International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-2 Issue-2, December 2012

An Efficient Methodology for Mapping Algorithms to Scalable Embedded Architectures

Ayman Elnaggar, Mokhtar Aboelaze

***Abstract*— This paper presents a general approach for generating higher order (longer size) multidimensional (m-d) architectures from 2^m lower order (shorter sizes) architectures. The objective of our work is to derive a unified framework and a design methodology that allows direct mapping of the proposed algorithms into embedded reconfigurable architectures such as FPGAs. Our methodology is based on manipulating tensor product forms so that they can be mapped directly into modular parallel architectures. The resulting circuits have very simple modular structure and regular topology.**

***Index Terms*—Reconfigurable Architectures, Recursive algorithms, multidimensional transforms, tensor products, permutation matrices.**

I. INTRODUCTION

This paper proposes an efficient and cost-effective general methodology for mapping multidimensional transforms onto efficient reconfigurable architectures such as FPGAs. The main objective of this paper is to derive a design methodology and recursive formulation for the multidimensional transforms which is useful for the true modularization and parallelization of the resulting computation. Our methodology employs tensor product (or Kronecker products) decompositions and permutation matrices as the main tools for expressing the general framework for multidimensional DSP transforms. We employ several techniques to manipulate such decompositions into suitable recursive expressions which can be mapped efficiently onto reconfigurable FPGAs structures. Our work is based on a non-trivial generalization of the one-dimensional DSP transforms. It has been shown that when coupled with stride permutation matrices, tensor products provide a unifying framework for describing and programming a wide range of fast recursive algorithms for various transform. This unifying framework is suited for parallel processing machines and vector processing machines [6], [10]. Some of the tensor product properties that will be used throughout this paper are [6], [10]:

Manuscript published on 30 December 2012.
***** Correspondence Author (s)
Ayman Elnaggar, Department of Computer Science & Engineering, German University in Cairo, New Cairo City, Egypt.
Mokhtar Aboelaze, Department of Computer Science,York University,Toronto, Canada .

$$AB \otimes CD = (A \otimes B)(C \otimes D) \quad (1)$$

$$(A \otimes B) \otimes C = A \otimes (B \otimes C) \quad (2)$$

If $n = n_1 n_2$, then

$$A_{n_1} \otimes B_{n_2} = P_{n,n_1}(I_{n_2} \otimes A_{n_1})\, P_{n,n_2}(I_{n_1} \otimes B_{n_2}) \quad (3)$$

If $n = n_1 n_2 n_3$, then

$$I_{n_1} \otimes A_{n_2} \otimes I_{n_3} = P_{n,n_1 n_2}(I_{n_1 n_3} \otimes A_{n_2})\, P_{n,n_3} \quad (4)$$

If $2n = n_1 n_2$, then

$$P_{n,2} = P_{n,n_1} P_{n,n_2} \quad (5)$$

Where $\otimes$ denotes the tensor product, I_n is the identity matrix of size n, and $P_{n,s}$, the permutation matrix, is $n \times n$ binary matrix whose entries are zeroes and ones, such that each row or column of has a single 1 entry. If $n = rs$ then $P_{n,s}$ is an $n \times n$ binary matrix specifying an $\frac{n}{s}$-shuffle (or s-stride) permutation. The effect of the permutation matrix $P_{n,s}$ on an input vector X_n of length n is to shuffle the elements of X_n by grouping all the r elements separated by distance s together. The first r element will be $x_0, x_s, x_{2s}, \cdots, x_{(r-1)s}$, the next r elements are $x_1, x_{1+s}, x_{1+2s}, \cdots, x_{1+(r-1)s}$, and so on.

The main result reported in this paper shows that a large two-dimensional (2-d) computation for a given DSP transform on an $n \times n$ input array can be decomposed recursively into three stages as shown in Fig. 1 for the case $n = 4$. The middle stage is constructed recursively from 2^2 parallel (data-independent) blocks each realizing a smaller-size computation of the same DSP transform. The pre-additions and the post-permutations stages serve as "glue" circuits that combine the 2^2 lower order blocks to construct the higher order architecture. We also show that the proposed unified approach can be extended such that an m-d DSP transform can be constructed from 2^m smaller size m-d ones.

