A 100 MHz 2-D 8 × 8 DCT/IDCT Processor for HDTV Applications

Avanindra Madisetti and Alan N. Willson, Jr., *Fellow, IEEE*

Abstract— **This paper discusses the design of a combined DCT/IDCT CMOS integrated circuit for real time processing of HDTV signals. The processor operates on 8 × 8 blocks. Inputs include the blocked pixels that are scanned one pixel at a time, and external control signals that control the forward or inverse modes of operation. Input pixels have a precision of 9-b for the DCT and 12-b for the IDCT. The layout has been generated with a 0.8 μm CMOS library using the Mentor Graphics GDT tools and measures under 10 mm^2. Critical path simulation indicates a maximum input sample rate of 100 MHz.**

I. Introduction

TRANSFORM coders, in particular the Discrete Cosine Transform (DCT), have been widely used in the implementation of low-rate codecs for video compression. The DCT has become an integral part of several standards such as JPEG [11], MPEG [12], and CCITT Recommendation H. 261 [10]. For real time implementation of the DCT at MPEG-1 rates (approximately a 3.5 MHz sample rate), general purpose DSP's [1]–[3] that adopt highly parallel and/or pipelined architectures have been proposed. However, with the imminent arrival of High Definition Television (HDTV) and the considerably higher sample rates (75 MHz sampling frequencies), processors specific to the DCT (ASIC's) have been recently reported. The use of such processors can be justified only if they result in a cost effective implementation. It is the aim of this paper to develop a simple but cost effective DCT/IDCT processor.

The paper is organized as follows: The 2-D DCT and relevant properties are briefly discussed in Section II. The architecture exploiting these properties is described in Section III. The architecture is entirely multiplierless and highly parallel. Clocking at 100 MHz, the processor is capable of handling real-time HDTV signals.

We have tried to minimize the core area while keeping the I/O requirements simple. In order to reduce hardware complexity, one must compromise on the accuracy and internal wordlength. Finite wordlength and accuracy studies have been carried out to ensure that the resulting implementation conforms to existing standards (H.261) [10]. These are described in Section IV. The core characteristics are described in Section V.

Manuscript received February 25, 1994; revised February 7, 1995. This work was supported by the Office of Naval Research under Grant N00014-95-1-0231. This paper was recommended by Associate Editor W. Li.

The authors are with the Integrated Circuits and Systems Laboratory, Department of Electrical Engineering, University of California, Los Angeles, CA 90024 USA.

IEEE Log Number 9410825.

II. The 2-D DCT and IDCT

A. Description of Algorithms

Given an input block $x(m,n)$ with $\{0 \le m,n < N\}$ the forward and inverse 2-D DCT can be written as

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

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

where k, l, m, and n range from 0 to $N-1$ and where $\alpha(0) = 1/\sqrt{2}$ and $\alpha(j) = 1$ for $j \neq 0$. The forward and inverse transforms are merely mappings from the spatial domain to the transform domain and vice versa. A straightforward implementation of (1) or (2) requires N^4 multiplications for the evaluation of the DCT or the IDCT, respectively. Fortunately, the DCT is a separable transform and it can be expressed in matrix notation (the row-column decomposition) as two 1-D DCT's as follows:

$$Z = AXA^T$$
$$X = A^TZA$$

where $AA^T = I_N$ by virtue of the orthogonality of A, which is an $N \times N$ matrix whose basis vectors are sampled cosines, defined as

$$a(k,n) = \sqrt{\frac{2}{N}}\alpha(k)\cos\frac{(2n+1)\pi k}{2N}$$

for $n = 0,\cdots,N-1$ and $k = 0,\cdots,N-1$

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

The above decomposition to a triple matrix product results in a reduction in computational complexity to $2N^3$ multiplications and has been used in practically all implementations. Since $2N^3$ multiplications must be performed in N^2 clock cycles (or input sample periods), the computational requirement of such an implementation is $2N$ multiplies per input sample. For example, for an input sample rate of 100 MHz, the computation requirement is 1.6 GOPS, where each operation is a multiply-accumulate.

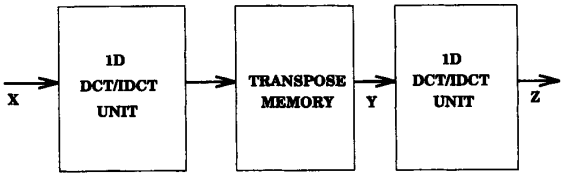

Fig. 1. 1-D DCT/IDCT unit.

Each $N \times N$ matrix-matrix multiply is separated into N matrix-vector products. A standard block diagram of a DCT processor is shown in Fig. 1 where the computation of the 2-D DCT has been broken down into two 1-D DCT's. The first computes $Y = AX$ and the second computes $Z = YA^T$. Each 1-D DCT unit must be capable of computing N multiplies per input sample. If the input block X is scanned column by column, the intermediate product $Y = AX$ is also computed column by column. However, since an entire row of Y has to be computed prior to the evaluation of the next 1-D DCT (given by $Z = YA^T$), the intermediate result Y must be stored in an on-chip buffer. Since columns are written into the buffer and rows are read from it, it is commonly called the *transpose memory*.

