Design for the Discrete Cosine Transform in VLSI

George S. Taylor and Gerard M. Blair

September 1997

Abstract

The discrete cosine transform is reviewed through recent implementations for the 8x8 transform. The distinct roles of algorithmic and multiplier design are identified, and key circuit and logic innovations are highlighted.

1 Introduction

The discrete cosine transform (DCT) forms a key role in several image compression standards including JPEG [1] for still picture compression, ITU-T H.261 [2] and H263 for teleconferencing, and ISO MPEG-1 and MPEG-2 [3] for audio-visual compression and communication. Because of the existence of these standards, new consumer markets are emerging including, cable access television, direct satellite broadcasting, high definition television (HDTV), digital video compact disc and multimedia personal computers. All of these standards use an 8x8 two-dimensional (2-D) DCT which has therefore become the focus for VLSI implementation.

The 2-D DCT can be described as a transform from a 2-D matrix of pixels to a 2-D matrix of 'spatial frequency information'. The DCT is used for image compression because the transformed matrix contains many small or zero entries. Further the response of the human eye is frequency dependent and this can be exploited by weighting the transform results according to the ability of the human eye to perceive them. For transmission the resulting matrix is quantised (commonly with many values of zero) and encoded using run-length and Huffman encoding to remove sequences of zeros. By changing the amount of quantisation, image quality and bit rate can be traded which thus enables a controllable loss of the perceived image quality.

In this article, we review several recent VLSI implementations reported since they illustrate many different algorithmic and architectural decisions. For many applications such as HDTV, the receiver is the cost critical component and so implementations for the inverse DCT (IDCT) are described in some cases without the corresponding forward DCT. There are three factors which differentiate these designs: the underlying DCT algorithm, the digital architecture for multiplication and specific circuit and logic techniques.

2 The 2-D Discrete Cosine Transform

For an input matrix $x(m,n)$ and an output matrix $z(k,l)$ with $\{0 \leq m,n,k,l < N\}$ the forward NxN 2-D DCT is defined as

$$z(k,l) = \frac{2}{N}\alpha(k)\alpha(l)\sum_{m=0}^{N-1}\sum_{n=0}^{N-1} x(m,n)\cos\frac{(2m+1)\pi k}{2N}\cos\frac{(2n+1)\pi l}{2N} \tag{1}$$

and the inverse NxN 2-D DCT as

$$x(m,n) = \frac{2}{N} \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} \alpha(k)\alpha(l) z(k,l) \cos \frac{(2m+1)\pi k}{2N} \cos \frac{(2n+1)\pi l}{2N} \tag{2}$$

where $\alpha(0) = \sqrt{\frac{1}{2}}$ and $\alpha(k) = 1$ for $k \neq 0$.

A naïve implementation of (1) or (2) requires N^4 multiplications. However, by noting that each cosine part only varies with one of the summations, the transform can be calculated by a *row-column decomposition* with only $2N^3$ multiplications; 2N multiplications per input pixel. The DCT and IDCT become:

$$Z = AXA^T \tag{3}$$

$$X = A^T ZA \tag{4}$$

where X is the source pixel (time domain) data, Z the DCT output coefficients (frequency domain) and A is an orthogonal matrix defined as:

$$a(u,v) = \sqrt{\frac{2}{N}} \alpha(u) \cos \frac{(2v+1)\pi u}{2N} \tag{5}$$

This row-column decomposition is equivalent to a 1-D DCT/IDCT followed by transposition and a second 1-D DCT/IDCT, as shown in Figure 1.

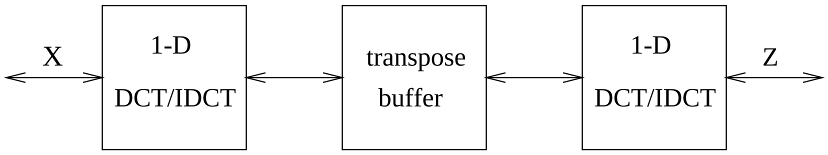

Figure 1: Row-column decomposition

Direct two-dimensional methods are also possible with alternative optimisations.

3 Some DCT algorithms

In this section we consider two 8-point 1-D algorithms for the row-column decomposition and two direct 8x8 2-D approaches.

3.1 Algorithm by Chen et al

Matrix A for the 8-point DCT can be written as:

$$A = \begin{bmatrix} d & d & d & d & d & d & d & d \\ a & c & e & g & -g & -e & -c & -a \\ b & f & -f & -b & -b & -f & f & b \\ c & -g & -a & -e & e & a & g & -c \\ d & -d & -d & d & d & -d & -d & d \\ e & -a & g & c & -c & -g & a & -e \\ f & -b & b & -f & -f & b & -b & f \\ g & -e & c & -a & a & -c & e & -g \end{bmatrix} \tag{6}$$

The multiplier coefficients are defined in appendix A.

The symmetry in this matrix can be exploited and the 1-D DCT rearranged to give:

$$\begin{aligned}
\begin{bmatrix} y_0 \\ y_2 \\ y_4 \\ y_6 \end{bmatrix} &= \begin{bmatrix} d & d & d & d \\ b & f & -f & -b \\ d & -d & -d & d \\ f & -b & b & -f \end{bmatrix} \begin{bmatrix} x_0 + x_7 \\ x_1 + x_6 \\ x_2 + x_5 \\ x_3 + x_4 \end{bmatrix} \\
\begin{bmatrix} y_1 \\ y_3 \\ y_5 \\ y_7 \end{bmatrix} &= \begin{bmatrix} a & c & e & g \\ c & -g & -a & -e \\ e & -a & g & c \\ g & -e & c & -a \end{bmatrix} \begin{bmatrix} x_0 - x_7 \\ x_1 - x_6 \\ x_2 - x_5 \\ x_3 - x_4 \end{bmatrix}
\end{aligned} \tag{7}$$

The $(N \times N)$ multiplication matrix has been replaced by two $(N/2) \times (N/2)$ matrices, which can be computed in parallel, as can the sums and differences forming the vectors on the right-hand side of (7).