Observe that, we have drawn our networks such that data flows from right to left. We chose this convention to show the direct correspondence between the derived algorithms and the proposed reconfigurable architecture.

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

II. A General framework For 1-D Recursive DSP Transforms

In this section, we present a general framework to derive recursive formulations for multidimensional transforms. Given a 1-d DSP algorithm in a matrix-vector form

$$Y_m = T_{m,n}\, X_n \tag{6}$$

Where, $T_{m,n}$ is the transform matrix, X_n and Y_m are the input vector of size n and the output vector of size m, respectively. Then, using sparse matrix factorization approach [9], the matrix $T_{m,n}$ can be factorized so that

$$T_{m,n} = T_1 T_2 \cdots T_k \tag{7}$$

Where, each of the matrices $T_1, T_2, \cdots, T_k$ is sparse. Sparseness implies that either most of the elements of the matrix are zeros or the matrix in the block diagonal form. By applying tensor product property (3) to the block diagonal matrices of equation (7), we have

$$T_{m,n} = (R_i)(I_j \otimes T_{m/2,n/2})(Q_k) \tag{8}$$

Where, Q_k and R_i are the pre- and post-processing glue structure that combine j blocks in parallel of the lower-order transform of size $T_{m/2,n/2}$.

A. The 1-d Linear Convolution

The 1-d linear convolution matrix $C(n)$ of size $n = 2^{\alpha}$, where α is an integer can be written as [5], [6]

$$C(n) = R_n (I_3 \otimes C(n/2)) Q_n . \tag{9}$$

Where,

$$Q_n = (P_{2^{\alpha-1}3,2^{\alpha-2}3})^{\alpha-1}[(I_{2^{\alpha-1}} \otimes A) P_{2^{\alpha},2^{\alpha-1}}].$$

$$R_n = R_{2^{\alpha-k}}(P_{3(2^{\alpha}-1),3}(I_{2^{\alpha}-1} \otimes B) P_{3(2^{\alpha}-1),(2^{\alpha}-1)})$$

$$A = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{bmatrix}, \qquad B = \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & -1 \\ 0 & 0 & 1 \end{bmatrix}$$

In this case $j = 3$ (three parallel blocks of the convolution of smaller size $n/2$)). The realization of the 1-d linear convolution is shown in Fig. 1.

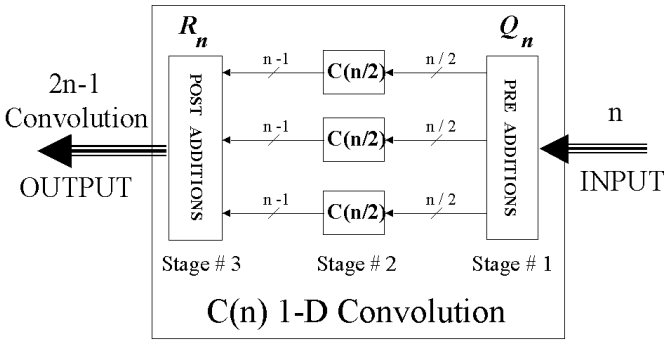

Fig. 1. The realization of the 1-d linear convolution

B. The 1-d DCT

The 1-d DCT T_n of size n can be written as [3], [7]

$$T_n = R_n (I_3 \otimes T_{n/2}) Q_n . \tag{10}$$

Where

$$R_n = P_{n,n/2}(I_{n/2} \oplus L_{n/2}) \quad ,$$

$$Q_n = (I_2 \otimes V_{n/2}^{-1})(I_{n/2} \oplus C_{n/2})(F_2 \otimes I_{n/2}) V_n .$$

$$C_n = diag\left[\frac{1}{2\cos\phi_n}\right],$$

$$\phi_n = \frac{\pi}{2n}(4M+1), M = 0, 1, \cdots, n-1,$$

$$L_m = \begin{bmatrix} 1 & 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & 0 & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 1 \end{bmatrix},$$

$$V_n = (I_{n/2} \oplus J_{n/2})\, P_{n,2}, \qquad F_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix},$$

