A NEW FLEXIBLE ARCHITECTURE FOR VARIABLE LENGTH DCT TARGETING SHAPE-ADAPTIVE TRANSFORM

Thuyen Le and Manfred Glesner

Institute of Microelectronic Systems
Darmstadt University of Technology
Karlstr. 15, 64283 Darmstadt, Germany
Thuyen.Le@mes.tu-darmstadt.de

ABSTRACT

Shape-assisted block-based texture coding methodologies such as the shape-adaptive DCT raise the need for an architecture which can perform efficiently the transform of variable length N. This paper presents an 1D DCT architecture satisfying the given requirement in terms of scalability and modularity for $2 \le N \le 8$. The architecture employs a Canonical-Signed-Digit serial multiplication to reduce hardware resources and requires only one multiplier to perform the final scaling. A proposed algorithm searching for an optimal assignment of cosine factors leads to a resource saving of about 12% for the multiplication blocks if Carry-Ripple-Adders are assumed to be used. Different area and speed requirements are possible since only feed-forward paths are involved and easily pipelined. The architecture represents a trade-off between time-recursive, fully modular but operation non-efficient structure and multiplication efficient but irregular and fixed-length implementation.

1. INTRODUCTION

The block-based DCT is widely used in MPEG-1, MPEG-2, H.261 and H.263 standards. Among various transform techniques for compression, DCT is the most popular and effective one in practical applications. Recently, a lot of research has been carried out toward the coding of images for low bit-rate applications. The separation of the video content into objects has become an emerging key technique in image coding algorithm for the content-based manipulation, scalability and interactivity. These techniques consider usually the coding of texture information in arbitrarily shaped objects instead of fixed-size blocks. Combining the transmission of objects and their shape information to the decoder in a shape-assisted block-based texture coding, the efficiency can be improved and coding errors inherent of block-based DCT schemes such as mosquito and blocking artifacts can be avoided [1, 2]. An extension of the baseline shape-adaptive DCT algorithm with a technique for global background motion estimation and compensation is described in [3] which significantly improves the compression efficiency of suitable video sequences compared to standard MPEG coding schemes.

Since the object is arbitrarily shaped within 8×8 image blocks a direct implementation of the 2D shape-adaptive DCT is rather difficult. Due to their reusability and simplicity in hardware realization the row-column decomposition is adopted. Fast algorithms for 1D DCT exploit the symmetry of the cosine function for the matrix factorization to reduce substantially the number of multiplications. In [4], a version with 11 multiplications and 29 additions (subtractions) for a fixed length $N = 8$ has been introduced. However, the fast flow graph results in an irregular structure with complex routing requirements due to resource sharing. In addition, such algorithms exhibit many stages of multiplication and accumulation which effects the numerical accuracy of the algorithm through truncation/rounding in each stage. Consequently, most of these operation-count efficient algorithm require wider multiplications to compensate for precision loss.

For a VLSI implementation, regularity and scalability are very important issues. Time-recursive structure has shown to be an effective way to generate a regular and parallel VLSI structure [5]. In [6] this architecture has been applied to the shape-adaptive DCT due to its scalability to adapt to arbitrary transform size requiring $\mathcal{O}(N)$ time and hardware complexity. A DCT generator for VLSI synthesis based on this architecture was proposed in [7]. However, the second-order recursive structure has the disadvantage of numerical inaccuracy since the computation for one DCT coefficient involves N multiplications in total. In terms of computational requirements, this does not represent the best solution.

The purpose of our paper is to investigate a suitable architecture in terms of scalability and computational requirements to perform a 1D DCT for variable length N. Since the application of this architecture is the 2D shape-adaptive DCT in block-based coding schemes, we restrict ourselves to $2 \le N \le 8$ ($N = 1$ represents a trivial case) but other values can be easily added without altering the architecture. The proposed architecture involves only one multiplication in each feed-forward path (if omitting the final scaling), thus overcomes the problem of numerical inaccuracy found in time-recursive structure. Moreover, the multiplication is performed in a digit-serial manner using the Canonical-Signed-Digit (CSD) representation of the cosine factor. Since the assignment of the cosine factors to different multiplication blocks is arbitrary, we have proposed a search procedure to look for an optimal assignment which yields the minimum hardware cost. An architecture named Selective Coefficient DCT Module presented in [8] can be considered as a special case for $N = 8$ of the architecture discussed in this paper.