The basic computation performed by the DCT is the evaluation of the $(N \times N)$ matrix by $(N \times 1)$ vector product. Let us first consider the computation of the triple matrix product $Z = AXA^T$ for the DCT or $Z = A^TXA$ for the IDCT. This is computed as $Y = AX$ and $Z = YA^T$ for the DCT and $Y = A^TX$ and $Z = YA$ for the IDCT as in Fig. 1. The first 1-D DCT/IDCT unit operates on rows of A (or A^T) and columns of X, while the second 1-D DCT unit operates on rows of Y and columns of A^T (or A). Since a column of A^T (A) is equivalent to a row of A (A^T), the second 1-D DCT/IDCT unit is unnecessary if we can *multiplex* the first 1-D DCT/IDCT unit between a column of X and a row of Y as shown in Fig. 2. However, the 1-D DCT/IDCT unit must now process samples at *twice* the input sample rate, i.e., the 1-D DCT/IDCT unit must be capable of computing $2N$ multiplies per input sample. We can exploit certain features of the 1-D DCT/IDCT to reduce the computational overhead of the basic building block as described below.

B. Computation of the DCT

The 8×8 DCT matrix can be written as

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

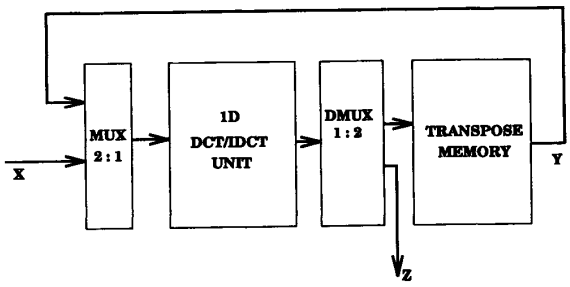

Fig. 2. Multiplexed DCT/IDCT unit.

where

$$\begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \end{bmatrix} = \sqrt{\frac{2}{N}} \begin{bmatrix} \cos\frac{\pi}{4} \\ \cos\frac{\pi}{16} \\ \cos\frac{\pi}{8} \\ \cos\frac{3\pi}{16} \\ \cos\frac{5\pi}{16} \\ \cos\frac{3\pi}{8} \\ \cos\frac{7\pi}{16} \end{bmatrix}.$$

Even rows of A are even-symmetric and odd rows are odd-symmetric. Thus, by exploiting this symmetry in the rows of A, and separating the even and odd rows, we get

$$\begin{bmatrix} Y(0) \\ Y(2) \\ Y(4) \\ Y(6) \end{bmatrix} = \begin{bmatrix} a & a & a & a \\ c & f & -f & -c \\ a & -a & -a & a \\ f & -c & c & -f \end{bmatrix} \begin{bmatrix} X(0)+X(7) \\ X(1)+X(6) \\ X(2)+X(5) \\ X(3)+X(4) \end{bmatrix}$$

$$\begin{bmatrix} Y(1) \\ Y(3) \\ Y(5) \\ Y(7) \end{bmatrix} = \begin{bmatrix} b & d & e & g \\ d & -g & -b & -e \\ e & -b & g & d \\ g & -e & d & -b \end{bmatrix} \begin{bmatrix} X(0)-X(7) \\ X(1)-X(6) \\ X(2)-X(5) \\ X(3)-X(4) \end{bmatrix}.$$

The number of multiplications are halved since the $(N \times N) \times (N \times 1)$ matrix-vector multiply has been replaced by two $((N/2) \times (N/2)) \times ((N/2) \times 1)$ matrix-vector multiplies. Moreover, these two can be computed in parallel.

C. Computation of the IDCT

The 1-D IDCT can be rewritten as follows:

$$\begin{bmatrix} Y(0) \\ Y(1) \\ Y(2) \\ Y(3) \end{bmatrix} = \begin{bmatrix} a & c & a & f \\ a & f & -a & -c \\ a & -f & -a & c \\ a & -c & a & -f \end{bmatrix} \begin{bmatrix} X(0) \\ X(2) \\ X(4) \\ X(6) \end{bmatrix} + \begin{bmatrix} b & d & e & g \\ d & -g & -b & -e \\ e & -b & g & d \\ g & -e & d & -b \end{bmatrix} \begin{bmatrix} X(1) \\ X(3) \\ X(5) \\ X(7) \end{bmatrix}$$

$$\begin{bmatrix} Y(7) \\ Y(6) \\ Y(5) \\ Y(4) \end{bmatrix} = \begin{bmatrix} a & c & a & f \\ a & f & -a & -c \\ a & -f & -a & c \\ a & -c & a & -f \end{bmatrix} \begin{bmatrix} X(0) \\ X(2) \\ X(4) \\ X(6) \end{bmatrix} - \begin{bmatrix} b & d & e & g \\ d & -g & -b & -e \\ e & -b & g & d \\ g & -e & d & -b \end{bmatrix} \begin{bmatrix} X(1) \\ X(3) \\ X(5) \\ X(7) \end{bmatrix}.$$

The number of multiplies is halved since the $(N \times N) \times (N \times 1)$ matrix-vector multiply has been replaced by two $((N/2) \times (N/2)) \times ((N/2) \times 1)$ matrix-vector multiplies. (Notice that the right-hand sides of the two preceeding equations differ only by a $\pm$ sign.) These can be computed in parallel as well. Since the computation for both the DCT and the IDCT has been halved, the 1-D DCT/IDCT unit in Fig. 2 now needs to compute only N multiplies per input sample. The savings in the multiplications comes at the expense of some data ordering and reordering that is required before and after the 1-D DCT/IDCT unit. This decomposition follows the fast algorithm described in [4].

