A Novel Reconfigurable Low Power Distributed Arithmetic Architecture for Multimedia Applications

Zhenyu Liu

School of Engineering and Electronic, The University of Edinburgh Edinburgh, EH9 3JL, UK
e-mail : zhenyu.liu@ed.ac.uk

Tughrul Arslan

School of Engineering and Electronic, The University of Edinburgh Edinburgh, EH9 3JL, UK
e-mail : T.Arslan@ed.ac.uk

Ahmet T. Erdogan

School of Engineering and Electronic, The University of Edinburgh Edinburgh, EH9 3JL, UK
e-mail : Ahmet.Erdogan@ed.ac.uk

Abstract - The use of reconfigurable cores in system on chip (SoC) designs is increasingly becoming a trend. Such cores are being used for their flexibility, powerful functionality and low power consumption. Distributed Arithmetic (DA) is a powerful algorithm wildly used in many fields of multimedia for its efficiency. This paper presents a novel reconfigurable adder-based architecture for DA to realize the inner product which is the key computation in many digital signal processing applications. 1D DCT is mapped onto the architecture. Compared with some existing ASIC designs, the new architecture achieves good performance in area, speed and power.

I. INTRODUCTION

Distributed Arithmetic (DA) has been wildly adopted for its computational efficiency in many digital signal processing applications such as DCT (Discrete Cosine Transform), DFT (Discrete Fourier Transform), FIR (Finite Impulse Response), and DHT (Discrete Hartley Transform) [1]. These applications involve the computation of inner products between two vectors, one of which is a constant.

The general method to generate the products is to use a MAC (multiply and accumulate) unit, which is fast but has a high cost in the case of long-length inner-products. In contrast, DA provides an efficient solution to realize the inner products by using memory look-up and accumulation operations. The idea behind the conventional DA, called ROM-based, is to replace multiplication operations by pre-computing all possible values and storing these in a ROM. According to [1], the ROM based DA can reduce the circuit size by 50-80% on average.

Custom reconfigurable technology emerged to satisfy the simultaneous demand for flexibility and efficiency. Custom/domain-specific reconfigurable arrays can be programmed to adapt for different applications, so the efficiency of the hardware and flexibility of the whole system is improved. Earlier works, such as [2-4], show good performance in area, power consumption and speed. Since a domain-specific reconfigurable architecture targets few application fields, it achieves better performance than a general purpose FPGA device.

In this paper, a novel reconfigurable DA architecture is presented which can implement inner products with less area usage and power consumption. The proposed architecture can implement any algorithm for the inner product computations, such as DCT, DFT, FIR, and DHT. All these signal processing algorithms are widely used in various multimedia standards such as H.261, H.263, MPEG-1, MPEG-2, MPEG-4, and JPEG2000. An adder-based DA is adopted in this architecture, which was introduced in [5]. Compared with a ROM-based DA, the approach needs only 10% of transistors and 30% of ROM area with comparable performance [5]. Our new architecture takes the advantage of the common summation terms when the fixed coefficients are decomposed into bit level which makes the architecture maximize the hardware efficiency. Due to its inherent hardware sharing property, the proposed architecture is very suitable for multiple inner product computations. The reconfigurable character of the architecture makes it flexible to switch form one function to another, which means the same hardware architecture can perform different algorithms at different times.

The rest of the paper is organized as follows. In section 2, we review the related work in the literature. In section 3, a brief description about the basic definitions of DA is provided and the reconfigurable DA is introduced. Our architecture is described in detail in section 4 which includes the overview of reconfigurable architecture, two-level adder butterfly structure, the Wallace tree matrix and the trade off between area and speed. The DCT algorithm and its implementation with our reconfigurable DA are described in section 5. The experimental results and performance evaluation are given in sections 6 and 7. Finally, summary is presented in section 8.