2. ALGORITHM FOR VARIABLE LENGTH DCT

The shape-adaptive DCT was first proposed by [2]. It is based on the normal DCT and thus requires the same computation complexity. The method relies on the basic concept of dividing the 8×8 pel block into two regions, foreground and background. Pels that belong to the object constitute to the foreground region. The purpose is to encode the two segments separately employing a shape-adaptive DCT. To perform the separable 2D shape-adaptive DCT the columns are first shifted and aligned to the upper border of the reference 8×8 block. Depending on the length ($2 \leq N \leq 8$) of each column, a 1D N-point DCT for each column is calculated as

$$y_N(k) \quad = \quad C(k) \sum_{n=0}^{N-1} x(n) \cdot cos \frac{(2n+1)k\pi}{2N} \qquad (1)$$

where $k = \{0, 1, \dots, N-1\}$ and $y_N(k)$ denotes the k-th DCT coefficient. $x(n)$ represents the data, $C(0) = \frac{1}{\sqrt{N}}$ and $C(k) = \sqrt{\frac{2}{N}}$ otherwise. Then, the rows are shifted to the left border and a horizontal 1D DCT of each row is again performed. Since the shape-adaptive transform is considered here, N varies for each row and column and is not fixed as in the classical case.

Eq. 1 is usually written as a matrix-vector product $\mathbf{y}_N = \mathbf{C}_N \cdot \mathbf{x}_N$ where $\mathbf{C}_N$ is a $N \times N$ matrix with cosine factors as entries, $\mathbf{x}_N = [x(0) \cdots x(N-1)]^T$ and $\mathbf{y}_N = [y_N(0) \cdots y_N(N-1)]^T$. Because the cosine function is periodical the row of $\mathbf{C}_N$ can either take the symmetric or antisymmetric form:
$[c_0 \quad c_1 \quad c_2 \ \pm c_2 \ \pm c_1 \ \pm c_0]$ for N even, e.g.$N = 6$,
$[c_0 \quad c_1 \quad c_2 \quad c_3 \ \pm c_2 \ \pm c_1 \ \pm c_0]$ for N odd, e.g.$N = 7$.

Exploiting this symmetry, as done in most fast algorithm the number of multiplications for one DCT coefficient is reduced to $\left\lfloor \frac{N}{2} \right\rfloor$. Further inherent symmetry of the matrix is available but will lead to irregular structure and is not considered in our approach. For odd N, we observed that the associated midpoint cosine factor of the symmetry can take on one of the values $\{-1, 0, 1\}$ depending on k. We consider this term as a correcture term in our matrix formulation:

$$y_N(k) = \sqrt{\frac{2}{N}} \cdot [\mathbf{S}_N \cdot \mathbf{F} \cdot \mathbf{P}_{k,N} \cdot \mathbf{d}_N(k) + \alpha(N) \cdot E_N(k)] \quad (2)$$

Defining $\gamma(n) = cos(n\pi)$, the matrix $\mathbf{F}$ of size 10×4 is given in an initial columnwise assignment as ($-$ for "don't care" value)

$$\mathbf{F} = \begin{bmatrix} \gamma(1/4) & \gamma(1/4) & \gamma(1/4) & \gamma(1/4) \\ \gamma(1/3) & \gamma(1/3) & \gamma(0) & - \\ \gamma(1/5) & \gamma(2/5) & - & - \\ \gamma(1/6) & \gamma(1/6) & \gamma(1/2) & - \\ \gamma(1/7) & \gamma(2/7) & \gamma(3/7) & - \\ \gamma(1/8) & \gamma(1/8) & \gamma(3/8) & \gamma(3/8) \\ \gamma(1/10) & \gamma(3/10) & - & - \\ \gamma(1/12) & \gamma(1/4) & \gamma(5/12) & - \\ \gamma(1/14) & \gamma(3/14) & \gamma(5/14) & - \\ \gamma(1/16) & \gamma(3/16) & \gamma(5/16) & \gamma(7/16) \end{bmatrix} \qquad (3)$$