$I_{n/2}$ is the identity matrix of dimension $n/2$, $\oplus$ is the direct sum operator, $J_{n/2}$ is the exchange matrix of order $n/2$ defined as

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

$$J_{n/2} = \begin{bmatrix} 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 1 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 1 & \cdots & 0 & 0 \\ 1 & 0 & \cdots & 0 & 0 \end{bmatrix},$$

In this case $j = 2$ (two parallel blocks of the DCT of smaller size $n/2$)). The realization of the 1-d DCT is shown in Fig. 2.

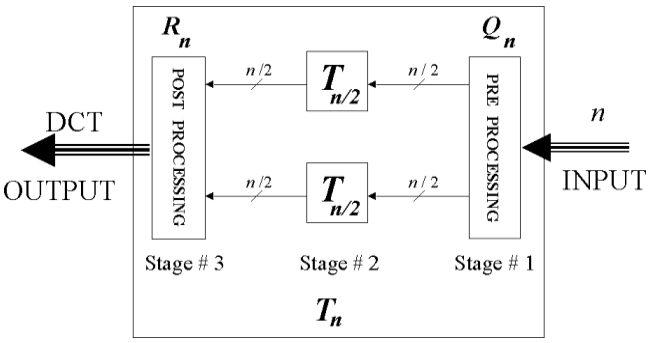

Fig. 2. The realization of the 1-d DCT

C. The 1-d WHT

Our last example is the 1-d WHT. The original 1-d WHT transform matrix is defined as [1], [2]

$$W_n = \begin{pmatrix} W_{n/2} & W_{n/2} \\ W_{n/2} & -W_{n/2} \end{pmatrix}, \quad W_2 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \tag{11}$$

Where, W_2 is the 2-point WHT. Let $k = \log_2 n$, we can write equation (11) in the iterative tensor-product form

$$W_n = W_2 \otimes W_{n/2} = W_2 \otimes W_2 \otimes \cdots \otimes \cdots W_2 = \prod_{i=0}^{k-1} (I_{2^i} \otimes W_2 \otimes I_{2^{k-i-1}}) \tag{12}$$

which using property (4), can be modified to

$$W_n = \prod_{i=0}^{k-1} P_{n,2^{i+1}} (I_{2^{k-1}} \otimes W_2) P_{n,2^{k-i-1}} \tag{13}$$

As an example, we can express W_8 as

$$W_8 = \left[P_{8,2} (I_4 \otimes W_2) P_{8,4}\right]\left[P_{8,4} (I_4 \otimes W_2) P_{8,2}\right] \left[P_{8,8} (I_4 \otimes W_2) P_{8,1}\right] \tag{14}$$

The realization of W_8 is shown in Fig. 3 (a).

Applying property (5) to equation (14) and noting that now the permutations in two adjacent stages can be grouped together into a single permutation, the adjacent permutations $P_{8,2}$ $P_{8,8}$ (from the first and the second stage) will be replaced by the single permutation $P_{8,2}$ and the adjacent permutations $P_{8,4}$ $P_{8,4}$ (from the second and the third stage) will be replaced by the single permutation $P_{8,2}$ as shown in Fig. 3 (b). Similarly, equation (13) can be simplified to

$$W_n = \prod_{i=0}^{k-1} P_{n,2} (I_{2^{k-1}} \otimes W_2) \tag{15}$$

Thus, W_n can be computed by the cascaded product of k similar stages (independent of i) of double matrix products instead of the triple matrix products in equation (8). Alternatively, we can realize (15) by a single block of $P_{n,2} (I_{2^{k-1}} \otimes W_2)$ and take the output after k iterations that allows a hardware saving without slowing down the processing speed and reduction in the hardware size as shown in Fig. 4 for the case $n = 8$.

It should be mentioned that we have applied property (5) to reduce the shuffling inherited in the original WHT algorithm to allow a uniform hardware blocks as shown in Fig. 3 (b). We haven't modified the original complexity of the WHT that are centered in the W_2 blocks as shown in Fig. 3 and Fig. 4.

