Novel Sparse OBC based Distributed Arithmetic Architecture for Matrix Transforms

S.Chandrasekaran and A.Amira
Electronic and Computer Engineering,
School of Engineering and Design
Brunel University, West London, UB8 3PH, UK
Email: shrutisagar.chandrasekaran@brunel.ac.uk, abbes.amira@brunel.ac.uk

Abstract— **Inner Product (IP) forms the basis of a number of signal processing algorithms and applications such as transforms, filters, communication systems etc. Distributed Arithmetic (DA) provides an effective methodology to implement IP of vectors and matrices using a simple combination of memory elements, adders and shifters instead of lumped multipliers. This bit level rearrangement results in much higher computational efficiencies and yields compact designs highly suited for high performance resource constrained applications. Offset Binary Coding (OBC) is an effective technique to further optimize the DA, and allows us to reduce the memory requirements by a factor of two, with minimum additional computational complexity. This makes OBC-DA attractive for applications that are both resource and memory constrained. In addition, sparse matrix factorization techniques can be exploited to further reduce the size of the DA-ROMs. In this paper, the design and implementation of a novel OBC based DA is demonstrated using a generic architecture for implementing Discrete Orthogonal Transforms (DOTs). Implementation is performed on the Xilinx Virtex-II Pro Field Programmable Gate Array (FPGA), and a detailed comparison between conventional and OBC based DA is presented to highlight the trade offs in various design metrics including performance, area and power.**

I. Introduction

Matrix multiplication forms the basic building block of a number of signal processing algorithms, including transformation kernels that are used in a number of image and signal processing applications. Key among these are Discrete Orthogonal Transforms (DOTs) that belong to the domain of linear transformations, which are mathematically well-founded [1]. Commonly used DOTs include the Discrete Fourier Transform (DFT), Walsh Hadamard Transform (WHT), Haar Wavelet Transform (HWT), Karhunen Loeve Transform (KLT) and Discrete Cosine Transform (DCT). Other orthogonal factorisation techniques such as Singular SVD are also based on matrix multiplication. The DFT is an indispensable tool for processing images and signals in the spectral domain. The WHT is among the simplest of the class of DOTs, and is widely used in compression, communication and encoding applications. The KLT is a statistically optimised transform that is used primarily for approximating a set of vectors, and finds applications in image processing and computer vision. The DCT is widely used in image processing and is one of the most popular techniques used in various compression algorithms [2].

Since DOTs are linear transformations on matrices, they are essentially composed of matrix-matrix or matrix-vector multiplications, which are computationally expensive. To process a K-length vector denoted by ϕ according to a transformation function given by $g = A \cdot \phi$ (where A is a square matrix) requires $O(K^2)$ operations. For large values of K, this is clearly a problem. There is a real need for dedicated processors for high speed computation of the transform to meet the requirements of real time signal processing. The choice of suitable arithmetic techniques for performing Inner Product (IP) is also an important factor.
For System on Chip (SoC) Application Specific Integrated Circuit (ASIC) architectures, Distributed Arithmetic (DA) has been regarded as an important tool for computing IP. ROM-based DA uses a ROM table to store the pre-computed data, which makes it regular and efficient in the use of silicon area [3][4]. The basic operations required for DA are a sequence of ROM accesses, addition, subtraction and shift operations of the input data sequence. All of these functions are efficiently mapped to FPGA structures [5]. However, when the size of the inner products increases the ROM area increases exponentially and becomes impractically large, even when using ROM partitions [6]. Reducing vector lengths by means of matrix factorisation to split the matrix multiplication into smaller blocks allows us to achieve compact area efficient designs. Additionally, mathematical transformations such as Offset Binary Coding (OBC) can be efficiently exploited for reducing the ROM size by a factor of two, with a marginal increase in computational complexity. In this paper, a mathematical framework for implementation of OBC based DA, in conjunction with matrix factorisation and sparse matrix techniques is discussed. Tradeoffs between various performance metrics and power are evaluated, and a comparison is made with the standard DA approach in order to quantify the gains achieved through the transformation processes.

The rest of the paper is organised as follows. Mathematical background behind DA, OBC-DA and factorisation techniques are described in Section II. The proposed architecture for factorised matrix OBC-DA is presented in Section III. FPGA implementation results are provided in Section IV. Concluding remarks are presented in Section V.