Since the assignment of γ in $\mathbf{F}$ is arbitrary, the permutation matrices $\mathbf{P}_{k,N}$ of size 4×4 select the correct elements in $\mathbf{d}_N$ for the multiplication. Note that $\mathbf{P}_{k,N}$ depends on the assignment of γ in $\mathbf{F}$. For e.g. $N = 7$ and the given $\mathbf{F}$ in Eq. 3, we have

$$\mathbf{P}_{\{0,1\},7} = \begin{bmatrix} 1 & 0 & 0 & - \\ 0 & 1 & 0 & - \\ 0 & 0 & 1 & - \\ - & - & - & - \end{bmatrix}, \quad \mathbf{P}_{2,7} = \begin{bmatrix} 1 & 0 & 0 & - \\ 0 & 0 & 1 & - \\ 0 & 1 & 0 & - \\ - & - & - & - \end{bmatrix}$$
$$\mathbf{P}_{3,7} = \begin{bmatrix} 0 & 0 & 1 & - \\ 1 & 0 & 0 & - \\ 0 & 1 & 0 & - \\ - & - & - & - \end{bmatrix}, \quad \mathbf{P}_{4,7} = \begin{bmatrix} 0 & 1 & 0 & - \\ 1 & 0 & 0 & - \\ 0 & 0 & 1 & - \\ - & - & - & - \end{bmatrix} \qquad (4)$$
$$\mathbf{P}_{5,7} = \begin{bmatrix} 0 & 1 & 0 & - \\ 0 & 0 & 1 & - \\ 1 & 0 & 0 & - \\ - & - & - & - \end{bmatrix}, \quad \mathbf{P}_{6,7} = \begin{bmatrix} 0 & 0 & 1 & - \\ 0 & 1 & 0 & - \\ 1 & 0 & 0 & - \\ - & - & - & - \end{bmatrix}$$

The column vector

$$\mathbf{d}_N(k) = [D_0(k,N) \ D_1(k,N) \ D_2(k,N) \ D_3(k,N)]^T \qquad (5)$$

combines the data samples $x(n)$ according to

$$D_0(k,N) = \quad x(0) + (-1)^k x(N-1) \quad : \ 2 \leq N \leq 8$$
$$D_1(k,N) = \begin{cases} 0 & : \ 2 \leq N < 4 \\ (-1)^{\left\lfloor \frac{k+1+\lfloor N/8 \rfloor}{\lfloor N/2 \rfloor + 1} \right\rfloor} \cdot & \\ \{x(1) + (-1)^k x(N-2)\} & : \ 4 \leq N \leq 8 \end{cases}$$
$$D_2(k,N) = \begin{cases} 0 & : \ 2 \leq N < 6 \\ (-1)^{\left\lfloor \frac{n+\lfloor N/7 \rfloor}{\lceil N/3 \rceil} \right\rfloor} \cdot & \\ \{x(2) + (-1)^k x(N-3)\} & : \ 6 \leq N \leq 8 \end{cases}$$
$$D_3(k,N) = \begin{cases} 0 & : \ 2 \leq N < 8 \\ (-1)^{\left\lfloor \frac{n}{2} \right\rfloor} \cdot & \\ \{x(3) + (-1)^k x(N-4)\} & : \ N = 8 \end{cases} \qquad (6)$$

The product $\mathbf{S}_{k,N} \cdot \mathbf{F}$ is the process of selecting the correct γ for the multiplication. $\mathbf{S}_{k,N}$ represents here the k-th row of the selection matrix $\mathbf{S}_N$ which is a $N \times 10$ matrix defined as