Applying property (1), equation (12) can be modified to

$$\begin{aligned} W_n &= W_2 \otimes W_{n/2} = I_2 W_2 \otimes W_{n/2} I_{n/2} \\ &= (I_2 \otimes W_{n/2}) (W_2 \otimes I_{n/2}) \\ &= (I_2 \otimes W_{n/2}) Q_n \end{aligned} \tag{16}$$

Where, $Q_n = (W_2 \otimes I_{n/2})$ (17)

Equation (16) represents the two-stage recursive tensor product formulation of the 1-d WHT (in this case $j = 2$) in which the first stage is the pre-additions (Q_n), followed by the second stage of the core computation $(I_2 \otimes W_{n/2})$ that consists of a parallel blocks of two identical smaller WHT computations each of size $n/2$ as shown in Fig. 5.

III. A GENERAL FRAMEWORK FOR 2-D RECURSIVE DSP TRANSFORMS

For a 2-d input data, X_{n_1,n_2}, of size $n_1 \times n_2$, and a separable 2-d transform, T_{n_1,n_2}, we can write the output, Y_{n_1,n_2}, in the form [11]

$$Y_{n_1,n_2} = Y_{n_1,n_2} X_{n_1,n_2} \tag{18}$$

where, X_{n_1,n_2} and Y_{n_1,n_2} are the input and output column-scanned vectors, respectively. For separable matrices, the 2-d transform matrix T_{n_1,n_2} can be written in the tensor product form as [9]