III. Architecture

Both the DCT and the IDCT require the computation of two (4×4) by (4×1) matrix-vector products. Our matrix-vector multipliers are designed to compute these matrix-vector products during four clock cycles. The two matrix-vector products are performed in parallel. For four clock cycles, the matrix-vector multipliers operate on a column of X. For the next four clock cycles, the matrix-vector multipliers operate on a row of Y. This is the multiplex operation illustrated in Fig. 2. From the discussions in Sections II-B and II-C we see that the DCT requires several additions/subtractions *prior* to the matrix-vector multiply whereas the IDCT requires the grouping of even and odd samples. This is performed by our Data Reorder Unit (DRU). Similarly, the DCT requires the regrouping of even and odd coefficients *after* the matrix-vector multiply while the IDCT requires some additions/subtractions. These are performed in the Inverse Data Reorder Unit (IDRU). The completed forward (reverse) transformed coefficients Z are written off-chip while Y is written into the transpose memory. All these operations occur in a seamless fashion. Input pixels $X(n)$ enter the chip a single pixel at a time and transformed coefficients $Z(n)$ leave the chip a single pixel at a time.

The overall architecture (and floorplan) of the chip is shown in Fig. 3. It consists of the following blocks:

1) Data Reorder Unit (DRU)
2) Two Matrix-Vector Multiplier Units
3) Inverse Data Reorder Unit (IDRU)
4) Transpose Memory.

Since the processor can operate in both DCT and IDCT modes, it is desirable that both modes share a common data path wherever possible. The similarities in the computations for the DCT and IDCT are used to develop a common architecture for the above units as described below.

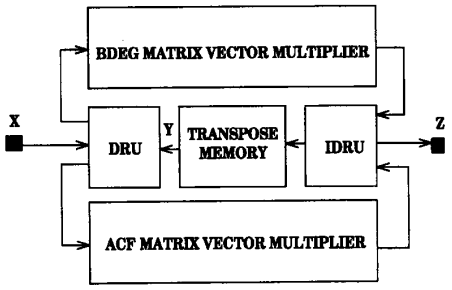

Fig. 3. DCT/IDCT processor architecture and floorplan.

A. Data Reorder Unit (DRU)

The DRU receives its inputs from the columns of X (off-chip) and the rows of Y (on-chip). Inputs arrive sequentially, a single pixel per clock cycle. The pixels of Y are delayed by four clock cycles such that $Y(0)$ arrives four clock cycles after $X(0)$. The operation of the DRU, shown in Fig. 4, occurs as follows:

Multiplexer control signal *INSEL* is high for four clock cycles and low for the next four clock cycles. It is generated by dividing the clock by eight. For the first four clock cycles MUXA selects the X inputs and MUXB selects the Y inputs. $X(0)$, $X(1)$, $X(2)$, and $X(3)$ are written into the LIFO in that order. For the next four clock cycles $Y(0)$, $Y(1)$, $Y(2)$, and $Y(3)$ are written into the LIFO and $X(3)$, $X(2)$, $X(1)$, and $X(0)$ are read from the LIFO. The outputs of the LIFO and MUXB are fed to two adders and two multiplexers.The adders compute the addition/subtraction pairs $(X(3) \pm X(4), X(2) \pm X(5), X(1) \pm X(6), X(0) \pm X(7))$ for the first four clock cycles and $(Y(3) \pm Y(4), Y(2) \pm Y(5), Y(1) \pm Y(6), Y(0) \pm Y(7))$ over the next four clock cycles. This is the preprocessing required by the DCT. The multiplexer MUXC and MUXD outputs are $(X(4), X(2), X(6), X(0))$, and $(X(3), X(5), X(7)$ and $X(1))$, respectively, during the first four clock cycles, then $(Y(4), Y(2), Y(6), Y(0))$, and $(Y(3), Y(5), Y(7)$, and $Y(1))$ during the next four clock cycles. This grouping of even and odd pixels is required by the IDCT.

B. Matrix-Vector Multiplication

The elements of the DCT/IDCT matrices consist of seven distinct values: a, b, c, d, e, f, and g. These are elements of the first column of matrix A. All other elements (or coefficents) are permutations (possibly negated) of these seven values.

The decomposition of the 1-D DCT/IDCT by the fast algorithm results in two 4×4 matrices. The first 4×4 matrix has elements that can assume one of three values: a, c, f (possibly negated). Any input to this matrix-vector multiplier can multiply only one of three coefficients. The second 4×4 matrix has elements that assume one of four values: b, d, e, g (possibly negated). Similarly, any input to this matrix-vector multiplier can multiply only one of four coefficients.