$$\mathbf{S}_2 = \begin{bmatrix} 1\,0\,0\,0\,0\,0\,0\,0\,0\,0 \\ 1\,0\,0\,0\,0\,0\,0\,0\,0\,0 \end{bmatrix} \quad \mathbf{S}_3 = \begin{bmatrix} 1\,0\,0\,0\,0\,0\,0\,0\,0\,0 \\ 0\,0\,0\,1\,0\,0\,0\,0\,0\,0 \\ 0\,1\,0\,0\,0\,0\,0\,0\,0\,0 \end{bmatrix}$$
$$\mathbf{S}_4 = \begin{bmatrix} 1\,0\,0\,0\,0\,0\,0\,0\,0\,0 \\ 0\,0\,0\,0\,0\,1\,0\,0\,0\,0 \\ 1\,0\,0\,0\,0\,0\,0\,0\,0\,0 \\ 0\,0\,0\,0\,0\,1\,0\,0\,0\,0 \end{bmatrix} \quad \mathbf{S}_5 = \begin{bmatrix} 1\,0\,0\,0\,0\,0\,0\,0\,0\,0 \\ 0\,0\,0\,0\,0\,0\,1\,0\,0\,0 \\ 0\,0\,1\,0\,0\,0\,0\,0\,0\,0 \\ 0\,0\,0\,0\,0\,0\,1\,0\,0\,0 \\ 0\,0\,1\,0\,0\,0\,0\,0\,0\,0 \end{bmatrix}$$
$$\mathbf{S}_6 = \begin{bmatrix} 1\,0\,0\,0\,0\,0\,0\,0\,0\,0 \\ 0\,0\,0\,0\,0\,0\,0\,1\,0\,0 \\ 0\,0\,0\,1\,0\,0\,0\,0\,0\,0 \\ 1\,0\,0\,0\,0\,0\,0\,0\,0\,0 \\ 0\,1\,0\,0\,0\,0\,0\,0\,0\,0 \\ 0\,0\,0\,0\,0\,0\,0\,1\,0\,0 \end{bmatrix} \quad \mathbf{S}_7 = \begin{bmatrix} 1\,0\,0\,0\,0\,0\,0\,0\,0\,0 \\ 0\,0\,0\,0\,0\,0\,0\,0\,1\,0 \\ 0\,0\,0\,0\,1\,0\,0\,0\,0\,0 \\ 0\,0\,0\,0\,0\,0\,0\,0\,1\,0 \\ 0\,0\,0\,0\,1\,0\,0\,0\,0\,0 \\ 0\,0\,0\,0\,0\,0\,0\,0\,1\,0 \\ 0\,0\,0\,0\,1\,0\,0\,0\,0\,0 \end{bmatrix}$$
$$\mathbf{S}_8 = \begin{bmatrix} 1\,0\,0\,0\,0\,0\,0\,0\,0\,0 \\ 0\,0\,0\,0\,0\,0\,0\,0\,0\,1 \\ 0\,0\,0\,0\,0\,1\,0\,0\,0\,0 \\ 0\,0\,0\,0\,0\,0\,0\,0\,0\,1 \\ 1\,0\,0\,0\,0\,0\,0\,0\,0\,0 \\ 0\,0\,0\,0\,0\,0\,0\,0\,0\,1 \\ 0\,0\,0\,0\,0\,1\,0\,0\,0\,0 \\ 0\,0\,0\,0\,0\,0\,0\,0\,0\,1 \end{bmatrix} \qquad (7)$$

The mentioned correcture termn $\alpha(N) \cdot E_N(k)$ to account for odd N is given as

$$\alpha(N) = \begin{cases} 0 : N = 2, 4, 6, 8 \\ 1 : N = 3, 5, 7 \end{cases}$$
$$E_N(k) = \frac{(-1)^{\lfloor k/2 \rfloor}}{2} \left\{ x\left(\frac{N-1}{2}\right) + (-1)^k x\left(\frac{N-1}{2}\right) \right\} \qquad (8)$$

3. ARCHITECTURE FOR VARIABLE LENGTH DCT

The formulation of Eq. 2 is advantageous since it results in a modular architecture. In Fig. 1, the block diagram of the derived architecture is shown. First, the generation of $\mathbf{d}_N(k)$ is performed according to a given N and k which can be done by an adder/-subtractor and some glue logic. The switch in the middle of the architecture is a non-blocking network to implement any possible input-ouput mapping as specified by $\mathbf{P}_{k,N}$. In the Cosine Factor Multiplication Block (CFMB), the multiplication of the input with selected cosine factors given by $\mathbf{S}_{k,N} \cdot \mathbf{F}$ takes place. The sum of all CFMBs and a possible correcture term are added and scaled by $\sqrt{2/N}$. Scalability to more or other lengths is preserved by just adding more CFMBs, extending the switch matrix and the pairwise addition/subtraction module at the input. Except for the switch, other components in the architecture requires only local connections and are fully regular.

For the odd case, the add/sub module can be configured to generate the $E_N(k)$ term on the busses D_1, D_2 or D_3 for $N = 3, 5, 7$, respectively. In all odd cases, the CFMB-3 module is unused. The switch can be configured so that the term $E_N(k)$ is directed into CFMB-3. A trivial multiplication by $\alpha(N) \in \{-1, 0, 1\}$ can be performed here to take advantage or resource sharing. Effectively, this means that we have to extend the set of cosine factors γ's in CFMB-3 with $\gamma(0)$ and $\gamma(1/2)$ which does not represent any major modifications.