II. RELATED WORK

Over the years, significant research work has been carried out on DA techniques and its implementation for DCT and other applications. In [6], a hardwired DA method is implemented for DCT with radix-2 multibit coding for the minimum resource and symmetric transpose memory for high speed. In [5], an adder-based DA is proposed to generate the inner product of vectors for DCT. The approach reduced 2/3 ROM area and nearly 9/10 transistor count with comparable performance when compared with ROM-based DA. In [7], a DA-based algorithm is introduced which can formulate 1-D any-length DHT as cyclic convolutions. It simplifies the

1-4244-0630-7/07/$20.00 ©2007 IEEE.

ROM design process and increases the processing speed for utilizing identical ROM modules and eliminating the accumulation loop in the processing elements.

Numerous works has been done on reconfigurable architectures and their implementations. However, no architecture has been designed specifically for reconfigurable DA yet. In [15], a special reconfigurable architecture for DCT is described. The architecture is designed especially for implementing DCT with different algorithms: pure-RAM, mixed-RAM and CORDIC. However, since this architecture could be used only for DCT application, its application field is fixed.

III. RECONFIGURABLE DA ALGORITHMS

A. DA Algorithms

DA is a bit-serial operation that computes the inner product of two vectors without needing to use multiply operations. Let us consider the computation of the following inner (dot) product with L-dimensional vectors:

$$Z = A \bullet X = \sum_{i=0}^{L-1} C_i X_i \quad (1)$$

where A = [C_0, C_1,···, C_{L-1}] is an M bits fixed coefficient vector and X = [X_0, X_1,···, X_{L-1}] is an N bits input vector. C_i and X_i can be expressed in two's complement binary as follows:

$$C_i = -C_{i,(M-1)} \bullet 2^{M-1} + \sum_{j=0}^{M-2} C_{i,j} \bullet 2^j \quad (2)$$

$$X_i = -X_{i,(N-1)} \bullet 2^{N-1} + \sum_{k=0}^{N-2} X_{i,k} \bullet 2^k \quad (3)$$

where $C_{i,j}$, $X_{i,k} \in \{0,1\}$ is the jth and kth bit of vector element C_i and X_i respectively. To realize the inner product computation, the conventional DA uses a ROM-based architecture. Another method is to adopt an adder-based architecture.

B. ROM Based DA

ROM-based DA speeds up the multiplication process by pre-computing all possible values and storing them in a ROM. By substituting (3) in (1), the output *Z* is given by:

$$Z = A \bullet X = \sum_{i=0}^{L-1} C_i(-X_{i,(N-1)} \bullet 2^{N-1} + \sum_{k=0}^{N-2} X_{i,k} \bullet 2^k)$$

$$= -\sum_{i=0}^{L-1} C_i X_{i,(N-1)} \bullet 2^{N-1} + \sum_{k=0}^{N-2} \left[\sum_{i=0}^{L-1} C_i X_{i,k} \right] \bullet 2^k \quad (4)$$

By defining the term R_k as

$$R_k = \sum_{i=0}^{L-1} C_i X_{i,k} \quad (5)$$

Then, we can obtain

$$R_{N-1} = \sum_{i=0}^{L-1} C_i X_{i,N-1} \quad (6)$$

Substituting (5) and (6) in (4), (4) can be written as

$$Z = -R_{N-1} \bullet 2^{N-1} + \sum_{k=0}^{N-2} R_k \bullet 2^k \quad (7)$$

$$= \sum_{k=0}^{N-1} S_k \bullet R_k \bullet 2^k$$

where *S* is defined as the sign of term for $k = 0,1,\cdots, N-2$, $N-1$.

$$S_k = \begin{cases} -1 & k = N-1 \\ 1 & 0 \le k \le N-2 \end{cases} \quad (8)$$

Since $X_{i,k} \in \{0,1\}$, R_k has 2^L possible values for $k = 0,1,\cdots, L-2$, these values can be precomputed and stored in a ROM. Then, (5) can be implemented with a ROM of size 2^L.

The bits of input data ($\{X_{0,k}, X_{1,k},\cdots, X_{i,k}\}$) are used to form the ROM addresses. An arithmetic shifter in the accumulator feedback path is used to form successive scalings with powers of two.