The architecture exploiting the above observations for the ACF matrix-vector multiplier is shown in Fig. 5. It consists of three *hardwired* multipliers implementing multiplications

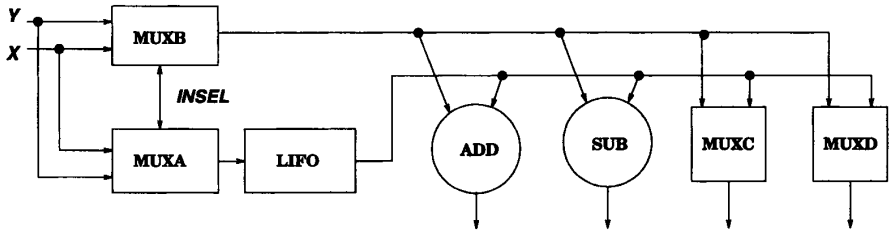

Fig. 4. Data reorder unit (DRU).

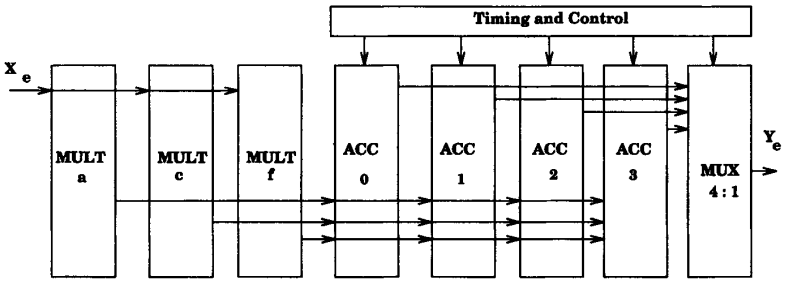

Fig. 5. ACF matrix-vector multiply unit.

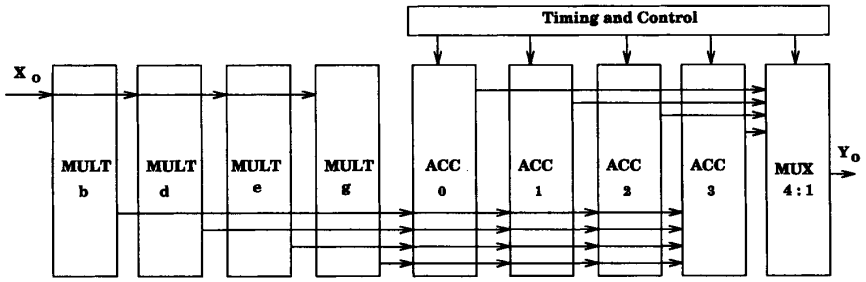

Fig. 6. BDEG matrix-vector multiply unit.

by fixed coefficients a, c, and f. An input x_e to the ACF multiplier bank is broadcast to all three multipliers. The products ax_e, cx_e, fx_e are available in parallel. A bank of four accumulators then selects one of these products to perform the various linear combinations required for the matrix-vector multiply over four clock cycles. Thus, at the end of four clock cycles, ACC0 has computed the sum of products corresponding to the first row of the matrix, ACC1 has computed the sum of products corresponding to the second row of the matrix, and so on. The sequence of inputs to the ACF matrix-vector multiplier bank for the DCT is: $X(3) + X(4)$, $X(2) + X(5)$, $X(1) + X(6)$, $X(0) + X(7)$, $Y(3) + Y(4)$, $Y(2) + Y(5)$, $Y(1) + Y(6)$, $Y(0) + Y(7)$. For the IDCT, the sequence of inputs is: $X(4)$, $X(2)$, $X(6)$, $X(0)$, $Y(4)$, $Y(2)$, $Y(6)$, $Y(0)$.

The architecture for the BDEG matrix-vector multiplier is shown in Fig. 6. It consists of four hardwired multipliers implementing multiplications by b, d, e, and g. An input x_o to the BDEG multiplier bank is broadcast to all four multipliers. The products bx_o, dx_o, ex_o, and gx_o are available in parallel to all four accumulators. The bank of four accumulators then selects one of these products to perform the various linear combinations over four clock cycles. As before, ACC0 computes the first sum of products, ACC1 computes the second sum of products, and so on. The sequence of inputs to the BDEG matrix-vector multiplier bank for the DCT is: $X(3) - X(4)$, $X(2) - X(5)$, $X(1) - X(6)$, $X(0) - X(7)$, $Y(3) - Y(4)$, $Y(2) - Y(5)$, $Y(1) - Y(6)$, $Y(0) - Y(7)$. For the IDCT, the sequence of inputs is: $X(3)$, $X(5)$, $X(1)$, $X(7)$, $Y(3)$, $Y(5)$, $Y(1)$, $Y(7)$. The order in which the accumulators select their products is described in Table I for the DCT and the IDCT where the first element in any column represents the product selected for the DCT and the second element, the product selected for the IDCT. The outputs of the accumulators are multiplexed into the IDRU for further post-processing.