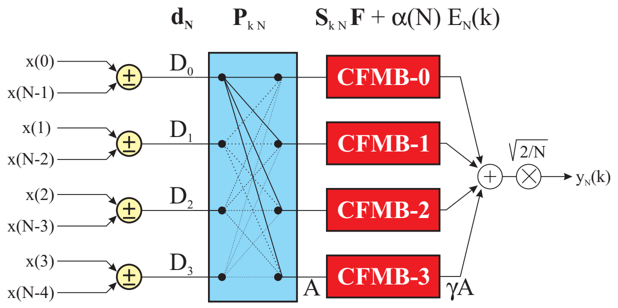

Figure 1: Block diagram of the proposed architecture.

3.1. Canonical-Signed-Digit Serial Multiplication

Since each CFMB has to perform the multiplication of the permuted data and a specific γ which is selected from a column of $\mathbf{F}$ an efficient realization needs to be investigated. A straight-forward architecture would require a standard multiplier and an internal register file to hold all possible γ's. Since the γ's are fixed and their number is finite, we prefer to consider their representation as Canonical Signed-Digits (CSD) to implement the radix-4 booth multiplication in a digit-serial manner based on shift and add steps.

The CSD representation of any binary vector contains no adjacent nonzero digits and has the minimal weight. For a radix-2 SD system the digits are taken from the set $\sum = \{\bar{1}, 0, 1\}$. Further, the CSD is unique and can be transformed easily as shown in [9]. In Tab. 1, the CSD vectors of all γ's associated to CFMB-0 are given using 13 digits. Considering pairs of digits of a CSD vector

$\gamma(1/4) = 1.0\bar{1}0\bar{1}01010000$	$\gamma(1/8) = 1.000\bar{1}0\bar{1}001000$
$\gamma(1/3) = 0.100000000000$	$\gamma(1/10) = 1.000\bar{1}0100\bar{1}000$
$\gamma(1/5) = 1.0\bar{1}01000\bar{1}0010$	$\gamma(1/12) = 1.0000\bar{1}00\bar{1}0100$
$\gamma(1/6) = 1.00\bar{1}000\bar{1}00\bar{1}0\bar{1}$	$\gamma(1/14) = 1.0000\bar{1}010\bar{1}001$
$\gamma(1/7) = 1.00\bar{1}010\bar{1}01010$	$\gamma(1/16) = 1.00000\bar{1}0\bar{1}0001$

Table 1: 13-digit CSD representation $\{d_0.d_{-1}d_{-2}\ldots d_{-B+1}\}$ of γ in the first column of $\mathbf{F}$ assigned to CFMB-0.

only five of nine possible combinations exist

$$d_{-i}d_{-i-1} \in [\bar{1}0,\ 0\bar{1},\ 00,\ 10,\ 01] \tag{9}$$

Looking at pairs of digits allows us to realize the recoded multiplication with two-digit shifting per cycle. The complete shift-add based multiplication is a cascaded structure of basic modules (Fig. 2). The d_0 digit to the left of the fixpoint represents a special case and is realized by the extra AND gate. The control signals for the basic modules are generated based on the digit pairs. Their basic function is to perform the addition/subtraction of $A \cdot 2^{-i}$ or $A \cdot 2^{-(i+1)}$ to/from the temporary result of the previous stage (lower input) as shown in Fig. 3.

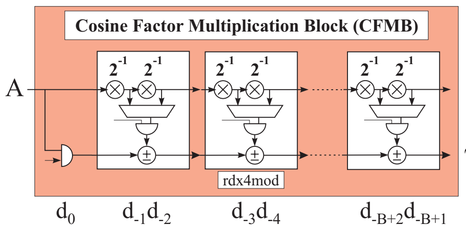

Figure 2: Basic structure of a CFMB.

$d_{-i}d_{-i-1}$	$cs1$	$cs2$	$cs3$
00	–	0	–
01	1	1	*Add*
$0\bar{1}$	1	1	*Sub*
10	0	1	*Add*
$\bar{1}0$	0	1	*Sub*

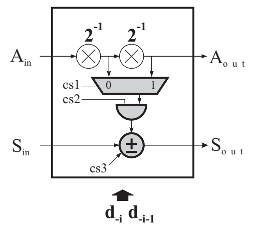

Figure 3: Structure of the basic module "rdx4mod". The control signals are denoted as csi. The shaded instances are not fixed and depend on the set of γ's to be implemented in a CFMB module.