II. Mathematical Background

A. Standard DA

Direct applications of the standard DA algorithm is mathematically presented as follows. Consider an IP of two vectors represented as the following sum of products:

$$Y = \sum_{k=1}^{K} A_k X_k \tag{1}$$

where A is the constant vector, X is the input vector, and Y is the transformed vector. If we consider X_k to be in the form of a scaled 2's complement binary number, it can be represented as:

$$X_k = -b_{k0} + \sum_{n=1}^{N-1} b_{kn} 2^{-n} \tag{2}$$

Substituting Eq. 2 in 1 we get:

$$Y = \sum_{k=1}^{K} A_k \left[-b_{k0} + \sum_{n=1}^{N-1} b_{kn} 2^{-n} \right] \tag{3}$$

By rearranging the order of summations in order to convert the conventional sum of products into a "distributed" form, we get:

$$Y = \sum_{n=1}^{N-1} \left[\sum_{k=1}^{K} A_k b_{kn} \right] 2^{-n} + \sum_{k=1}^{K} A_k (-b_{k0}) \tag{4}$$

The term $\sum_{k=1}^{K} A_k b_{kn}$ can have only 2^k possible values, it is possible to precompute and store these values in a ROM. By addressing this ROM through N cycles using the input data and performing simple shift-accumulate operations, the IP can be calculated. In order to accommodate both the terms in Eq. 3, a $2 \cdot 2^k$ word ROM is required to store the precomputed data. By using an adder/subtractor block; this can be reduced to a size of 2^k. By using DA, we can clearly see that a typical IP has been reduced in complexity from $O(K)$ multipliers to just $O(K-1)$ additions and shift operations. This represents an order of magnitude reduction in computational circuit complexity, which is traded off for memory of size 2^k, making DA ideal for resource constrained systems. However, the keen observer would notice that the reduction in computational complexity by offloading logic to memory does not necessarily result in a reduction of area complexity. This makes reduction of ROM size, without significant increase in control overhead a key requirement for efficient hardware implementation of DA.

B. OBC based DA

The ROM size of standard DA can be further reduced to 2^{k-1} by applying OBC technique. It is worth mentioning that OBC-DA does not necessitate recoding inputs or outputs, since it is only used to interpret and not convert the input data to be in OBC form [4]. An element of the input vector can be alternatively expressed as:

$$X_k = \frac{1}{2} \left[X_k - (-X_k) \right] \tag{5}$$

Substituting Eq. 2 in 5 we get:

$$X_k = \frac{1}{2} \left[-(b_{k0} - \overline{b}_{k0}) + \sum_{n=1}^{N-1} (b_{kn} - \overline{b}_{kn}) 2^{-n} - 2^{-(N-1)} \right] \tag{6}$$

Let $c_{kn} = b_{kn} - \overline{b}_{kn} \quad n \neq 0$ and $c_{k0} = -(b_{kn} - \overline{b}_{kn})$. Eq. 6 is now expressed as:

$$X_k = \frac{1}{2} \left[\sum_{n=0}^{N-1} c_{kn} 2^{-n} - 2^{-(N-1)} \right] \tag{7}$$

Substituting Eq. 7 in Eq. 1 we get:

$$Y = \sum_{n=0}^{N-1} Q(b_n) 2^{-n} + 2^{-(N-1)} Q(0) \tag{8}$$

where

$Q(b_n) = \sum_{k=1}^{K} \frac{A_k}{2} c_{kn}$ and $Q(0) = \sum_{k=1}^{K} \frac{A_k}{2}$. The lower half of the ROM table obtained from the precomputation of the OBC DA combinations is a mirror image of the upper half, with sign reversed. By using the most significant address line (first element in the input vector) as an inverting signal, it is possible to reduce the ROM size by a factor of two, when compared to regular DA. Theoretically, it is possible to recursively apply OBC to reduce the ROM size, at the expense of additional logic. For the maximum case of recursion, the logic depth becomes extremely large, and the purpose of trading computational complexity for memory is defeated.