The implementations by Madisetti and Willson [4], Uramoto *et al* [5], Matsui *et al* [6] and Jang *et al* [7] are based upon this decomposition which requires 32 multiplications. However, Madisetti and Willson observe that the first part of (7) only involves multiplication by three rather than four different constants and so reduce the number of multiplications to 28.

The frequently referenced algorithm by Chen *et al* [8] is also derived from (7) but only requires 16 multiplications with 2 multiplications on the critical path. The data-flow graph for Chen's algorithm is shown in Figure 2. To compute the IDCT, the role of the inputs and outputs are reversed.

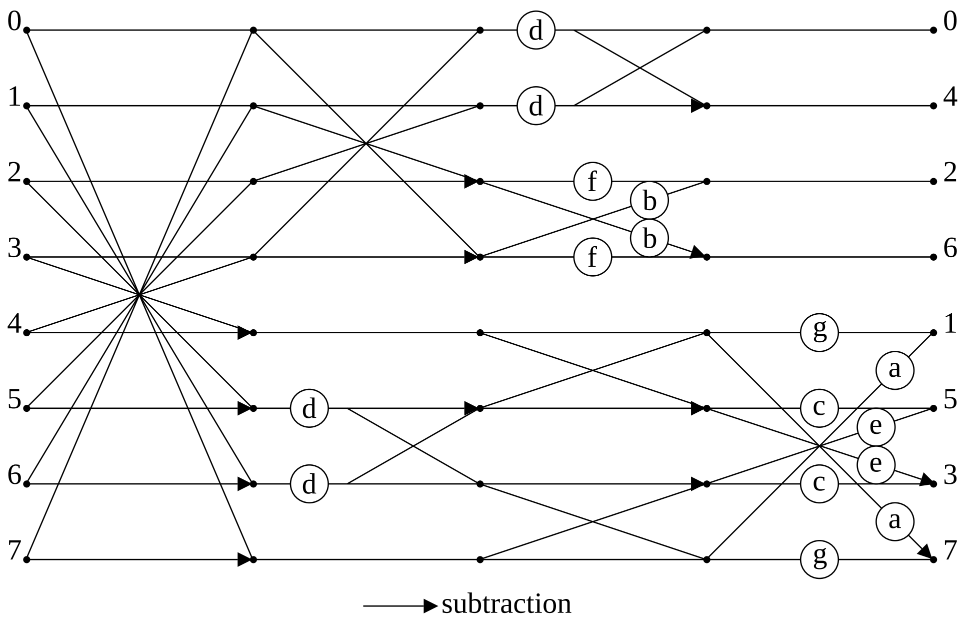

Figure 2: The algorithm by Chen *et al* [8]

If Chen's algorithm is directly implemented in 4 clock cycles, one for each stage in the data-flow graph, each cycle requires at most 8 additions and at most 8 multiplications. The implementation used by Madisetti and Willson also takes 4 cycles and requires 8 additions per cycle, but only 7 multiplications per cycle are needed (although it is 7 in every cycle); additionally, 1 adder and 1 subtractor are needed to prepare the right-hand side sums and differences.

3.2 Algorithm by Arai et el

Although the minimum number of multiplications for the 8-point 1-D DCT algorithm is 11 [9], the algorithm by Arai *et al* [10] requires only 5. This is possible because the outputs are scaled: to obtain the true DCT value each requires a further multiplication. However, in many systems this can be incorporated into multiplication coefficients used in the subsequent stage, for instance the perceptual weights in the video encoding algorithm. This type of algorithm is known as a scaled-DCT. If the two dimensional transform is produced by row-column decomposition, then the scaling for both layers of one dimensional transforms can be combined and performed after the second layer.

As can be seen from Figure 3, the Arai algorithm also has the useful properties that the 5 multiplications can be performed in parallel and that no path through the data-flow graph includes more than one multiplication. Details of the inverse transform may be found in [10].

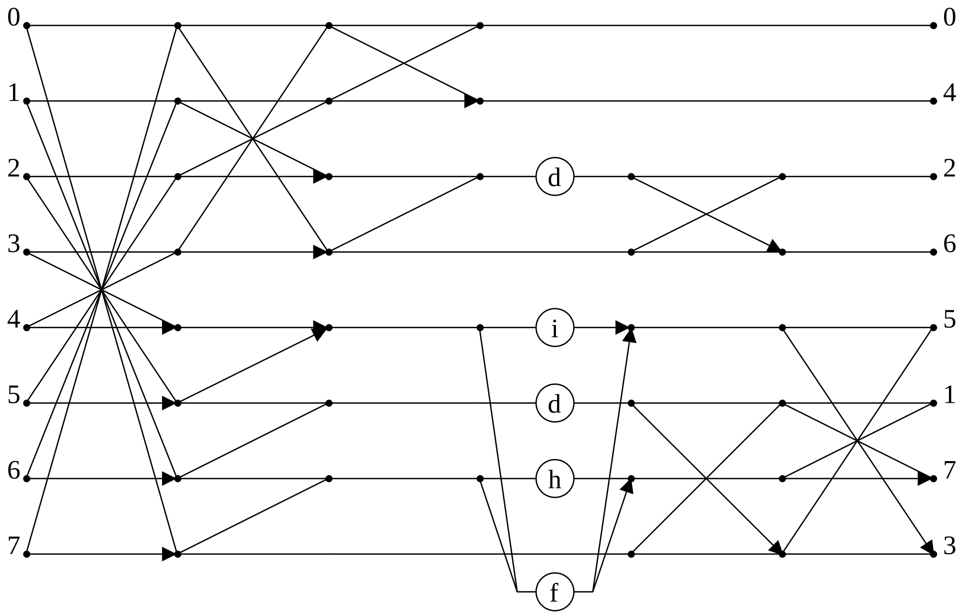

Figure 3: Forward algorithm by Arai *et al* [10]

The Arai algorithm provides a good basis for a low cost scaled-DCT of other orders since it views the scaled N-point DCT as the real part of a 2N-point discrete Fourier transform, which can be effectively implemented using fast DFT algorithms based upon cyclic-convolutions and Winograd's algorithm. Such solutions will display the same basic structure as the 8-point data-flow graph.