TABLE I
INPUT SELECTION FOR ACCUMULATORS FOR THE DCT/IDCT

ACF ACCBANK FOR DCT/IDCT				*BDEG ACCBANK FOR DCT/IDCT*			
ACC0	ACC1	ACC2	ACC3	ACC0	ACC1	ACC2	ACC3
a/a	c/a	a/a	f/a	g/d	e/g	d/b	b/e
a/c	f/f	a/f	c/c	e/e	b/b	g/g	d/d
a/f	f/c	a/c	c/f	d/b	g/d	b/e	e/g
a/a	c/a	a/a	f/a	b/g	d/e	e/d	g/b

TABLE II
SIGNED DIGIT REPRESENTATION OF THE DCT COEFFICIENTS

coefficient	value	12 bit signed	SD representation	No. bits
a	0.35355	0.35351	$2^{-2}+2^{-4}+2^{-5}+2^{-7}+2^{-9}=0.35352$	5
b	0.49039	0.49023	$2^{-1}-2^{-7}-2^{-9}+2^{-13}=0.49036$	4
c	0.46194	0.46191	$2^{-1}-2^{-5}-2^{-7}+2^{-10}=0.46191$	4
d	0.41573	0.41552	$2^{-2}+2^{-3}+2^{-5}+2^{-7}+2^{-9}-2^{-12}=0.41577$	6
e	0.27779	0.27734	$2^{-2}+2^{-5}-2^{-8}+2^{-11}=0.27783$	4
f	0.19134	0.19091	$2^{-3}+2^{-4}+2^{-8}-2^{-14}=0.19135$	4
g	0.09755	0.09716	$2^{-4}+2^{-5}+2^{-8}-2^{-13}=0.09753$	4

1) Hardwired Multipliers: Since the multiplier coefficients are fixed, two implementations are possible: 1) a ROM-based LUT using the input as the address, and 2) hardcoding the multiplier coefficients. We first designed a ROM-based multiplier bank for each multiplier. However, this resulted in a larger multiplier area with only a marginal improvement in mean-square error when compared to the hardwired solution. Hence we adopted a hardwired multiplier approach with the coefficients represented by a Signed Digit (SD) code.

The radix-2 signed digit representation of a fractional number c has the general form

$$c = \sum s_k 2^{-k}$$

where $s_k \in \{-1, 0, 1\}$. It enables the representation of a number as the *algebraic* sum of several powers-of-two. The primary advantage of such a representation is that it allows most numbers to be represented with fewer nonzero digits [17]. Table II lists the hardwired coefficients and their signed-digit representations. The number of nonzero bits is dictated by accuracy and finite wordlength specifications, and has been determined by simulation. These simulations are described in greater detail in Section IV. The multiplier coefficients are accurate to four decimal places. From Table II we can see that four to six nonzero bits are adequate to represent the coefficients, which *significantly* reduces the hardware complexity of the multiplier bank.

2) Accumulator: A block diagram of the accumulator is shown in Fig. 7. It consists of a 4-to-1 multiplexer that selects between one of the (three) four products at any given time instance. A negative coefficient requires that the selected product be subtracted from the current accumulated value. This is accomplished by the conditional invert (xor) unit. The contents of the accumulator are latched into the output register (OUTREG) once every four clock cycles and then reset to zero. This is accomplished by the signal *reset*. The contents of the output register remain valid for four clock cycles until the next accumulated value is computed. For four clock cycles, the accumulator accumulates products corresponding to X. For the next four clock cycles, the accumulator is accumulating products corresponding to Y. The controls for the 4-to-1 multiplexer *sel*[0 : 1], conditional invert, and reset for both the DCT/IDCT modes of operation, are stored in a ROM.

C. Transpose Memory

The operation of the Transpose Memory can be explained if we visualize it as an 8×8 array. It is actually implemented as a 64-word DRAM. The first eight words of the DRAM

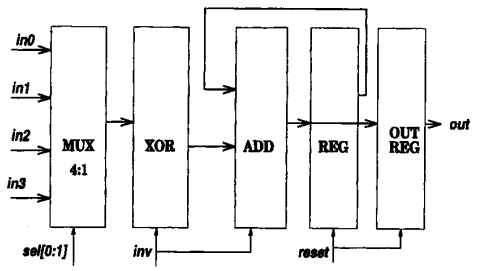

Fig. 7. Accumulator.

TABLE III
ACCURACY OF PROCESSOR

	Specification	L,H = -256,255	L = H = 300	L = H = 5
Peak Pixel Error	≤ 1	1	1	1
Peak Pixel Mean Square Error	≤ 0.06	0.0134	0.0153	0.0139
Overall Mean Square Error	≤ 0.02	0.0104	0.0101	0.0028
Peak Pixel Mean Error	≤ 0.015	0.0133	0.0125	0.0139
Overall Mean Error	≤ 0.0015	0.00096	0.0011	0.0011

correspond to the first row of the array, the second eight words, to the second row, and so on. Let *mode1* be a sequence of accesses to locations {0, 1, 2, 3, 4, 5, 6, 7, 8,$\cdots$} in that order. This corresponds to scanning rows starting at the top left corner. Let *mode2* be accesses to locations {0, 8, 16, 24, 32, 40, 48, 56, 1, 9,$\cdots$} in that order. This corresponds to scanning columns starting at the top left corner. The transposition occurs as follows. For the first 64 clock cycles data is read out according to *mode1*. New data (that needs to be transposed) is also written according to *mode1*. A write always follows a read; i.e., a read from a location is always followed by a write to that location. For the next 64 clock cycles, reads and writes occur according to *mode2*. It is quite easy to see that the data currently being read out is the transpose of the data that was written in during the previous 64 clock cycles. The latency of the transpose operation is 64 clock cycles. The two modes of addresses are generated by circulating a "one" through a shift register chain and tapping the read/write signals from the appropriate shift registers.