C. DOT Manipulation for Complexity Reduction

For most unitary transforms (and for K an integral power of 2) with the exception of KLT (because of data dependency), fast algorithms exist. They are essentially based on the fact that the transformation kernel can be partitioned into some intermediate steps which can be subsequently reused in further iterations. The factorisation and partitioning techniques for each individual DOT is specific to the kernel structure, and must be applied on a case to case basis. For example, in the case of the DFT, Cooley-Tukey factorisation is used as follows:

$$A = A_1 A_2 ... A_m \tag{9}$$

where and $A_1, A_2 .... A_{m-1}$ are 2-sparse matrices and A_m is $K/\left(2^{(m-1)}\right)$ sparse. For a 1-D DFT the area complexity of the OBC-DA is now reduced to:

$$O\left(2\left[(m-1) + 2^{\left(\frac{K}{2^{m-1}} - 2\right)}\right]\right) \tag{10}$$

For a 2 dimensional case (matrix-vector), such as the K-point WHT or K-point HWT, sparse matrix factorization can be effectively exploited for reducing the ROM size to:

$$O\left(4(m-1) + \frac{K}{2^{m-1}}^{\left(\frac{K}{2^{m-1}} - 1\right)}\right) \tag{11}$$

as compared to $O(2^K)$ for the conventional DA approach. In the case of 8-point DCT, Chen's algorithm is used to exploit symmetry of the coefficients effectively reducing an 8×8 matrix operation into two 4×4 operations. This is effectively the same as ROM decomposition which is also an effective technique for minimising ROM size in DA. Further reduction in ROM size when compared to conventional DA can be achieved by applying OBC technique. Quantitatively, for an L-point DCT, applying Chen's decomposition in conjunction with OBC reduces the area complexity of the algorithm to $O(K(2^{\frac{K}{2}-1}))$.

It is clear that sparse factorisation and matrix decomposition techniques when used in conjunction with OBC can yield highly efficient and compact structures with minimal overhead and are highly suitable for resource constrained systems [7]. A full discussion of all existing decomposition techniques is beyond the scope of this paper.

III. Proposed Architecture and OBC-DA Operation

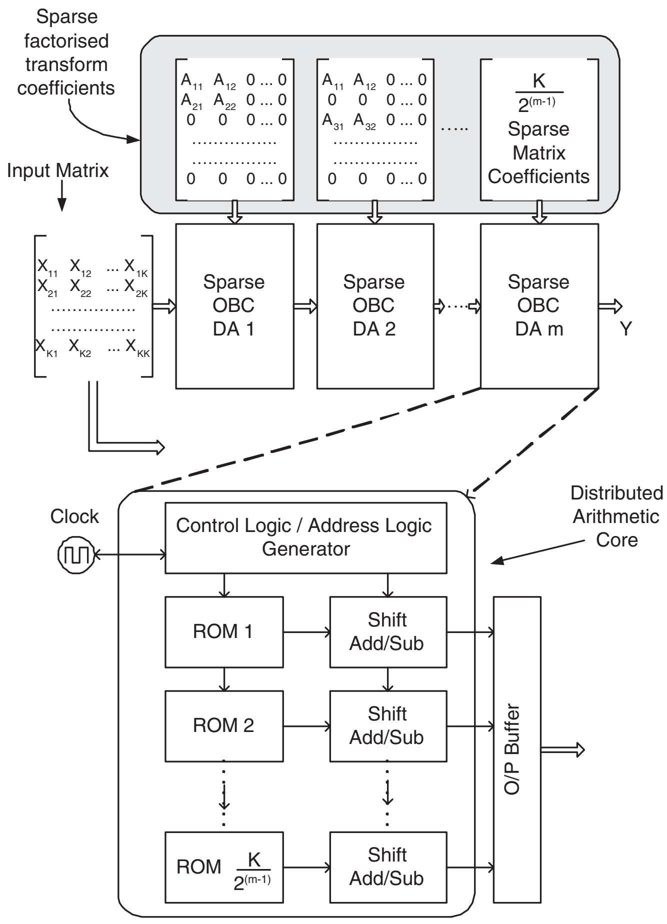
Sparse factorised transform coefficients
Input Matrix
K / 2^(m-1) Sparse Matrix Coefficients
Sparse OBC DA 1
Sparse OBC DA 2
Sparse OBC DA m
Y
Clock
Control Logic / Address Logic Generator
Distributed Arithmetic Core
ROM 1
ROM 2
ROM K/2^(m-1)
Shift Add/Sub
O/P Buffer

Fig. 1. Block diagram of the overall OBC-DA-sparse matrix based DOT implementation

The architecture of the whole system is presented in Fig. 1. It can be seen that the output of each DA stage is sequentially passed on to the following stages. It is worthwhile to mention that complete sparse factorisation, or reducing all but the last matrix to 2-sparse matrices is not necessary, and the exact level of factorisation and number of factors need to be evaluated on a case to case basis for each DOT by the designer, and the best tradeoff between latency (pipeline length), complexity and DA size needs to be carefully evaluated. In the case of Fig. 1, a generic architecture for $(m-1)$ 2-sparse factors and one additional $K/(2^{(m-1)})$ sparse matrix has been presented.