The problem with ROM-based DA is that its ROM size (2^L word) grows exponentially as the order L increases. As the number of inputs and the internal precision becomes large, the ROM-based DA suffers from extremely large ROM requirements.

C. Adder-Based DA

From the arithmetic point of view, the adder-based DA has little difference compared with the ROM-based DA. In ROM-based DA, (4) is obtained by substituting (3) in (1). While one of the two factors in (1), X_i, is changed to two's complement binary format, the other factor, C_i, is led to keep its original format. For adder-based DA, the roles of the two factors are exchanged: X_i keeping its original format and C_i being changed to two's complement binary format.

By substituting (2) in (1), the output *Z* is given by:

$$Z = A \bullet X = \sum_{i=0}^{L-1} X_i(-C_{i,(M-1)} \bullet 2^{M-1} + \sum_{j=0}^{M-2} C_{i,j} \bullet 2^j)$$

$$= -\sum_{i=0}^{L-1} X_i C_{i,(M-1)} \bullet 2^{M-1} + \sum_{j=0}^{M-2} \left[\sum_{i=0}^{L-1} X_i C_{i,j} \right] \bullet 2^j \quad (9)$$

We define term T_j as

$$T_j = \sum_{i=0}^{L-1} X_i C_{i,j} \quad (10)$$

then, (9) can be written as

$$Z = -T_{M-1} \bullet 2^{M-1} + \sum_{j=0}^{M-2} T_j \bullet 2^j \quad (11)$$

$$= \sum_{j=0}^{M-1} S_j \bullet T_j \bullet 2^j$$

Since $C_{i,j}$ is fixed and known, T_j can be realized with adders. Clearly, only the inputs corresponding to nonzero coefficient bits $C_{i,j}$ need to be added. This results in reducing the number of additions in half on average.

The adders with shifts replace the multipliers in the original DA algorithm. The adoption of adders also makes the architecture more hardware efficient. However, the benefits of adoption of adders is not limited to hardware efficiency, it also achieves *N* times speed as faster as ROM-based DA, where *N* is the bitwidth of the input vectors. In ROM-based DA, the vectors are imported serially to generate the ROM addresses for computing R_k terms. Whereas, in adder-based DA, all inputs are fed parallel to the adders for computing T_j regardless how long they are. The time is only consumed in the addition, the longer the inputs are the more time is taken.

Besides the advantages of adder-based DA described above, common terms sharing brings additional advantages

in reducing hardware complexity further.

The ideas behind common term sharing are not new in two's complement binary multiplier, which can be found in [16-20], namely common subexpression sharing or common subexpression elimination. The purposes of common term sharing and common subexpression elimination are to reduce the number of adders. However, the implementation strategies are greatly different for the two methods.

Common subexpression elimination is to find multiple common subexpressions in the coefficient set. These patterns are used in the way that the input signal multiplies each signed-powers-of-two pattern only once. It is also well known that the so-called canonic-signed-digit (CSD) code is the best signed-powers-of-two code for its minimal number of nonzero digits. Obviously, the efficient common subexpression elimination is based on CSD code. For the characteristic of CSD code, changes are made to the order of expression evaluation when common subexpression elimination is implemented. While common term sharing is realized by the multi-level adder arrays without any changes in algorithm expression. The comparison between the two methods targeting the same application will be made in section 7.

Fig. 1 shows an example case for common terms sharing.