$$T_{n_1,n_2} = T_{n_1} \otimes T_{n_2} \tag{19}$$

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

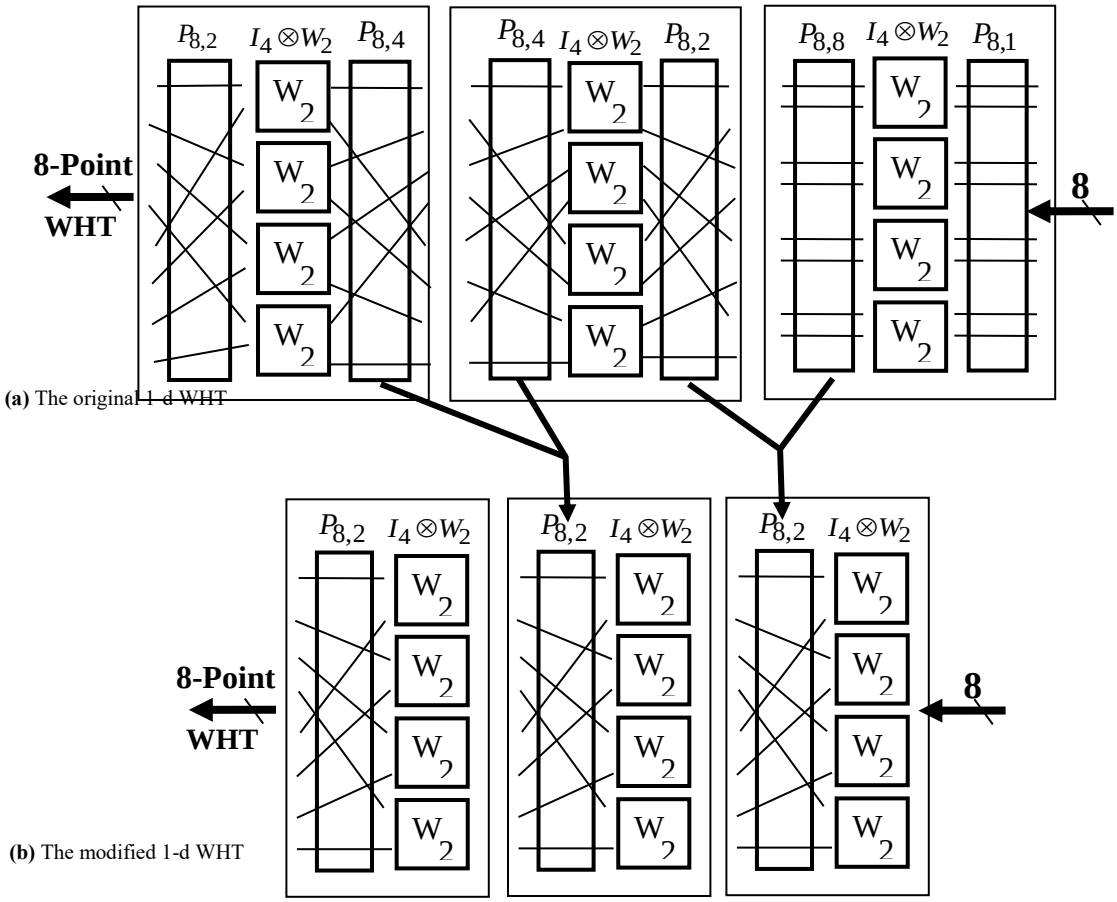

Fig. 3. The realization of the 1-d WHT of size 8 (W_8)

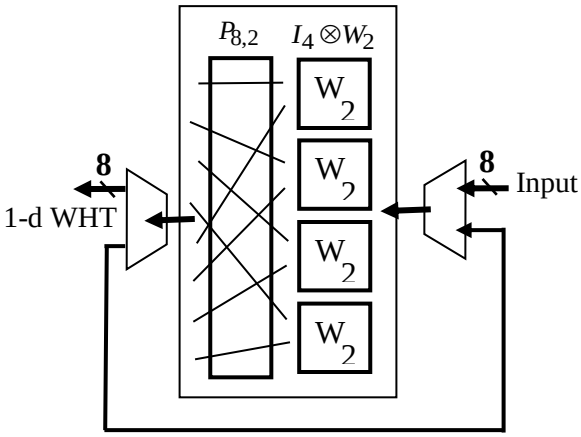

Fig. 4. The realization of the iterative 1-d WHT of size 8 (W_8)

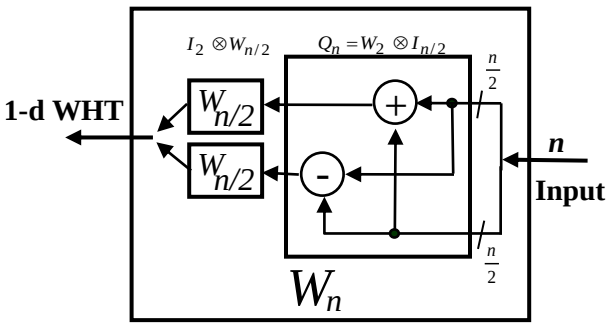

Fig. 5. The realization of the recursive 1-d WHT

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

Where T_{n_1} and T_{n_2} are the row and column 1-d transforms, respectively as defined in (8). By replacing T_{n_1} and T_{n_2} by their corresponding values from equation (8) and applying properties (1) to (4) to derive the 2-d recursive form

$$T_{n_1,n_2} = (\tilde{R}_{n_1,n_2})(I_{j^2} \otimes T_{n_1/2,n_2/2})(\tilde{Q}_{n_1,n_2}) \quad (20)$$

Where, $\tilde{Q}_{n_1,n_2}$ and $\tilde{R}_{n_1,n_2}$ are the 2-d pre- and post-processing glue structure, respectively that combine j^2 of the lower-order (smaller size) 2-d transform $T_{n_1/2,n_2/2}$ of dimension $n_1/2 \times n_2/2$.

A. The 2-d convolution

Let $n_1 = 2^{\alpha 1}$ and $n_2 = 2^{\alpha 2}$. Pratt [9] has shown that for an $n_1 \times n_2$ input data image, the 2-d convolution output is given by

$$q = C_{n_1,n_2} f \quad (21)$$

Where, C_{n_1,n_2} is the 2-d convolution transform matrix; and q and f are the output and input column-scanned vectors, respectively of size $n = n_1 n_2$. Pratt has also shown that, for separable transforms, the matrix C_{n_1,n_2} can be decomposed into the tensor form

$$C_{n_1,n_2} = C(n_1) \otimes C(n_2) \quad (22)$$

Where, $C(n_1)$ and $C(n_2)$ represent row and column 1-d convolution operators on f, respectively, as defined in (8) and (9). From (9) and (22), we can express the 2-d convolution matrix as a function of 1-d convolutions as follows [4]

$$C_{n_1,n_2} = [R_1 (I_3 \otimes C(n_1/2) Q_1] \otimes [R_2 (I_3 \otimes C(n_2/2) Q_2] \quad (23)$$