Within each row of $\mathbf{F}$, the assignment of γ's to columns is arbitrary. In terms of hardware resources, it is advantageous for each CFMB to have only γ's which minimize the requirement to implement all five possible combinations for every digit pairs $d_{-i}d_{-i-1}$. Thus, we consider an optimization process which looks for an optimal columnwise reassignment of γ's to reduce the number of adders, subtractors as well as multiplexers in Fig. 3. For the hardware estimation, we assume the use of a Carry-Ripple-Adder (CRA) as well as basic gates in conventional CMOS techniques. For CRA, the realization of a full-adder for addition/subtraction requires about 9 gates/bit. The cost for a two-input multiplexer is 2 gates/bit (two transmission-gates and one buffer) and 3.5 gates/bit for a multiplexer with a following AND-gate. The total cost $\mathcal{C}$ for a CFMB in Fig. 2 is given as

$$\mathcal{C} = L \cdot \left(C_{AND} + \sum_{j}^{\lfloor B/2 \rfloor} C_m(j) + C_{rca}(j) \right) \tag{10}$$

where L is the data bitwidth, C_m, C_{rca} represent the cost of multiplexers and CRA in the j-th "rdx4mod", respectively. C_{AND} is the cost of the AND-gate associated to the digit d_0. The summation is taken over all cascaded pairs of $d_{-i}d_{-i-1}$ which depends on the digit width B of the CSD representation. The procedure to look for the optimal order of $\mathbf{F}$ can be formulated as given in

Algorithm 1 Search for the optimal assignment of γ in $\mathbf{F}$

```
1:  Calculate the CSD representation for all $\gamma$'s of $\mathbf{F}$.
2:  Set $\mathbf{F}_{opt}$ to be the first row of $\mathbf{F}$.
3:  for $r$ = 2 to 10 do
4:      Append the $r$-th row of $\mathbf{F}$ to $\mathbf{F}_{opt}$.
5:      for all permutations of the $r$-th row of $\mathbf{F}_{opt}$ do
6:          Calculate total cost $\mathcal{C}$ of all four CFMBs based on the
            generated $r \times 4$ matrix $\mathbf{F}_{opt}$.
7:      end for
8:      Select all permutations which yield the minimal cost $\mathcal{C}_{min}$.
9:      if more than one permutation with cost $\mathcal{C}_{min}$ then
10:         Choose the permutation which yields minimal standard deviation
            of number of adders/subtractors among CFMBs.
11:     end if
12:     Set new $\mathbf{F}_{opt}$ as the one with the chosen row permutation.
13: end for
```

Algorithm 1. Note that the permutation of γ in the i-th row will lead to a permutation of the corresponding $\mathbf{P}_{k,N}$.

Using the assumptions above for the hardware cost estimation, an optimal assignment of γ's results in

$$\mathbf{F}_{opt} = \begin{bmatrix} \gamma(1/4) & \gamma(1/4) & \gamma(1/4) & \gamma(1/4) \\ \gamma(1/3) & \gamma(1/3) & \gamma(0) & - \\ \gamma(1/5) & \gamma(2/5) & - & - \\ \gamma(1/6) & \gamma(1/6) & \gamma(1/2) & - \\ \gamma(1/7) & \gamma(2/7) & \gamma(3/7) & - \\ \gamma(1/8) & \gamma(3/8) & \gamma(1/8) & \gamma(3/8) \\ \gamma(3/10) & \gamma(1/10) & - & - \\ \gamma(1/12) & \gamma(5/12) & \gamma(1/4) & - \\ \gamma(1/14) & \gamma(5/14) & - & \gamma(3/14) \\ \gamma(1/16) & \gamma(5/16) & \gamma(7/16) & \gamma(3/16) \end{bmatrix} \quad (11)$$

Assuming a bitwidth of $L = 16$ for all data busses in Fig. 2, the total cost $\mathcal{C}$ of CFMB-0/1/2/3 comprises of 4330 gates compared to $\mathcal{C}_{max} = 4 \cdot 16 \cdot (1.5 + 6 \cdot 3.5 + 6 \cdot 9) = 4896$ gates. The saving using this simple hardware cost estimation is 12%. Please note that we have not taken the routing as well as control structure cost into account which would lead to an additional saving. The reason is that we are more interested in the relative rather than the absolute hardware cost of two realizations. If other adder architectures like Carry-Look-Ahead or Carry-Save are used and the routing cost is also considered, one can perform a more accurate cost estimation which may end up in another $\mathbf{F}$ matrix and saving.