3.3 Algorithm by Chang and Wang

Chang and Wang [11] describe an algorithm which performs a direct 2-D transform using row-column decomposition within a systolic array. The NxN 2-D DCT in (3) is split by applying row-column decomposition into two steps of calculation, with intermediate result Y. Denoting $c_{ij} = \cos\frac{(2j+1)i\pi}{2N}$ from (5), and neglecting the scale factor, the matrix multiplications can be written as:

$$Y_{kn} = \sum_{m=0}^{N-1} X_{mn} c_{km} \tag{8}$$

$$Z_{kl} = \sum_{n=0}^{N-1} Y_{kn} c_{ln} \tag{9}$$

$$\tag{10}$$

By using the symmetries of the cosine function, and assuming that N is even, they define

$$U_{mn} = X_{mn} + (-1)^k X_{(N-1-m)n} \tag{11}$$

and

$$V_{kn} = Y_{kn} + (-1)^l Y_{k(N-1-n)}. \tag{12}$$

to give:

$$Y_{kn} = \sum_{m=0}^{N/2-1} U_{mn} c_{km}. \tag{13}$$

$$Z_{kl} = \sum_{n=0}^{N/2-1} V_{kn} c_{ln} \tag{14}$$

The algorithm is implemented as a systolic array in four stages corresponding to (11), (13), (12), (14). This requires a total of 64 multipliers for an 8x8-pt DCT.

3.4 Algorithm by Liu and Chiu

A different approach, taken by Liu and Chiu [12–14], is to calculate a running (or recursive) DCT in which the values of the DCT are updated with each new sample. Given a sequence of input data a 1-D DCT of the last N input values is output. Each DCT utilises the previous DCT result, the next DCT is obtained by adding the difference between it and the previous DCT. The discrete sine transform (DST) is needed in the calculation of the DCT and hence both DST and DCT outputs are available.

The 2-D version is achieved using a circular shift register between two 1-D DCTs and N-length delays in each of the second layer filters. These together replace the usual transposition RAM and networks. An implementation is provided by Srinivasan and Liu [15, 16]. Aburdene *et al* [17] describe a similar time recursive approach for a 1-D DCT based on Clenshaw's recurrence formula [18].

3.5 Algorithm by Hsia et al

In [19] Hsia *et al* present an algorithm to calculate the 2-D IDCT directly by skipping non-zero coefficients. The algorithm may be understood by considering the symmetries of the matrix formed by the product of the two cosines in (2). For each value of (k, l), the matrix can be divided into 4 quadrants where the values of each are either the same as, the negative of, or a reordering of values of the first quadrant. Furthermore, the values of the first 4x4 quadrant can be calculated from the sum and difference of just four multiplications. Thus for each non-zero input (DCT coefficient) to the IDCT, there are four multiplications, various additions and negations.

3.6 Possible developments

For the un-scaled DCT, the family of 11 multiplier algorithms given by Loeffler *et al* [20] seems attractive as does an alternative in the same paper which requires 12 multipliers (only one more than the minimum of 11) and 32 adders. As with the Arai algorithm, this has the advantage that no path includes more than one multiplier and that all the multiplications can be performed in parallel. McGovern *et al* [21] propose a similar algorithm.

A direct 2-D algorithm which has not yet been implemented in hardware, is a 2-D development of the Arai *et al* [10] scaled-DCT algorithm proposed by Feig and Winograd [22]. In this, the multiplications from the two stages of the row-column decomposition are combined in much the same manner as for the Winograd DFT algorithm design. The result is a complete 8x8 2-D DCT with only 54 multiplications (with some shift operations for 1/2) and only 7 distinct coefficients (combined with a further shift operation for an eighth). The latter property leads to either low multiplier coefficient storage requirements or a practical number of hardwired multipliers. Furthermore, the networks for additions are formed by the repeated application of only two sub-networks. The main disadvantage is that the algorithm requires two data transpositions.

4 Multiplier architectures

There are three main multiplier architectures:

Combinational

Combinational multipliers take an n bit word and multiply by an m bit word to give an $n+m$ bit word. In many DCT implementations, purely combinational logic is used without pipelining. Such multipliers are typically implemented as a tree structure of adders and consume a large chip area, but are very fast. A variation is the hard-coded combinational multiplier which multiplies by a constant coefficient and therefore can be optimised for both area and speed.

Distributed arithmetic

Distributed arithmetic replaces combinational multipliers with a lookup table and accumulator. The computation of the vector product multiply-accumulation is broken down as follows:

Given a multiplicand vector x_k for $k = 0, ..., K$ and the constant multiplier coefficient vector a_k for $k = 0, ..., K$ the vector product is

$$y = \sum_{k=1}^{K} a_k x_k. \tag{15}$$

If x_k is in N bit twos complement fractional form, i.e.

$$x_k = -b_{k0} + \sum_{n=1}^{N-1} b_{kn} 2^{-n} \tag{16}$$

where b_{kn} represents the value, 0 or 1, of bit n of x_k then (15) can be computed as

$$y = \sum_{k=1}^{K} a_k(-b_{k0}) + \sum_{n=1}^{N-1} \left[\sum_{k=1}^{K} a_k b_{kn} \right] 2^{-n}. \tag{17}$$

The partial products $[\sum_{k=1}^{K} a_k b_{kn}]$ are stored in a ROM and accumulated for each bit of the multiplicand. Distributed arithmetic leads to a smaller area than a combinatorial multiplier but only allow multiplication by a fixed set of coefficients. These are, however, easier to program (by changing the ROM contents) than a corresponding set of hardwired combinatorial multipliers.

Serial

The bit-serial architectures balance the long processing time of each operation (only one-bit per clock cycle) by performing them on several words in parallel. Thus the input is word-parallel (a row/column of 8 words) and bit-serial. A typical bit-serial system takes $n + 1$ cycles to add two n bit values and $n + m$ cycles to multiply an n bit word by an m bit coefficient. As for the other multiplier designs a hard coded version can be made to multiply by a constant coefficient.

Bit-serial systems can be made to operate at very high clock frequencies due to the small propagation delays involved in the small blocks of combinational logic, however, this is not evident from the reported DCT designs to date. Bit-serial DCT implementations are simply a direct mapping of a data-flow graph.