Mapping the OBC-DA algorithm stated mathematically in Eq. 8 yields the architecture shown in Fig. 2.

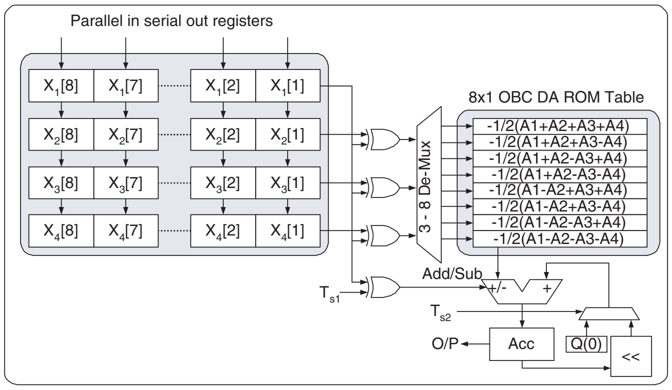

Fig. 2. Architecture for the OBC-DA block for the case K = 4 and N = 8

The operation of the OBC-DA core for an example case $N = 8$ and $K = 4$ is described as follows. During the 1^{st} clock cycle, the signal T_{s2} is asserted and the value $Q(0)$ is preloaded into the accumulator. For the first 7 clock cycles, the address locations of the OBC-DA ROM are selected using a bitwise XOR combination of the first element of the variable vector in the IP, X_1 with all the other elements $X_2...X_4$ starting with the LSB of each element. The mode of operation of the accumulator is controlled by the Add/Sub line. On the 8^{th} Clock cycle, the signal T_{s1} is also asserted, and the final result is obtained at the end of the cycle. At the end of this process, the input vector of the IP is reloaded, and the entire operation is repeated. This circuit can be easily generalised for all feasible values of N and K.

IV. Implementation Details and Results Obtained

The IP cores developed for comparing various metrics of regular DA and OBC-DA have been implemented on the Xilinx Virtex-II-Pro xc2vp100 SRAM FPGA which is capable of handling large, complex designs. It has a total of 44096 slices and 1040 I/O Buffers. The FPGA is also incorporates embedded PowerPC processors and 3.125 Gbps RocketIO serial transceivers [8]. The choice of this high performance platform for prototyping the algorithms presented is influenced by the fact that IP is a highly computationally intensive operation.

A. Design Methodology

The hardware is designed using a hybrid combination of Handel-C [9] and parametrisable VHDL cores. The VHDL based cores are generated using Xilinx Coregen [8], and are used for small frequently used blocks in the design such as shifters, adders etc. Handel-C is used at the top level for architecture description and integration of the cores. Synthesis of EDIF netlist from the top level Handel-C code is performed using Celoxica DK4. Pin assignment details, timing constraints and hand routing of critical blocks yields highly optimised design and improves the power and energy consumption metrics. This is particularly important, as non optimal place and route tends to use long nets that consume more power than short nets, due to higher capacitance and DC load. Power estimation is performed using Xilinx XPower [8].

B. Performance Metrics

Performance metrics obtained for the implementation of standard DA and OBC-DA are presented graphically in Fig. 3. The graph yields some interesting insights into the tradeoffs and general trend for both cases. With respect to area, it can be clearly seen that for the case $K = 4$, the area of OBC-DA implementation is infact marginally higher than standard DA. This can be attributed to the fact that the increase in circuit complexity slightly outweighs the influence of reduction of ROM size. However, for larger values of K, a divergent trend is clearly observed, and we can see that OBC-DA implementation consumes less area when compared to the standard DA implementation. This gap becomes wider as K increases. In terms of frequency, for all cases of $K \geq 6$ it can be seen that OBC-DA outperforms standard DA. This indicates that the area reduction achieved in OBC-DA is not at the expense of performance. The overall trend is similar for both architectures. This can be explained by the fact that maximum frequency is a function of logic depth of the critical path, which increases at the same rate for both architectures, unlike in the case of area.