IV. FINITE WORDLENGTH SIMULATIONS

The Joint CCITT/ISO committee (also known as JPEG) has established a compression standard to which all decoders must comply. This includes stringent requirements on the error incurred by the IDCT. Compliance testing involves computing the IDCT on a given set of randomized inputs and measuring the error between the output of the implementation and a reference test output. These specifications are summarized in Table III. The error is tabulated as an 8×8 array so that the error performance of various pixel locations in a block can be evaluated.

We have carried out finite wordlength simulations to ensure that our implementation meets these specifications. Simulations were carried out on a set of 10 000 8×8 blocks. These blocks were generated by the random number generator specified in [10]. Simulations were carried out for the three

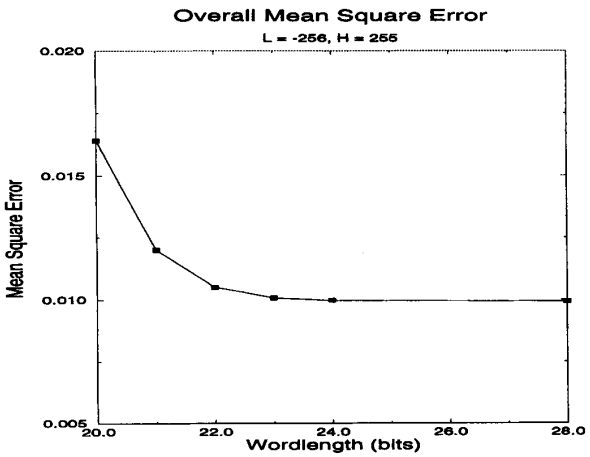

Fig. 8. Overall mean square error versus wordlength.

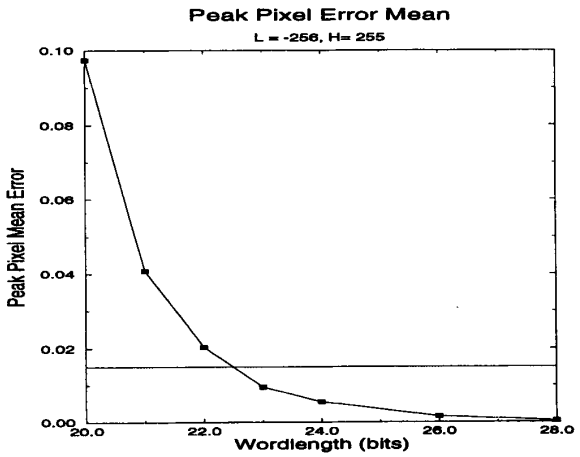

Fig. 10. Peak pixel mean error versus wordlength.

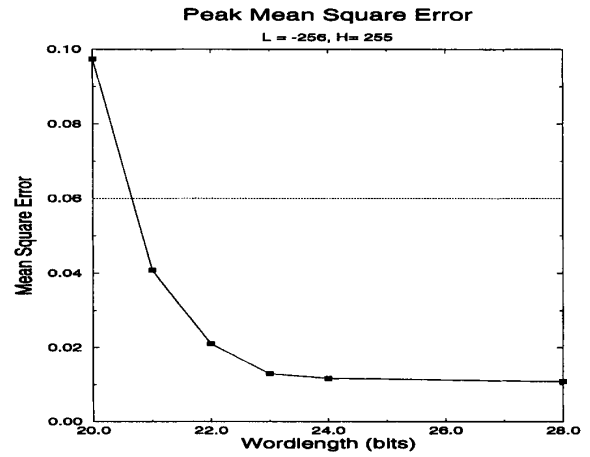

Fig. 9. Peak mean square error versus wordlength.

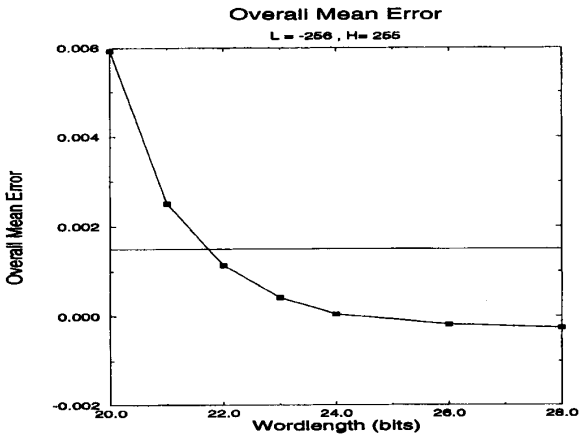

Fig. 11. Overall mean error versus wordlength.

cases a) $L = -256, H = 255$, b) $L = H = 5$, and c) $L = H = 300$, where L and H are the lower and upper bounds on the random numbers (i.e., $-L < X < H$). The results of these simulations are summarized in Figs. 8–11 for the case $L = -256$, $H = 255$.