Bit-serial multiplication progresses least significant bit first. It has a high latency since the more significant half of the product is only generated after the less significant half, which is generally discarded. On-line arithmetic avoids this delay by progressing most significant digit first and by using redundant number encoding. Bruguera and Lang [23] describe a DCT implementation using this technique, although no performance figures are given.

5 Implementations

The following recent implementations were chosen to illustrate the use of specific logic and circuit techniques to enhance performance.

5.1 1-D based designs

Madisetti and Willson [4] implement the Chen based algorithm discussed in section 3.1. The 1-D DCT unit consists of 7 combinational multipliers, one for each of the matrix elements $a, ..., g$ as shown in Figure 4, and 8 combinational accumulators to sum the outputs. To minimise the critical path within the DCT unit, hardwired multipliers are used. These multiply by a constant coefficient which is fixed at design time leading to a faster and smaller module at the expense of programmability. Thus one multiplier is needed for each each unique multiplier coefficient. Each accumulator contains a multiplexer to select a particular multiplier output and logic to perform either addition or subtraction according to the sign of the coefficient.

A 'data reorder unit' (not shown in Figure 4 prepares the sums and differences for the DCT, and reorders the data into even and odd inputs for the IDCT. The whole 1-D DCT/IDCT takes 4 clock cycles. Rather than use a second 1-D DCT unit, a single 1-D DCT unit is multiplexed between the input data and the output of the transpose buffer implemented by a 64 word DRAM (shown in Figure 5). However, since the single 8-pt transform requires only 4 clock cycles and each data point is processed in two transforms, the clock rate is actually equal to the data rate.

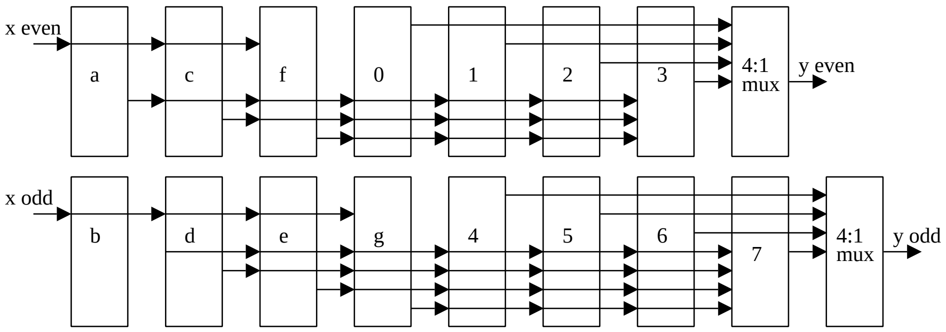

Figure 4: Multiply accumulate by Madisetti and Willson [4]

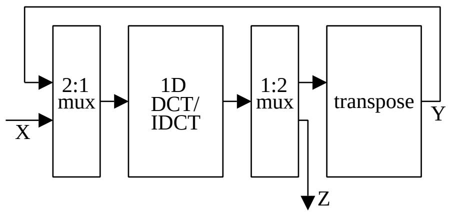

Figure 5: Multiplexed 1-D DCT by Madisetti and Willson [4]

The multipliers use radix-2 signed digit representation in which an n bit fractional number can be expressed as

$$f = \sum_{k=0}^{n-1} s_k 2^{-k} \qquad s_k \in \{-1, 0, 1\}. \tag{18}$$

This form of representation allows most numbers to be represented with fewer non-zero digits than in twos-complement representation. In the hardwired implementation one adder is required per non-zero digit, thus giving a smaller implementation. Between 4 and 6 adders are required for the constants $a, ..., g$ to give 12-bit accuracy. The products are accumulated in 22-bit wide carry-select adders.

In total 7 multipliers and 8 accumulators are needed. Using 0.8μm CMOS the $N = 8$ implementation by Madisetti and Willson has an area of 10mm^2. A complete 2-D transform is computed every N^2 cycles and an input sample rate of 100MHz is possible.

Kovac and Ranganathan [24] describe a JPEG encoder based on the algorithm by Arai *et al* (section 3.2) using row-column decomposition with two separate 1-D units. Although not designed for real time MPEG a 100MHz input pixel rate is possible.

Each 1-D unit, shown in Figure 6, consists of 6 stages corresponding to the 6 columns of operations shown in Figure 3. Each stage contains a pipeline register set (RS) and a combinational adder capable of negating the inputs, except for stage 4 which includes a single Wallace-Tree combinational multiplier. Each register set consists of two columns of 14-bit registers which act as data buffers. Data from the previous stage is written into the left column and, once complete is moved into the right column where it is loaded into the next stage.