$$T_j = \sum_{i=0}^{L-1} X_i C_{i,j}; \quad T_0 = \sum_{i=0}^{L-1} X_i C_{i,0} =$$

$$\begin{array}{l} X_0\,(1\ 1\ 0\ 1) \\ X_1\,(1\ 0\ 1\ 1) \\ X_2\,(1\ 1\ 1\ 0) \\ +)\ X_3\,(0\ 0\ 1\ 1) \\ \hline X_0\ X_0\ 0\ X_0 \\ X_1\ 0\ X_1\ X_1 \\ X_2\ X_2\ X_2\ 0 \\ +)\ 0\ 0\ X_3\ X_3 \end{array}$$

$$= (X_0 + X_1 + X_2)2^3 + (X_0 + X_2)2^2 + (X_1 + X_2 + X_3)2^1 + (X_0 + X_1 + X_3)2^0$$

Fig. 1 Example of adder-based DA

Suppose the input vector and fixed coefficient vector are

$X = [X_0, X_1, X_2, X_3]$

$C_{00} = 1101_b$, $C_{10} = 1011_b$, $C_{20} = 1110_b$, $C_{30} = 0011_b$;

First, substitute C_i and X_i in (1), and then decompose fixed coefficients into bit level. After multiplying input vectors and their corresponding coefficient in bit level, add all terms with the format of power of two. Notice that the additions are taken only at the nonzero bits of coefficients. One finds that there are some common terms between different bit weights: term X_1+X_2 between bit weights 2^3 and 2^1, term X_1+X_3 between bit weights 2^1 and 2^0.

Obviously, the sharing common terms can save the hardware area and reduce the redundant computing to save power consumption. One finds that the diverse common term schemes will result in different amount of resource savings. For the example shown in Fig. 1, seven two-input adders are needed for accumulation without sharing common terms. There are three common term sharing schemes: X_0+X_1 as the common term, X_1+X_2 as the common term, X_0+X_2 and X_1+X_3 as the common terms. In the first two schemes, six two-input adders are needed. Whereas five two-input adders are needed for the third scheme. Therefore, the selection of common terms sharing scheme will determine the final hardware efficiency in the implementation.

It should be emphasized that the example in Fig. 1 is a special case. For all possible 4-input, 4-bits coefficient cases, only few cases have two or more common terms and the cases having one common term are only the part of all possible cases. Is the common terms sharing property available in real cases? Fortunately, the coefficients in the applicable applications are symmetric and regular, which will make the application taking full advantages of common terms. An example for DCT is introduced in the following section. It will be shown that the common terms sharing contributes greatly to the resource savings.

IV. ARCHITECTURE OF RECONFIGURABLE DA

A. Architecture overview

The proposed reconfigurable DA architecture is shown in Fig. 2.

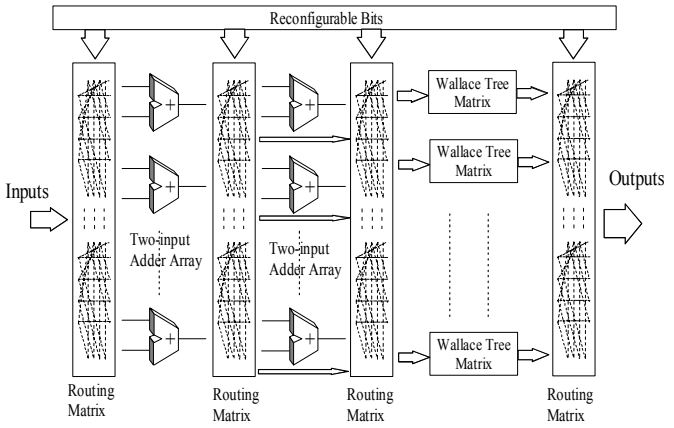

Fig. 2 Architecture of the reconfigurable adder-based DA

It consists of two major parts: the first part is the two-input adder array in two levels which realizes T_j term as defined in (10), and the second part is the parallel Wallace tree matrix which generates the final result Z. The architecture can cope with up to 8 inputs with 9 bits width each.

B. Two-level adder structure

In the two-level adder butterfly structure, the first level consists of up to 12 9-bit adders which work in parallel. The inputs are fed to this level through a routing matrix. The first level outputs, determined by the application, are routed to the next level inputs by another routing matrix. Due to common terms sharing, two or more adders in the second level can share one output form the previous level. The second level adder array consists of up to 24 10-bit adders working in

parallel. Compared with the first level adder array, each adder in the second level is followed with a bypass path which allows the outputs of the first level adder array pass straight to the Wallace tree matrix when the application can be implemented with only one level common terms sharing. The bypass path makes the architecture more flexible by switching between one or two-level adder structure depending on the target application.