Applying property (1), leads to

$$C_{n_1,n_2} = [(R_1 \otimes R_2)((I_3 \otimes C(n_1/2)) \otimes ((I_3 \otimes C(n_2/2))(Q_1 \otimes Q_2)], \quad (24)$$
$$= \tilde{R} \tilde{C}_{n_1,n_2} \tilde{Q}.$$

Where,

$$\tilde{R} = (R_1 \otimes R_2),$$
$$\tilde{C}_{n_1,n_2} = (I_3 \otimes C(n_1/2)) \otimes (I_3 \otimes C(n_2/2)), \quad (25)$$
$$\tilde{Q} = (Q_1 \otimes Q_2)].$$

Note that the matrix $\tilde{C}_{n_1,n_2}$ contains the 1-d convolutions matrices $C(n_1/2)$ and $C(n_2/2)$ in an involved tensor product expression. By applying property (2), we can write (24) as

$$\tilde{C}_{n_1,n_2} = ((I_3 \otimes C(n_1/2) \otimes I_3) \otimes C(n_2/2)) \quad , \quad (26)$$

Applying property (4), yields to

$$\tilde{C}_{n_1,n_2} = (P_{9(n_1-1),3(n_1-1)} (I_9 \otimes C(n_1/2) P_{9(n_1/2),3}) \otimes C(n_2/2)), \quad (27)$$

Since the convolution matrix $C(n/2)$ is of dimension $[(n-1) \times n/2]$, we can write $C(n_2/2)$ as

$$C(n_2/2) = I_{n_2-1} . C(n_2/2) . I_{n_2/2} \quad (28)$$

Substituting (21) in (20) and applying property (1),

$$\tilde{C}_{n_1,n_2} = (P_{9(n_1-1),3(n_1-1)} (I_9 \otimes C(n_1/2) P_{9(n_1/2),3}) \otimes (I_{n_2-1} . C(n_2/2) . I_{n_2/2})$$
$$= (P_{9(n_1-1),3(n_1-1)} \otimes I_{n_2-1}) (I_9 \otimes C(n_1/2) \otimes C(n_2/2)) (P_{9(n_1/2),3} \otimes I_{n_2/2}). \quad (29)$$

Now, substituting (29) in (24) gives

$$C_{n_1,n_2} = \tilde{\tilde{R}} (I_9 \otimes C_{n_1/2,n_2/2}) \tilde{\tilde{Q}} \quad , \quad (30)$$

Where, $C_{n_1/2,n_2/2} = C(n_1/2) \otimes C(n_2/2)$ is the lower order 2-d convolution matrix for an $n_1/2 \times n_2/2$ input image, $\tilde{\tilde{Q}} = (P_{9(n_1/2),3} \otimes I_{n_2/2})(Q_1 \otimes Q_2)$ and $\tilde{\tilde{R}} = (R_1 \otimes R_2)(P_{9(n_1-1),3(n_1-1)} \otimes I_{n_2-1})$ are the new 2-d pre- and post-additions, respectively. Equation (30) represents the recursive 2-d convolution algorithm. In this case we use 9 ($j^2 = 3^2 = 9$) of the lower-order $C_{n_1/2,n_2/2}$ convolution blocks in parallel to generate the higher order C_{n_1,n_2} convolution as shown before in Fig. 6.