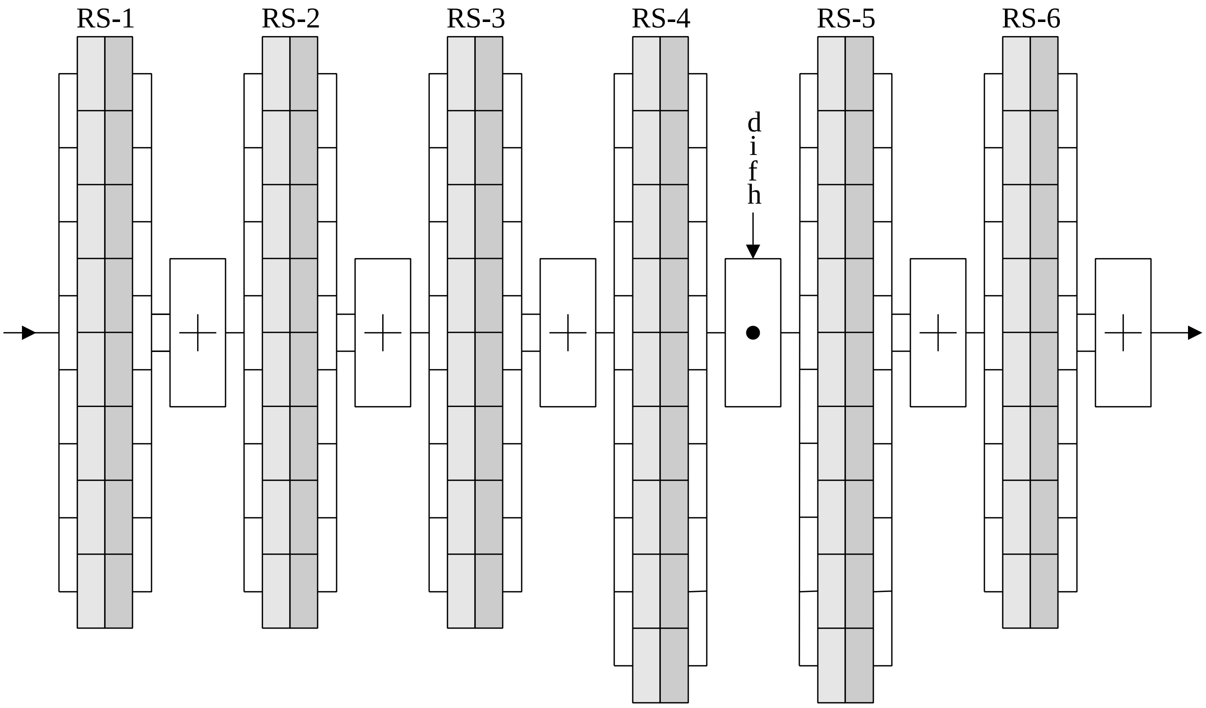

Figure 6: 1-D DCT by Kovac and Ranganathan [24]

Uramoto *et al* [5] implement the Chen based algorithm using distributed arithmetic with two separate 1-D units. To halve the computation time the input values are processed in pairs so that two partial products are accumulated at a time; this requires a dual port ROM. Throughput is further enhanced by pipelining.

By changing the ROM contents the same circuit can be used to calculate the DCT or the IDCT. Alternatively a ROM holding twice as many words can be employed and the appropriate half selected. Uramoto *et al* have developed a 'dual plane ROM' which holds two banks of values where only one bank can be accessed at a time. The dual plane ROM gives a 30% area saving and a speed increase: combined with carefully designed accumulators and routing a 100MHz input pixel rate is possible.

Karathanasis [25] also proposes an algorithm for use with distributed arithmetic, simulated to be capable of a 100MHz pixel rate, but with only 10% of the ROM requirement of Uramoto *et al* [5].

Matsui *et el* [6] give another implementation using the Chen based algorithm capable of a 200MHz input pixel rate. Each of the two 1-D DCT units contain 8 distributed arithmetic multiply-accumulators. The key feature is the high-speed logic design based upon differential NMOS pass transistor logic [26]. Here a logic signal is transmitted as two complementary signals with sense-amplifiers to detect the differential voltage (as small as 100mV) between the signals thus reducing the effect of parasitic capacitance.

Masaki *et al* [27] describe another distributed arithmetic implementation of the IDCT using the Chen algorithm with specific focus on producing a single chip MPEG decoder. A single 1-D IDCT unit with pipelined Wallace-Tree adders is multiplexed with the input coefficients and transpose buffer. A more sophisticated distributed arithmetic scheme is used resulting in a 200MHz pixel rate with only a 100MHz clock.

For comparison we mention two bit-serial implementations. Cucchi and Fratti [28] implement the Chen algorithm with bit-serial arithmetic. Two bits are processed at a time so that the internal clock rate is equal to the input pixel rate (40MHz). McGovern *et al* [21] also use bit-serial arithmetic for an algorithm similar to Loeffler *et al* [20] requiring the minimum number of adders and multipliers. Both of these implementations use 2 1-D DCT units and a transpose buffer.

5.2 Direct 2-D

Hsia *et al* [19] implement the algorithm in section 3.5 using pipelined combinational multipliers to decrease the critical path. A K-bit by K-bit multiplier is composed from a tree structure of four K/2-bit by K/2-bit multipliers, registers are inserted into the tree such that the critical path involves only a

K-bit adder instead of a K-bit multiplier. A bank of 64 accumulators holds the current total for each of the IDCT output pixels. The implementation achieves an average pixel rate varying from 150MHz to a maximum of 400MHz as the compression ratio varies from 4 to 16. This corresponds to an average of 33% to 9% non-zero input coefficients.

Another direct 2-D architecture exploiting zero valued DCT coefficients to reduce power consumption is proposed by Xanthopoulos *et al* [29]. Further power reduction is achieved by connecting latches to the inputs of an adder to stop unwanted transitions passing through the adder. Finally the supply voltage and clock frequency are lowered when there are fewer non-zero coefficients to be processed.

A systolic array is a structure of identical cells with many local and few global interconnections. Such arrays are ideally suited to VLSI implementation due to good regularity, modularity and concurrency. The algorithm by Chang and Wang [11] is implemented using systolic structures. The structure to implement the first two stages, see section 3.3, is given here in Figure 7 for $N = 4$. In a similar way structures for the second and third stages can be created (as detailed in [11]). The IDCT can be implemented by simply inverting the order of the structures.

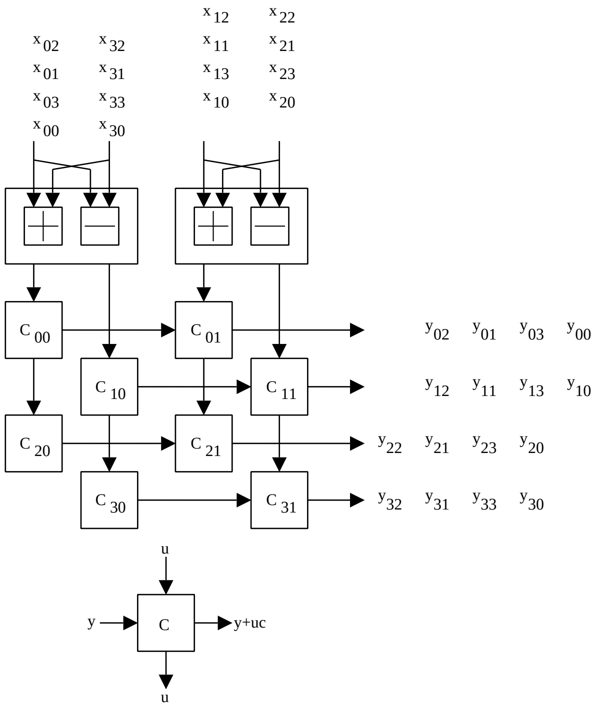

Figure 7: First two stages of array for $N = 4$ by Chang and Wang [11]

The adders (16 bit) and multipliers (12×12 bit) are combinational; a total of N^2 multipliers and N^2+3N adders are required; a complete 2-D transform is computed every N cycles. The implementation for $N = 8$ reads the data in word-parallel bit-parallel format (8 words per clock cycle) and can support a pixel rate of over 320MHz – well in excess of current HDTV requirements.