There are two sources of error in our implementation: a) coefficient quantization, b) finite wordlength. The effect of the coefficient quantization is to plateau out the M.S.E curve with increasing wordlength, and represents a lower bound on the error. This can be improved by adding more nonzero bits to the Signed Digit representation of the coefficients.

A wordlength of 22 b meets the Peak Mean Square Error, Overall Mean Square Error, and Overall Mean specifications. We found that the dc term (pixel location (0, 0)) consistently has a nonzero mean, which degrades the Peak Mean Error performance of the processor. This can easily be corrected by adding a bias (=0.02) to this term, which improves both the Peak Mean Error as well as the Peak Mean Square Error. The error characteristics of the core processor are summarized in Table III.

V. Performance Comparison and Core Characteristics

The popularity of the DCT has resulted in numerous implementations spanning a broad spectrum of architectures. For example, the implementation in [3] consists of a single multiply-accumulate element clocking at 250 MHz. The implementation in [14] is a highly parallel approach with 16 parallel multipliers clocking at 20 MHz on chip. Others [1], [2] choose some other optimal combination of the number of multipliers and clock rate to achieve real-time capability. Since these video signal processors perform various other signal processing functions, we limit the scope of our comparison to processors specifically designed to implement only the DCT at HDTV rates. A high data rate (55 MHz) DCT/IDCT processor in 2 μm technology is reported in [6]. It is not included in Table IV since a direct scaling of area and speed does not always result in a fair comparison.

In [8], two matrix-vector product modules are used to perform the matrix-vector multiplies required by the DCT. Each module consists of four MAC's. However, this approach results in somewhat larger areas. The most commonly proposed

TABLE IV
PROCESSOR COMPARISON

Implementation	Technology	Core Area	Transistors	Clock Rate
T. Miyazaki *et al.* [8]	0.8-μm	160.89 mm^2	180,000	50 MHz
S. Uramoto *et al.* [7]	0.8-μm	21.12 mm^2	102,000	100 MHz
M. Matsui *et al.* [9]	0.8-μm	13.3 mm^2	120,000	200 MHz
Our Implementation	0.8-μm	10.00 mm^2	67,000	100 MHz

TABLE V
CORE CHARACTERISTICS

Inputs	9 bits (DCT), 12 bits (IDCT)
Outputs	12 bits (DCT), 9 bits (IDCT)
Internal Wordlength	22 bits
Technology	0.8-μm CMOS, triple metal
No. of Transistors	67,000
Core Size	10 mm^2
Clock Rate	100 MHz
Mode Selection	DCT or IDCT
Block Size	8×8
Accuracy	CCITT compliant

TABLE VI
PROCESSOR I/O SPECIFICATIONS

Function	Number
Input	12
Output	12
Clock	1
Power	8
Ground	8
DCT/IDCT control	1
Start	1

high-speed implementation is based on a distributed arithmetic Look-Up-Table based architecture [16]. These architectures result in compact, accurate, and high-speed implementations. For example, the processor in [7] uses distributed arithmetic LUT techniques along with a fast algorithm to achieve a 100 MHz sample rate. The processor in [9] also uses a distributed arithmetic based LUT approach with low-swing differential logic to achieve even higher speeds and smaller sizes. The main features of these implementations are summarized in Table IV.

Our DCT/IDCT processor has been designed using a 0.8-μm CMOS cell library. The library includes high-speed parameterizable carry-select adders, registers and multiplexers, among other cells. The 22-bit carry-select adder determines the critical path, which is well under 10 ns. This permits the realization of the 100 MHz clock and sample rate. The layout has been generated using the Mentor Graphics GDT design tools. The core measures under 10 mm^2 with a transistor count of 67 000 transistors. The core characteristics have been summarized in Table V. The chip I/O specifications are summarized in Table VI and reflect the simplicity of control and I/O interface where a single pin determines the forward or inverse modes of operation. A single start pulse is required to initialize the chip. All other control signals are generated internally on-chip.

VI. CONCLUSION

The 10 mm^2 core area of our 2-D DCT/IDCT processor compares very favorably with core areas reported in the recent literature. This results from a) the use of a fast algorithm that permits the use of only one 1-D DCT/IDCT unit for the 2-D transform, and b) an architecture that results in a common datapath that is shared by both the DCT and the IDCT. The architecture requires seven hardwired multipliers and eight multiply-accumulates (MAC's). A significant savings in area has been obtained by allocating just enough precision to the hardwired multipliers that the implementation is compliant with the existing standards. The processor has been designed with a parameterizable input wordlength for applications that might need more precision. Clocking at 100 MHz, the processor can meet the real time processing requirements of currently proposed HDTV systems.