C. *Wallace tree matrix*

After one or two-level adder structure, all outputs will be routed to the parallel Wallace tree matrix to generate the final outputs. The delay incurred with a serial addition approach cannot fully satisfy the requirements for some real time applications. With serial addition, only one input is added each cycle. Obviously, a serial computing model does not make full use of the previous level hardware and results.

To avoid timing bottleneck and inefficient use of hardware, a parallel processing approach is adopted in the second part. A structure with 8 parallel Wallace tree blocks provides 8 outputs at once. A traditional Wallace tree and 3:2 compressors are used for each accumulation.

D. *Balance between area and speed*

Different types of addrers are used in different parts of the architecture. For example, full-adders and RCA (Ripple Carry Adder) are adopted in the first part and 3:2 compressors are adopted in Wallace tree. The purpose of this choice is to achieve a balance between area and speed.

There are many different types of adders, but generally they can be divided into four main classes: RCA, CS (Carry Select adder), CLA (Carry Look-ahead Adder) and CSA (Conditional Sum Adder). RCA is the most area efficient but the slowest among the four types. Compared with RCA, other three types adopt extra hardware to speed up processing. In most designs, the use of a parallel architecture is the straightforward method to solve the timing bottleneck. The same also applies to 3:2 compressors. Compared with 3:2 compressor, 4:2 compressor is an optimized structure for speed with the extra hardware cost.

For our architecture, the critical path can be further shortened with some increase in area. However, this is not necessary since with the adoption of parallel adders and Wallace trees in our architecture, the current delay time can meet the requirements of most real time applications. The reconfigurable property of the architecture makes the routing matrix area-consuming. Obviously, the system will become larger if the speed is optimized further by adopting faster adders and compressors.

V. DCT ALGORITHM AND ITS IMPLEMENTATION

DCT is one of the most widely used algorithms in digital signal processing, which removes artificial discontinuities form highly correlated signals. As one of the major operations in current image/video compression, it can be found in JPEG for still picture compression, ITU H.261 and H.263 for video conferencing standard, and ISO MPEG (MPEG-1, MPEG-2, and MPEG-4) for audio, visual compression and communication.

The precision of DCT implementation lies on the coefficient representation when the input vector is given with the fixed precision. To fully support international standard ISO/IEC 14496-2:2004 and IEEE Std 1180–1990, the coefficient precision in our architecture is chosen to be 12 bits.

$$F_0(i) = \begin{vmatrix} 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 1&1&1&1&1&1&1&1\\ 0&0&0&0&0&0&0&0\\ 1&1&1&1&1&1&1&1\\ 1&1&1&1&1&1&1&1\\ 0&0&0&0&0&0&0&0\\ 1&1&1&1&1&1&1&1\\ 0&0&0&0&0&0&0&0\\ 1&1&1&1&1&1&1&1\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0 \end{vmatrix} \qquad F_1(i) = \begin{vmatrix} 0&1&0&1&0&1&0&1\\ 0&1&0&1&0&1&0&1\\ 1&0&0&1&0&1&1&0\\ 1&0&1&0&1&0&1&0\\ 0&1&1&0&1&0&0&1\\ 1&0&1&0&1&0&1&0\\ 1&1&0&1&0&1&0&0\\ 1&0&0&1&0&1&1&0\\ 1&1&0&0&1&1&0&0\\ 1&1&1&0&1&0&0&0\\ 0&0&0&0&1&1&1&1\\ 0&0&0&0&1&1&1&1 \end{vmatrix}$$

Fig. 3 $F_0(i)$ and $F_1(i)$ in 2's complement format

Following the steps of adder-based DA, Eq.(14) can be represented in 2's complement format as shown in Fig. 3, where matrices $F_0(i)$ and $F_1(i)$ are listed as examples.