Srinivasan and Liu [15, 16] implement the 2-D lattice algorithm by Liu and Chiu [12–14] (section 3.4) using distributed arithmetic multipliers. The partial product lookup ROM is addressed by a 12 bit word. This would give 4096 rows, to reduce area the ROM is split into two 6 bit addressed 64 row ROMS. One ROM output is then shifted, sign extended and added to the other. Although only 32 multipliers are needed, the algorithm requires a large number of bit-parallel registers.

6 Comparison and conclusions

Figure 8 shows the area each implementation requires, scaled to a constant 0.8μm process technology. The multiplier styles are combinational multiplier (CM), distributed arithmetic (DA) and bit-serial (BS). Also shown is the number of multipliers for the 8x8 transform. Figure 9 shows the pixel rate achievable of each implementation, along with the feature size in microns and whether the implementation is electrically switchable (E) between IDCT and DCT, IDCT only (I), DCT only (D), or both but not electrically switchable (D/I). The pixel rates are un-scaled due to the difficulties in scaling operation frequency with feature size (which may vary from constant to quadratic depending on which scaling model is used). For the implementation by Hsia *et al* [19] a pixel rate of 400MHz is assumed, this corresponds to the average number of non-zero coefficients in a typical MPEG sequence reported by both [19] and [29].

The direct 2-D implementations require a larger area, but deliver a considerably higher pixel rate, except the implementation by Srinivasan and Liu [15, 16]. The relatively poor performance of Srinivasan and Liu's implementation may be due to the fact that the underlying algorithm was originally developed for overlapping blocks (a running DCT) for which a smaller implementation is attained.

The bit-serial designs (Cucchi and Fratti [28], McGovern *et al* [21]) perform badly, neither of them being fast enough to meet HDTV requirements, and the implementation by Cucchi and Fratti requires over twice the area of the other 1-D with transpose implementations.

The 1-D designs use either combinational multipliers or distributed arithmetic and all employ one of the algorithms from section 3.1. The design by Madisetti and Willson [4] achieves a small area by multiplexing a single 1-D DCT, however the design by Matsui *et al* [6] achieves twice the pixel rate (200MHz) for a 30% increase in area, due to use of a novel sense-amplifier and pass-transistor logic. The Masaki *et al* [27] implementation although also capable of 200MHz is larger and performs the IDCT only.

Based upon this study we conclude the following guidelines for designers. Higher performance 1-D implementations require multipliers which are either hardwired-combinatorial logic or sophisticated ROM architectures and/or circuit techniques for distributed arithmetic. However, high-speed circuit techniques for the multiplier/accumulator will be significant to performance. If even higher throughput is required, then this may be traded-off against area with a full 2-D approach (which often, however, avoids the area cost of the transpose memory). In data compression applications algorithms which exploit zero coefficients are to be preferred.