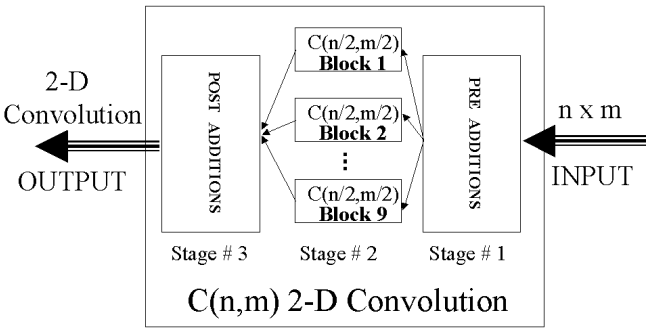

Fig. 6. The realization of the recursive 2-d convolution

B. The 2-d DCT

Since the DCT matrix is separable, the 2-d DCT for an image of dimension $n_1 \times n_2$ can be computed by a stage of n_2 parallel 1-d DCT computations on n_1 points each, followed by another stage of n_1 parallel 1-d DCT computations on n_2 points each. This can be represented by the matrix-vector form [11]

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

$$X = T_{n_1,n_2}\, x, \qquad (31)$$

Where T_{n_1,n_2} is the 2-d DCT transform matrix for an $n_1 \times n_2$ image, *X* and *x* are the output and input column-scanned vectors, respectively. By substituting (10) in (31), we have

$$X = (T_{n_1} \otimes T_{n_2})\, x. \qquad (32)$$

By further manipulation of equation (32) in a similar way to that we did to (23) of the 2-d convolution, we can write (23) as [3]

$$X = (\tilde{R}_{n_1,n_2}\ (I_4 \otimes T_{n_1/2,n_2/2})\, \tilde{Q}_{n_1,n_2})\, x \qquad (33)$$

Where, $\tilde{Q}_{n_1,n_2} = (P_{2n_1,2} \otimes I_{n_2/2})\, Q_{n_1,n_2}$, and $\tilde{R}_{n_1,n_2} = R_{n_1,n_2}\, (P_{2n_1,n_1} \otimes I_{n_2/2})$.

Equation (33) represents the truly recursive 2-d DCT in which $\tilde{Q}_{n_1,n_2}$ and $\tilde{R}_{n_1,n_2}$ are the pre- and post-processing glue structures, respectively, that combine 2^2 ($j^2 = 2^2 = 4$) identical lower-order 2-d DCT modules each of size $n_1/2 \times n_2/2$ in parallel, to construct the higher order 2-d DCT of size $n_1 \times n_2$.

IV. A GENERAL FRAMEWORK FOR M-D RECURSIVE DSP TRANSFORMS

We can extend the steps in deriving recursive formulae of the 1-d and the 2-d transforms to the multidimensional case. For an m-d transform T_{n_i} ,The general form will be

$$\overset{m}{\underset{i=1}{\otimes}} T_{n_i} = \hat{R}\,(I_{j^m} \overset{m}{\underset{i=1}{\otimes}} T_{n_i/2})\,\hat{Q} \qquad (34)$$

Where, $\hat{Q}$ and $\hat{R}$ are the m-d pre- and post-processing glue structures that combine j^m parallel blocks of the lower-order m-d transforms of size $T_{n_i/2}$.

A. *The m-d WHT*

We can extend the 2-d WHT derivation to the *m*-d case. From (12) and (34), the *m*-d WHT can be written in the tensor product form

$$\begin{aligned} X &= (W_{n_1} \otimes W_{n_2} \otimes \cdots \otimes W_{n_m})\, x \\ &= (\overset{m}{\underset{i=1}{\otimes}} W_{n_i})\, x \end{aligned} \qquad (35)$$

Where, $(W_{n_1} \otimes W_{n_2} \otimes \cdots \otimes W_{n_m})$ is the *m*-d WHT transform matrix for an *m*-d input, W_{n_i} is the 1-d WHT coefficient matrix for an input vector of length n_i as defined in (16), *X* and *x* are the output and input column-scanned vectors, respectively.