In theory, 96 (=12*8) terms, which are the summation of eight inputs, are needed for 8-point 1D DCT with 12-bit coefficients based on the adder-based DA. It means that 672 two-input adders are required when implemented directly. However, because of the periodic conjugate symmetry inherent in the DCT, the real implementation consumes just a small part of the theoretic hardware cost. In total, there are 96 terms of eight matrices F_k. Deducting the zero and duplicate terms which need no further calculation, there are one term of 8 inputs and 22 terms of 4 inputs, as shown in Table 1.

Table 1: Unique terms of DCT

	Terms
8 inputs	T(01234567)
4 inputs	T(0123),T(4567),T(0124),T(0145),T(0356),T(0135), T(0246),T(1247),T(2435),T(1357),T(0257),T(0167), T(1346),T(1237),T(3567),T(1457),T(0236),T(1256), T(0347),T(2467),T(2367),T(0456),

As an example, T(0356) represents the summation with inputs X_0, X_3, X_5 and X_6.

As described in section 3, sharing common terms can affect the overall hardware efficiency and power consumption significantly. While considering all different available schemes, the frequency of the terms used is one of the key factors. The scheme with the occurrence of all terms in the same level will reduce the complexity of the routing most. The twelve sharing common terms adopted in our design are:

T(01), T(23), T(45), T(67), T(06), T(35),

T(24), T(17), T(07), T(25), T(16), T(34)

For the selection of 2-input sharing common terms, a selection is made such that each term is used minimum number of times possible in order to reduce the load capacitance at the output nodes. Consider T(0167) as an example. It can be decomposed into T(01)+T(67) or T(07)+T(16). Considering the overall common terms occurrence, T(07)+T(16) is used in our design for the reason that its fan out is lower than the other pair. By sharing common terms, a total of 35 (12+22+1) two-input adders are needed for the 1D DCT, which gives 94.8% reduction in the number of adders compared with the 672 adders required by the theoretic implementation without optimization.

VI. EXPERIMENTAL RESULTS

To verify the functionality of the reconfigurable architecture, an 8 X 1D DCT is implemented. In the realization of DCT, finite accuracy is achieved due to the fixed DA precision. Obviously, more accurate data can be obtained through increasing the precision of the coefficients and the width of the results. This, however, results in larger area and higher power consumption, and adversely affects the computing speed in the adder array.

The requirements of DCT and IDCT hardware implementations are strictly imposed by the various standards, such as ISO/IEC 14496-2:2004 and IEEE Std 1180–1990. A brief summary is given below:

- Image pixel representation: 8 bits for 8*8 DCT
- Input bits for the forward transform: 9 bits
- Coefficients representation: 12 bits
- 1D DCT outputs: 14 bits

With the common terms discussed in the previous section, the 8 X 1D DCT was implemented with the proposed reconfigurable architecture. The architecture was implemented using the Verilog hardware description language. A standard-cell based synthesis and layout was performed with Design Compiler from Synopsys, Inc., targeting the UMC 0.18um CMOS technology library, where the area of a two-input NAND was 12.197 um^2. The power consumption of our reconfigurable architecture was obtained after post-layout simulation by the Synopsys PrimePower. The area of the 8 X 1D DCT was 600929 um^2 and the power consumption was 15.2mW with a 20MHz system clock. The design can run with up to 144MHz (6.93ns) with 112-bits (=14bits×8) outputs which means our architecture can reach up to 16.128Gbps for 1D DCT.

VII. PERFORMANCE EVALUATION

To compare with the performance of common subexpression elimination with CSD code, two implementations from [19, 20] are taken. All the implementations are targeted on 8X8 DCT with bitwidth at 8. The number of required adders is 65 and 130 respectively in [19] and [20]. For our architecture, a total of 35 adders are needed in three levels to obtain all the products. This indicates that our method achieves 46% and 73% reduction respectively compared with existing CSD common subexpression elimination implementations. The figures prove that the adopted strategy is efficient and the scheme selected is optimal.