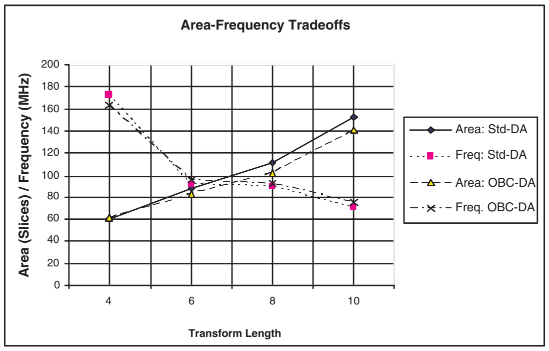

Fig. 3. Frequency-Area trends for both architectures for different values of K and $N = 8$. Frequency is in MHz and area is represented in FPGA slices.

C. Power Analysis

Power metrics for both architectures at constant frequency of 50MHz are presented in Table I. Total dynamic power consumption of both DA implementations is presented in Fig.4. It can be seen that as K increases, power dissipation of both cores increases. However, OBC-DA is more energy efficient than standard DA for any given value of K. I/O power for both architectures are identical at 3.44 mW for input and 57.58 mW, 60.78 mW, 60.78 mW, 63.98 mW for output at $K = 4, 6, 8, 10$ respectively. This is completely in accordance with expectations as no change has been made to the I/O sections of the two DA implementations.

TABLE I
ON-CHIP DYNAMIC POWER AT CONSTANT FREQUENCY

	Standard DA			OBC-DA		
K	**Clock**	**Logic**	**Signal**	**Clock**	**Logic**	**Signal**
4	4.68	9.46	11.9	3.93	9.5	12.68
6	12.29	13.37	33.85	13.61	12.86	30.28
8	13.42	16.56	39.67	13.99	15.75	36.93
10	14.55	23.46	61.61	13.47	22.24	49.23

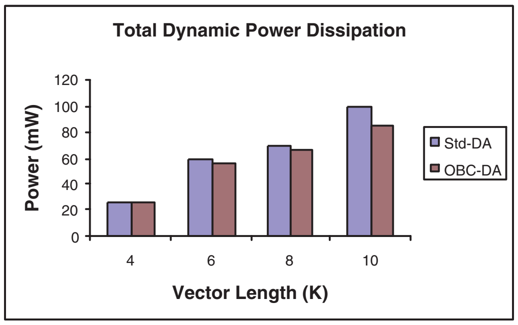

Fig. 4. Total dynamic power dissipation for different values of K and $N = 8$.

V. Conclusions

Novel architectures for resource constrained implementations of IP, particularly for DOTs based on sparse factorisation techniques combined with OBC-DA has been presented. It has been mathematically shown that area complexity can be greatly reduced by using these techniques in conjunction with each other. Despite increased complexity of control circuitry, it has been shown that OBC-DA outperforms standard DA in all key performance metrics including area, frequency and power dissipation. This shows that OBC-DA is preferable for developing the IP core, particularly in resource constrained systems.

References

[1] N. U. Ahmed and K. R. Rao, *Orthogonal Transforms for Digital Signal Processing*. Springer-Verlag, 1975.
[2] K. R. Rao and P. Yip, *Discrete Cosine Transform, Algorithms, Advantages and Applications*. New York: Academic Press, 1990.
[3] 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.
[4] K. K. Parhi, *VLSI Digital Signal Processing Systems Design and Implementation*. John Wiley and Sons, 1999.
[5] Application Note, "The role of distributed arithmetic in FPGA-based signal processing," www.xilinx.com/appnotes/.
[6] S. Chandrasekaran and A. Amira, "An area efficient low power inner product computation for discrete orthogonal transforms," in *Proceedings of the IEEE International Conference on Image Processing*, September 2005, pp. 1024–1027.
[7] ——, "FPGA implementation and power modeling of the fast walsh transform," in *Proceedings of the* 16^{th} *International Conference on Field Programmable Logic and Applications*, August 2006.
[8] [Online]. Available: www.xilinx.com
[9] [Online]. Available: www.celoxica.com