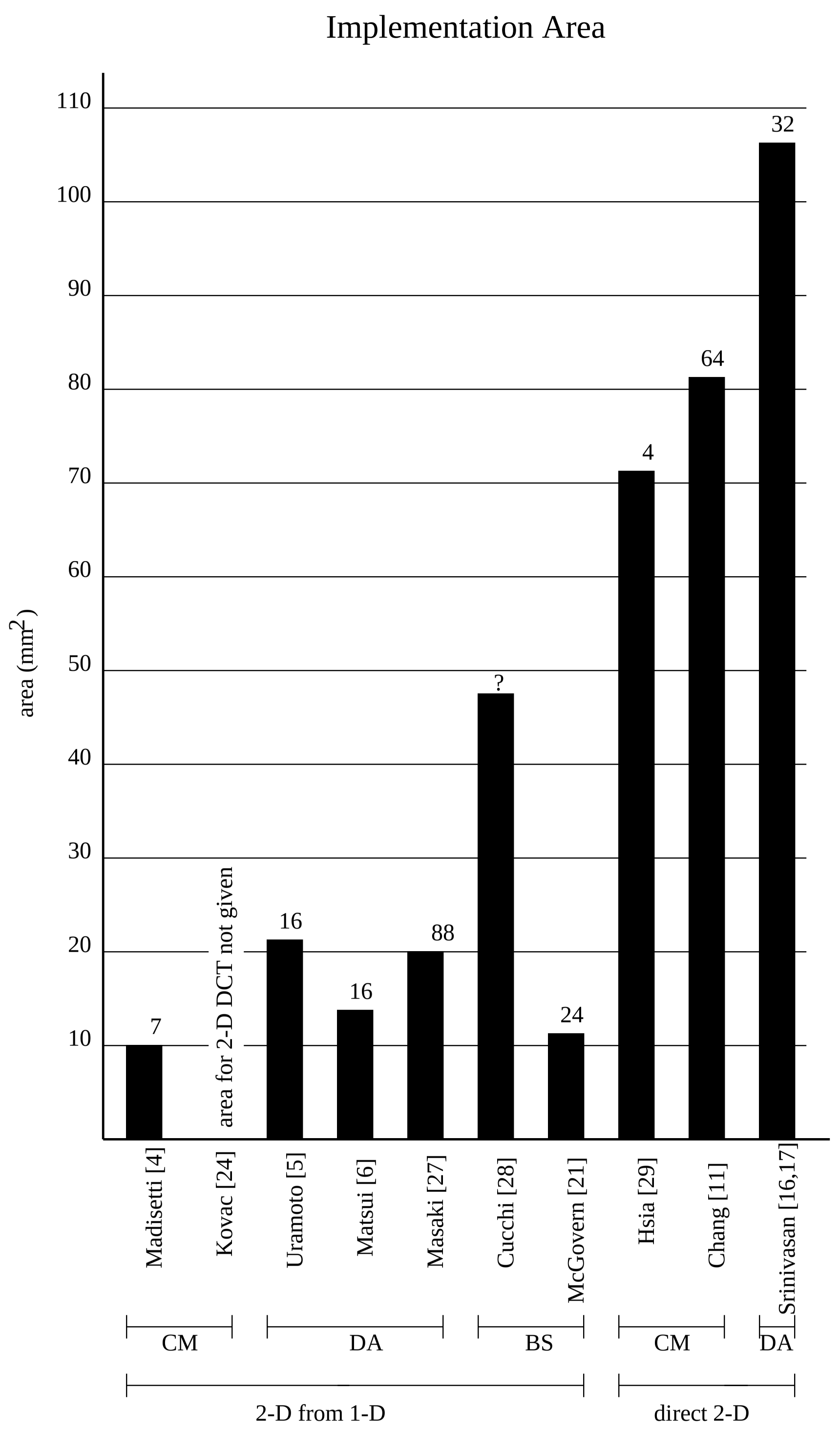

Figure 8: Implementation area scaled to 0.8μ process

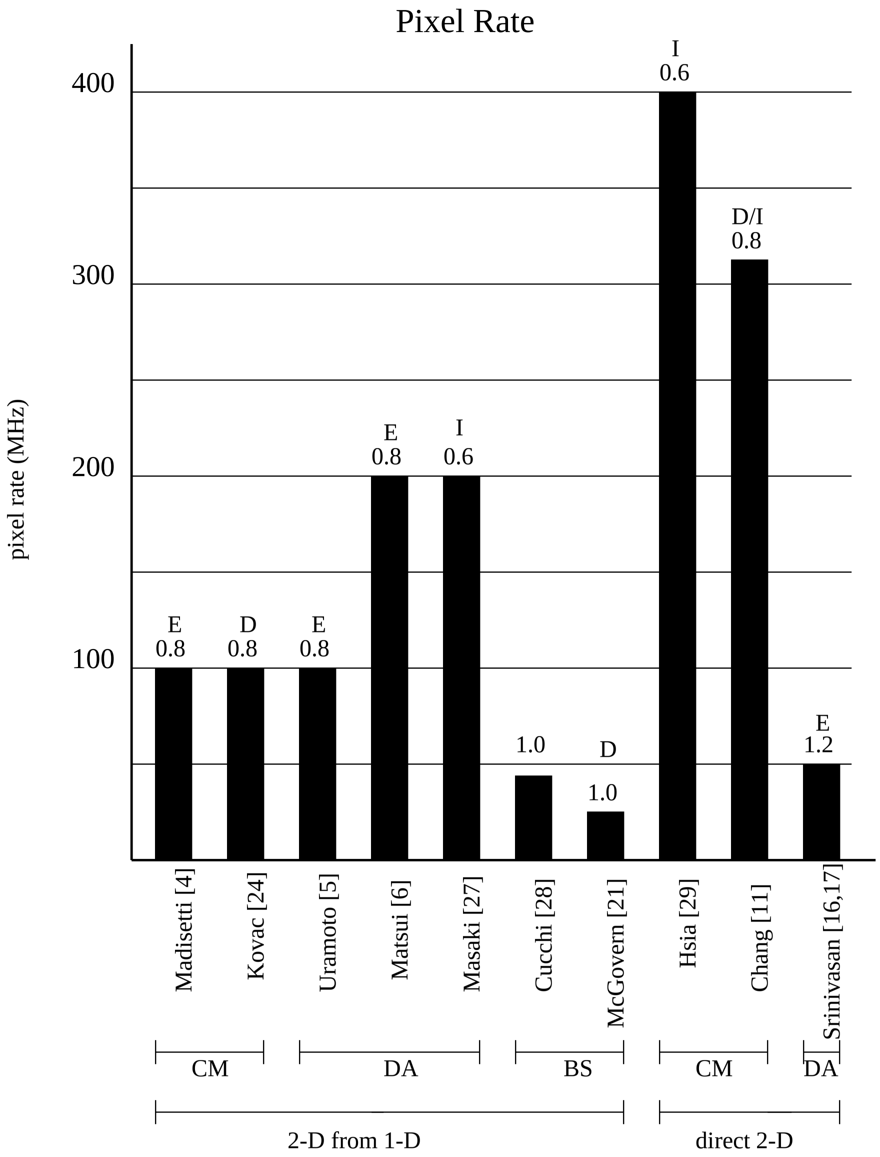

Figure 9: Pixel rate (no scaling)

References

[1] G. K. Wallace, "The JPEG still picture compression standard," in *Communications of the ACM*, vol. 34, pp. 31–44, April 1991.

[2] "CCITT Recommendation H.261," 1990.

[3] D. L. Gall, "MPEG: a video compression standard for multimedia applications," in *Communications of the ACM*, vol. 34, pp. 46–58, April 1991.

[4] A. Madisetti and A. N. Willson, "A 100MHz 2-D 8x8 DCT/IDCT processor for HDTV applications," in *IEE Transactions on Circuits and Systems for Video Technology*, vol. 5, pp. 158–165, April 1995.

[5] S.Uramoto, Y.Inoue, A.Takabatake, J.Takeda, Y.Yamashita, H.Terane and M.Yoshimoto, "A 100-MHz 2-D discrete cosine transform core processor," in *IEEE Journal of Solid State Circuits*, vol. 27, pp. 492–499, April 1992.

[6] M. Matsui, Y. Uetani, L. Kim, T. Nagamatsu, Y. Watanabe, A. Chiba, K. Matsuda, and T. Sakuri, "A 200MHz $13mm^2$ 2-D DCT macrocell using sense-amplifying pipeline flip-flop scheme," in *IEEE Journal of Solid State Circuits*, vol. 29, pp. 1482–1490, December 1994.