To evaluate the performance of our architecture, we need to make comparisons with alternative solutions. However, no architecture exists specifically designed for the distributed arithmetic applications. Therefore, several ASIC solutions for 1D DCT with the same throughput will be taken as the reference for evaluating the performance.

As area can often be traded for delay and to eliminate the impact of different technologies, normalized delay-area product [8] was adopted to evaluate our architecture. It defined as the product of the hardware cost (NAND gate count) and normalized average computation time which is the consumption time normalized by the delay of a NAND gate. This is used to evaluate the design performance for area and speed together. The lower the normalized delay-area product of a design is the better the performance of that design. The performances of some existing designs with 12-bit word length of data path are listed in Table 2.

The reconfigurability of proposed architecture is obtained at the cost of time, area, and power consumption. The more applications fitted, the higher is the cost. For a specific function, an ASIC implementation is the most efficient among all implementations, including FPGAs and domain specific reconfigurable architectures. Considering the internal routing network in our design consumes over 80% area of the whole architecture, the normalized delay-area product of our design in Table 2 was divided by 5. It should be emphasized that the figure also includes the impact of routing resource for timing.

Table 2: Performances of some existing designs and ours

Designs	Index
[9]	1330855.61
[10]	2223371.27
[11]	1073023.72
[12]	853112.48
[13]	2005223.00
[8]	1033565.85
Average	1419858.66
Proposed (Scaled)	1241578.80

It can be concluded from the table that our architecture achieves better performance than the average of 6 selected reference designs.

To evaluate the power consumption of our architecture, an ASIC design in [14] was taken as an example, which adopted similar algorithm with ours. The power consumption in [14] is 12.45mW for 1D DCT with ST Microelectronics, hcmos9, 0.12μm technology at 1.5V, 50MHz. Considering dissipated power is approximatively proportional to the square of supply voltage, the power consumption of the design in [14] can be scaled to 7.97mW for 1.2V. Our architecture consumes 7.13mW with UMC 0.13um CMOS technology library at 1.2V, 50MHz. It means the power consumption of our reconfigurable architecture is less than the ASIC design. It should be noted that the power consumption of our architecture includes the power dissipation caused by interconnection network which provides the architecture

powerful reconfigurability.

VIII. SUMMARY

A novel reconfigurable low-power architecture for DA was introduced in this paper. An adder-based DA was adopted in the design. It was shown that the adder-based DA can achieve 94.8% reduction in area in the case of a DCT implementation. Compared with the common subexpression elimination with CSD, up to 73% saving is obtained in hardware resources. The results with the proposed architecture prove its efficiency in terms of area, power and speed.

1D DCT was mapped onto the architecture for the functionality verification and performance evaluation. The experimental data showed that the proposed architecture achieved better performance in area and speed than the average of six selected ASIC designs when the impact of interconnection resource in our architecture was removed. The right policy for trading off area and speed made the architecture consume even less power than the ASIC designs using a similar algorithm.

Besides the good performance in area, power and delay, the general purpose of the design, compared with a domain-specific design, makes the design suitable for inner product computation targeting different applications, such as DCT, DFT, FIR, DHT and so on. The experimental results showed that the proposed reconfigurable architecture could provide an efficient hardware platform for implement various multimedia applications.

IX. REFERENCES

[1] S.A. White, "Application of Distributed Arithmetic to Digital Signal Processing: A Tutorial Review," IEEE Acoustics, Speech, and Signal Processing vol. 6, pp. 4-19, July 1989.

[2] Z. Liu, T. Arslan, S. Khawam, I. Lindsay "A High Performance Synthesisable Unsymmetrical Reconfigurable Fabric For Heterogeneous Finite State Machines," ASP-DAC 2005, pp. 639-642

[3] Hartej Singh, Ming-Hau Lee, Guangming Lu et.al. "MorphoSys: A Reconfigurable Architecture for Multimedia Applications". Proceedings of XI Brazilian Symposium on Integrated Circuit Design, Rio De Janeiro, Oct 98.