REFERENCES

[1] K. Kikuchi *et al.*, "A single chip 16-bit 25 ns realtime video/image signal processor," in *ISSCC Dig. Tech. Papers*, Feb. 1989, pp. 170–171.
[2] S. Nakagawa *et al.*, "A 50 ns video signal processor," in *ISSCC Dig. Tech. Papers*, Feb. 1989, pp. 168–169.
[3] J. Goto *et al.*, "A 250 MHz 16b 1 million transistor BiCMOS super high speed video signal processor," in *ISSCC Dig. Tech. Papers*, Feb. 1991, pp. 254–255.
[4] W. Chen *et al.*, "A fast computational algorithm for the discrete cosine transform," *IEEE Trans. Commun.*, vol. COM-25, pp. 1004–1009, Sept. 1977.
[5] A. Heiman and K. Rose, "A look-up based universal real-time transformer for image coding," *IEEE J. Select. Areas Commun.*, vol. SAC-5, pp. 1207–1213, Aug. 1987.
[6] D. Slawecki and W. Li, "DCT/IDCT processor design for high data rate image coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 2, pp. 135–146, June 1992.
[7] S. Uramoto *et al.*, "A 100 MHz 2-D discrete cosine transform core processor," *IEEE J. Solid-State Circuits*, vol. 27, pp. 492–499, Apr. 1992.
[8] T. Miyazaki *et al.*, "DCT/IDCT processor for HDTV developed with DSP silicon compiler," *J. VLSI Signal Process.*, vol. 5, pp. 39–46, June 1993.
[9] M. Matsui *et al.*, "200 MHz compression macrocells using low-swing differential logic," in *ISSCC Dig. Tech. Papers*, Feb. 1994, pp. 254–255.
[10] CCITT Recommendation H.261, 1990 (Annex 1).
[11] ISO/IEC JTC1/SC29/WG10, JPEG Committee Draft CD10918, 1991.
[12] ISO/IEC JTC1/SC29/WG11, MPEG Committee Draft CD11172, 1991.
[13] M. Liou, "Overview of the $p \times 64$ kbit/s video coding standard," *Commun. ACM*, vol. 34, pp. 59–63, Apr. 1991.
[14] T. Yamazaki *et al.*, "Multi-purpose inner-product LSI for video rate signal processing," in *Proc. IEEE Workshop on VLSI Signal Process.*, Nov. 1990, pp. 1–12.
[15] N. Ahmed *et al.*, "Discrete cosine transform," *IEEE Trans. Comput.*, vol. C-23, pp. 90–93, Jan. 1974.
[16] M. T. Sun *et al.*, "VLSI implementation of a 16×16 discrete cosine transform," *IEEE Trans. Circuits Syst.*, vol. 36, pp. 610–617, Apr. 1989.
[17] H. Samueli, "An improved search algorithm for the design of multiplierless FIR filters with power-of-two coefficients," *IEEE Trans. Circuits Syst.*, vol. 36, pp. 1044–1047, July 1989.

Avanindra Madisetti received the B.Tech. (Hons) degree in electronics and electrical communication engineering from Indian Institute of Technology (IIT) Kharagpur, India, and the M.S. degree from the University of California, Davis.

He is currently a Graduate Student Researcher at the University of California, Los Angeles, where he is pursuing the Ph.D. degree in electrical engineering. His research interests are in VLSI architectures for digital signal processing, video compression, and digital communication systems.

Alan N. Willson, Jr. (S'66–M'67–SM'73–F'78) received the B.E.E. degree from the Georgia Institute of Technology, Atlanta, in 1961, and the M.S. and Ph.D. degrees from Syracuse University, Syracuse, NY, in 1965 and 1967, respectively.

From 1961 to 1964 he was with IBM, Poughkeepsie, NY. He was an Instructor in Electrical Engineering at Syracuse University from 1965 to 1967. From 1967 to 1973 he was a Member of the Technical Staff at Bell Laboratories, Murray Hill, NJ. Since 1973 he has been on the faculty of the University of California, Los Angeles, where he is now Professor of Engineering and Applied Science in the Electrical Engineering Department. In addition, he served the UCLA School of Engineering and Applied Science as Assistant Dean for Graduate Studies, from 1977 through 1981, and is currently Associate Dean of Engineering. He has been engaged in research concerning computer-aided circuit analysis and design, the stability of distributed circuits, properties of nonlinear networks, theory of active circuits, digital signal processing, analog circuit fault diagnosis, and integrated circuits for signal processing. He is editor of the book *Nonlinear Networks: Theory and Analysis* (IEEE Press, 1974).

Dr. Willson is a member of Eta Kappa Nu, Sigma Xi, Tau Beta Pi, the Society for Industrial and Applied Mathematics, and the American Society for Engineering Education. From 1977 to 1979 he served as Editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS. In 1980 he was General Chairman of the 14th Asilomar Conference on Circuits, Systems, and Computers. During 1984 he served as President of the IEEE Circuits and Systems Society. He was the recipient of the 1978 and 1994 Guillemin-Cauer Awards of the IEEE Circuits and Systems Society, the 1982 George Westinghouse Award of the American Society for Engineering Education, the 1982 Distinguished Faculty Award of the UCLA Engineering Alumni Association, the 1984 Myril B. Reed Best Paper Award of the Midwest Symposium on Circuits and Systems, and the 1985 and 1994 W. R. G. Baker Awards of the IEEE.