Using properties (1) to (4), we can write (35) in the form [2]

$$X = [\,\hat{R}\,(I_{2^m} \overset{m}{\underset{i=1}{\otimes}} W_{n_i/2})\,\hat{Q}]\ \ x \quad , \qquad (36)$$

Where $\overset{m}{\underset{i=1}{\otimes}} W_{n_i/2}$ is the lower order *m*-d WHT and

$$\begin{aligned} \hat{Q} &= \left(\prod_{i=1}^{m-1}\left(P_{u_1,u_2} \overset{i}{\underset{k=1}{\otimes}} I_{u_3}\right)\right)\left(\overset{m}{\underset{i=1}{\otimes}} Q_i\right) \quad , \\ \hat{R} &= \prod_{i=1}^{m-1}\left(P_{w_1,w_2} \overset{m-1}{\underset{l=i}{\otimes}} I_{w_3}\right) \quad , \end{aligned} \qquad (37)$$

$$u_1 = 2\prod_{j=1}^{m-i} n_j \quad , \quad u_2 = 2\,, \quad u_3 = \frac{1}{2}\prod_{j=1}^{i}(n_{m-j+1}),$$

$$w_1 = \frac{1}{2}\prod_{k=1}^{i}(n_k),\ \ w_2 = \prod_{k=1}^{i}(n_k),\ \ w_3 = \frac{1}{2}\prod_{j=i+1}^{i}(n_j)\,,$$

Q_i is the 1-d pre-processing as defined by (17). Equation (37) extends our results by showing that a large *m*-d WHT can be computed from a single stage of smaller *m*-d WHTs.

V. CONCLUSIONS

In this paper, we presented a general approach for decomposing higher order (longer size) multidimensional (*m*-d) architectures from 2^m lower order (shorter sizes) architectures. We have shown several examples for the 1-d and 2-d common transforms such as linear convolution, DCT, and WHT. We have extended our results to cover the m-d case as well. The objective of our work was to derive a unified framework and a design methodology that allows direct mapping of the proposed algorithms into reconfigurable architectures. The resulting circuits have very simple modular structure and regular topology that can be mapped directly to FPGAs.

REFERENCES

[1] A. E. Cetin, O. N. Gerek, and S. Ulukus, "Block Wavelet Transforms for Image Coding," *IEEE Trans. on Circuits and systems for Video Technology*, Vol. 3, pp. 433-435, 1993.

[2] A. Elnaggar, Mokhtar Aboelaze, "A Scalable Formulation for 2-D WHT," *Proc. of the IEEE International Symposium on Circuits and Systems (ISCAS' 2003)*, pp IV484-IV487, Thailand, May 2003.

[3] A. Elnaggar, H. M. Alnuweiri, " A New Multi-Dimensional Recursive Architecture for Computing The Discrete Cosine Transform," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 10, No. 1, pp. 113-119, February 2000.

[4] A. Elnaggar and M. Aboelaze, "An Efficient Architecture for Multi-Dimensional Convolution," *IEEE Trans. on Circuits and Systems II*, Vol. 47, No. 12, pp. 1520-1523, 2000.

[5] A. Elnaggar and M. Aboelaze, "A Modified Shuffle Free Architecture for Linear Convolution," *IEEE Trans. on Circuits and Systems II*, Vol. 48, No. 9, pp. 862-866, 2001.

[6] J. Granata, M. Conner, R. Tolimieri, "A Tensor Product Factorization of the Linear Convolution Matrix", *IEEE Trans on Circuits and Systems*, Vol. 38, p. 1364--6, 1991.

[7] H. S. Hou, "A Fast Recursive Algorithm for Computing the Discrete Cosine Transform," *IEEE Trans. On ASSP*, Vol. Assp-35, No. 10, 1987.

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

[8] K. R. Rao, P. Yip, "Discrete Cosine Transform: Algorithms, Advantages, and Applications," Academic Press, 1990.
[9] W. K. Pratt, *Digital Image Processing*, John Wiley & Sons, Inc., 1991.
[10] R. Tolimieri, M. An, C. Lu, *Algorithms for Discrete Fourier Transform and Convolution*, Springer-Verlag, New York 1989.
[11] Chi-Li Yu, et. al, "Architecture for 2D Discrete Fourier Transform Based on 2D Decomposition for Large-sized Data", *Springer Science, Business Media, LLC* 2010.