[4] Marcos R. Boschetti Alexando et.al. "Techniques and Mechanisms for Dynamic Reconfiguration in an Image Processor" Integrated Circuits and Systems Design, 2002 pp.177-182.

[5] Chang, T.-S.; Chen, C.; Jen, C.-W., "New distributed arithmetic algorithm and its application to IDCT" Volume 146, Issue 4, Circuits, Devices and Systems, 1999 pp:159 – 163

[6] Dae Won Kim; Taek Won Kwon; Jung Min Seo; Jae Kun Yu et.al. "A compatible DCT/IDCT architecture using hardwired distributed arithmetic" ISCAS 2001, pp:457 - 460 vol. 2

[7] Jiun-In Guo, "A new DA-based array for one dimensional discrete Hartley transform" ISCAS 2001, pp:662 - 665 vol. 4

[8] Jiun-In Guo; Rei-Chin Ju; Jia-Wei Chen, "An efficient 2-D DCT/IDCT core design using cyclic convolution and adder-based realization" Circuits and Systems for Video Technology, Volume 14, Issue 4, 2004 pp:416 – 428

[9] S. F. Haiso, W. R. Shiue, and J. M. Tseng, "Design and implementation off a novel linear-array DCT/IDCT processor with complexity of order log2N," IEE Proc. Visions, Images, and Signal Processing, vol. 147, no.5, pp. 400–408, Oct. 2000.

[10] A. Madisetti and A. N.Willson, Jr., "A 100 MHz 2-D 8 8 DCT/IDCT processor for HDTV applications," IEEE Trans. Circuits Syst. Video Technol., vol. 2, pp. 135–146, Apr. 1995.

[11] M. T. Sun, T. C. Chen, and A. M. Gottlieb, "VLSI implementation of a 16 X16 discrete cosine transform," IEEE Trans. Circuits Syst. II, vol. 36, pp. 610–616, Apr. 1989.

[12] D. Slawecki andW. Li, "DCT/IDCT processor design for high data rate image coding," IEEE Trans. Circuits Syst. Video Technol., vol. 2, pp. 135–146, Apr. 1992.

[13] D. W. Kim et al., "A compatible DCT/IDCT architecture using hardwired distributed arithmetic," in Proc. ISCAS, 2001, pp. II-457–II-460.

[14] Ghosh, S.; Venigalla, S.; Bayoumi, M., "Design and implementaion of a 2D-DCT architecture using coefficient distributed arithmetic" VLSI 2005, pp:162 - 166

[15] Pai, A.K.; Benkrid, K.; Crookes, D., "Embedded reconfigurable DCT architectures using adder-based distributed arithmetic" CAMP 2005, pp:81-86

[16] R. Pasˇko, P. Schaumont, V. Derudder, S. Vernalde, and D. Dˇ uracková, " A new algorithm for elimination of common subexpressions," IEEE Trans. Computer-Aided Design, vol. 18, pp. 58–68, Jan. 1999.

[17] M. Martínez-Peiró, E. I. Boemo, and L. Wahammar, "Design of highspeed multiplierless filters using a nonrecursive signed common subexpression algorithm," IEEE Trans. Circuits Syst. II, vol. 49, pp. 196–203, Mar. 2002.

[18] O. Gustafsson and L.Wahammar, "ILP modeling of the common subexpression sharing problem," in Proc. 9th IEEE Int. Conf. Electronic Circuits Systems, vol. 3, Dubrovnik, Croatia, Sept. 2002, pp. 1171–1174.

[19] Macleod, M.D.; Dempster, A.G.," Common subexpression elimination algorithm for low-cost multiplierless implementation of matrix multipliers" Electronics Letters Volume 40, Issue 11, 27 May 2004 Page(s):651 – 652

[20] Tian-Sheuan Chang; Jiun-In Guo; Chein-Wei Jen," Hardware-efficient DFT designs with cyclic convolution and subexpression sharing" IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing, Volume 47, Issue 9, Sept. 2000 Page(s):886 - 892