are sufficient, the memory requirements are not large. As stated earlier, by either increasing the number of adders or speeding up the adders, one might be able to reduce the memory considerably. Table 1 compares the direct DA implementation with $p$ bits at a time with the proposed correlator with respect to the speed and memory requirements. In the table, $m$, $m_1$ and $m_3$ are defined as follows: $m = 0.5 \times 2^K$, $m_1 = 2K(2^4 - 1)$ and $m_2 = 2K(2^8 - 1)$. The expression for $m$ assumes that the input data is recast into offset binary code. This table, however, does not take into account the additional time required in shifting the data for alignment in the case of 1BAAT, 2BAAT and 4BAAT approaches. This additional time may significantly reduce

throughput rates.

# 5 Conclusion

DA is a very efficient means of evaluating the sum of products. However, direct implementation of DA principles involves use of large memory. The memory requirements are even larger if the speed of the computation is to be increased or whenever the number of coefficients required is large. In this paper, a new correlator has been proposed in which the input data is partitioned into fields consisting of smaller number of bits. This way the total memory needed is reduced. The only additional requirement is that the existing adders be clocked at twice the data rate or high speed adders be used. In the design, carry save adders have been proposed, since in the evaluation of the sum of products, the output is not expected until all the samples arrive at the correlator. This design lends itself very well for VLSI fabrication of the correlation operation.

# References

[1] J. Canaris and S. Whitaker, "A High Speed CMOS Correlator", *2nd NASA SERC Symposium on VLSI Design*, Univ. of Idaho, Moscow, Idaho, Nov. 1990.

[2] A. Peled and B. Liu, "A New Hardware Realization of Digital Filters", *IEEE Trans. on Acoustics, Speech and Signal Processing*, vol. ASSP - 22, Dec. 1974, pp. 456-462.

[3] S. Zohar, "New Hardware Realizations of Nonrecursive Digital Filters", *IEEE Trans. on Computers*, vol. C-22, Apr. 1973, pp. 328-338.

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

[5] S. G. Smith and P. B. Denyer, "Efficient Bit Serial Complex Multiplication and Sum of Product Computation Using Distributed Arithmetic", *Proc., International Conference on Acoustics, Speech and Signal Processing*, Tokyo, Apr. 1986, pp. 2203-2206.

[6] C. F. N. Cowan and J. Mavor, "New Digital Adaptive Filter Implementation Using Distributed Arithmetic Techniques", *IEE Proc*, vol. 128, Pt. F, no. 4, Aug. 1981, pp. 225-230.

[7] C. S. Burrus, "Digital Filter Realization in Distributed Arithmetic", *Proc. European Conf. on Circuit Theory and Design*, Genoa, Italy, Sept. 1976.

[8] S. Zohar, "A VLSI Implementation of a Correlator/Digital Filter Based on Distributed Arithmetic", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 37, no. 1, Jan. 1989, pp. 156-160.

[9] A. B. Premkumar, J. Purviance and M. Shamanna, "A Two Dimensional VLSI Correlator", *Northcon Conference*, Seattle, Wasington, Oct. 1990.