4. CIRCUIT COMPLEXITY

In terms of scalability and arbitrary transform length, the time-recursive structure is also suitable to implement the shape-adaptive transform [6, 7]. Different second-order recursive modules can be combined to computed several DCT coefficients in parallel. If no independent data streams are interleaved and the minimal sample period cyle is one multiply-accumulate delay, the recursive architecture requires 8 modules resulting in $8 \times 2 = 16$ multipliers and $8 \times 3 = 24$ adders in total. Another known algorithm for arbitrary number of points in [10] requires 14 multipliers and 32 adders for $N = 8$. Our proposed architecture requires only 1 multiplier for the final scaling and $4 + 22 + 3 = 29$ adders after optimizing the assignment of γ's. The architecture can be easily pipelined to have the minimal sample period equal to one multiplication delay.

5. CONCLUSIONS

The proposed architecture is able to adapt the 1D DCT calculation to variable transform length $2 \leq N \leq 8$. It is modular, regular and employs only one multiplication in each path, thus overcoming the problem of numerical inaccuracy or bitwidth explosion as found in some fast algorithms. The assignment of cosine factors to the module CFMBs is arbitrary and an iterative algorithm has been proposed to minimize the hardware overhead. Representing the cosine factors in CSD, a digit-serial approach has been adopted to realize the multiplierless CFMB's. The architecture represents a trade-off between very regular and modular architecture (with non-optimal operation counts) versus fast, operation-efficient algorithm (but completely irregular and not scalable to variable length). A 2D shape-adaptive DCT can be built using this architecture following the conventional row-column decomposition.

6. REFERENCES

[1] M. Gilge, T. Engelhardt, and R. Mehlan. Coding of Arbitrarily Shaped Image Segments based on a Generalized Orthogonal Transform. *Signal Processing: Image Commun.*, 1:153–180, 1989.

[2] Thomas Sikora. Low Complexity Shape-Adaptive DCT for Coding of Arbitrarily Shaped Image Segments. *Signal Processing: Image Communication*, 7:381–395, 1995.

[3] Peter Kauff, Jan L. P. De Lameillieure, T. Sikora, et al. Functional Coding of Video Using a Shape-Adaptive DCT Algorithm and an Object-Based Motion Prediction Toolbox. *IEEE Trans. Circuits Syst. Video Technology*, 7(1):181–196, February 1997.

[4] Christoph Loeffler et al. Algorithm-Architecture Mapping for Custom DSP Chips. In *IEEE Intern. Symp. on Circuits and Systems*, pages 1953–1956, Espoo, Finnland, June 7–9 1988.

[5] K. J. R. Liu et al. Optimal Unified Architectures for the Real-Time Computation of Time-Recursive Discrete Sinusoidal Transforms. *IEEE Trans. Circuits Syst. Video Technology*, 4(2):168–180, April 1994.

[6] Thuyen Le and Manfred Glesner. VLSI-Architecture of a Time-Recursive 2-D Shape-Adaptive DCT Processor for Generic Coding of Video. In *Proc. of the Intern. Conf. on Signal Processing Applications and Technology (ICSPAT)*, pages 1238–1242, San Diego, California, Sept. 14–17 1997.

[7] Jill Hunter and John V McCanny. Discrete Cosine Transform Generator for VLSI Synthesis. In *Proc. of International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 2997–3000, Seattle, Washington, USA, May 12–15 1998.

[8] Chung-Yen Lu and Kuei-Ann Wen. On the Design of Selective Coefficient DCT Module. *IEEE Trans. Circuits Syst. Video Technology*, 8(2):143–146, April 1998.

[9] Kai Hwang. *Computer Arithmetic: Principles, Architecture, and Design.* John-Wiley & Sons, Inc., 1979.

[10] Meghanad D. Wagh and H. Ganesh. A New Algorithm for the Discrete Cosine Transform of Arbitrary Number of Points. *IEEE Trans. on Computers*, C-29(4):269–277, April 1980.