[7] Y. Jang, J. Kao, J. Yang, and P. Huang, "A 0.8μ 100-MHz 2-D DCT core processor," in *IEEE Transactions on Consumer Electronics*, vol. 40, pp. 703–709, August 1994.

[8] W. Chen, C. H. Smith, and S. Fralick, "A fast computation algorithm for the discrete cosine transform," in *IEEE Transactions on Communications*, vol. 25, pp. 1004–1009, September 1977.

[9] P. Duhamel and H. H'Mida, "New 2^n DCT algorithms suitable for VLSI implementation," in *Proceedings IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 1805–1808, April 1987.

[10] Y. Arai, T. Agui, and M. Nakajima, "A fast DCT-SQ scheme for images," in *The Transactions of the IEICE*, vol. E71, pp. 1095–1097, November 1988.

[11] Y. T. Chang and C. L. Wang, "New systolic array implementation of the 2-D discrete cosine transform and its inverse," in *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 5, pp. 150–157, April 1995.

[12] K. Ray-Liu and C. T. Chiu, "Unified parallel lattice structures for time-recursive discrete cosine/sine/hartley transforms," in *IEEE Transactions on Signal Processing*, vol. 41, pp. 1357–1377, March 1993.

[13] C. T. Chiu and K. Ray-Liu, "Real-time parallel and fully pipelined two-dimensional DCT lattice structures with application to HDTV systems," in *IEEE Transactions on Circuits and Systems for Video Techonology*, vol. 2, pp. 25–37, March 1992.

[14] C. T. Chiu and K. Ray-Liu, "Real-time recursive two-dimensional DCT for HDTV systems," in *ICASSP-92: IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 3, pp. 205–208, March 1992.

[15] V. Srinivasan and K. Ray-Liu, "VLSI design of high-speed time-recursive 2-D DCT/IDCT processor for video applications," in *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 6, pp. 87–96, February 1996.

[16] V. Srinivasan and K. Ray-Liu, "Full custom VLSI implementation of high-speed 2-D DCT/IDCT chip," in *IEEE Proceedings ICIP-94*, vol. 3, pp. 606–610, November 1994.

[17] M. F. Aburdene, J. Zheng, and R. J. Kozick, "Computation of discrete cosine transform using Clenshaw's recurrence formula," in *IEEE Signal Processing Letters*, vol. 2, pp. 155–156, August 1995.

[18] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery, *Numerical Recipes in C: The Art of Scientific Computing*. UK: Cambridge University Press, 2nd ed., 1992.

[19] S. C. Hsia, B. D. Liu, J. F. Yang, and B. L. Bai, "VLSI implementation of parallel coefficient-by-coefficient two-dimensional IDCT processor," in *IEEE Transactions On Circuits and Systems for Video Technology*, vol. 5, pp. 396–406, October 1995.

[20] C. Loeffler, A. Ligtenberg, and G. S. Moschytz, "Practical fast 1-D DCT algorithms with 11 multiplications," in *Proceedings of ICASSP*, pp. 988–991, 1989.

[21] F. A. McGovern, R. F. Woods, and M. Yan, "Novel VLSI implementation of (8x8) point 2-D DCT," in *Electronics Letters*, vol. 30, pp. 624–626, April 1994.

[22] E. Feig and S. Winograd, "Fast algorithms for the discrete cosine transform," in *IEEE Transactions on Signal Processing*, vol. 40, pp. 2174–2193, September 1992.

[23] J. Bruguera and T. Lang, "2-D DCT using on-line arithmetic," in *IEEE Internation Conference on Acoustics, Speech and Signal Processing*, pp. 3275–3278, 1995.

[24] M. Kovac and N. Ranganathan, "JAGUAR: a fully pipelined VLSI architecture for JPEG image compression standard," in *Proceedings of the IEEE*, vol. 83, pp. 247–257, February 1995.

[25] H. C. Karathanasis, "A low ROM distributed arithmetic implementation of the forward/inverse DCT/DST using rotations," in *IEEE Transactions on Consumer Electronics*, vol. 41, pp. 263–272, May 1995.

[26] J. Pasternak and A. Shubat, "CMOS differential pass-transistor logic design," in *IEEE Journal of Solid-State Circuits*, vol. 22, pp. 216–222, April 1987.

[27] T. Masaki, Y. Morimoto, T. Onoye, and I. Shirakawa, "VLSI implementation of inverse discrete cosine transformer and motion compensator for MPEG2 HDTV video decoding," in *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 5, pp. 387–395, October 1995.

[28] S. Cucchi and M. Fratti, "A novel architecture for VLSI implementation of the 2-D DCT/IDCT," in *IEEE*, vol. V, pp. 693–696, 1992.

[29] T. Xanthopoulos, A.P.Chandrakasan, C.G.Sodini, and W.J.Dally, "A data-driven IDCT architecture for low power video applications," in *ESSCIRC'96*, September 1996.

About the authors

George S. Taylor (G.S.Taylor@ee.ed.ac.uk) is researching VLSI design, and Gerard M. Blair is a senior lecturer, at the Department of Electrical Engineering, University of Edinburgh, The King's Buildings, Edinburgh, EH9 3JL, Scotland, UK.

A Multiplier coefficients

Table 1 gives the multiplier coefficients referred to in this paper. Define $c_n = \cos n\pi/16$.

a	c_1	o	$\sqrt{2}(c_2 - c_6)$
b	c_2	p	$\sqrt{2}c_6$
c	c_3	q	$\sqrt{2}(c_2 + c_6)$
d	c_4	r	$-c_1 + c_3 + c_5 - c_7$
e	c_5	s	$c_1 + c_3 - c_5 + c_7$
f	c_6	t	$c_1 + c_3 + c_5 - c_7$
g	c_7	u	$c_1 + c_3 - c_5 - c_7$
h	$c_2 + c_6$	v	$c_3 - c_7$
i	$c_2 - c_6$	w	$c_1 + c_3$
j	$1/2c_2$	x	$c_3 + c_5$
k	$1/2c_4$	y	$c_3 - c_5$
m	$1/2c_6$		

Table 1: Multiplier coefficients