A NEW DESIGN AND IMPLEMENTATION OF 8×8 2-D DCT/IDCT

Yung-Pin Lee Liang-Gee Chen Mei-Juan Chen
Chung-Wei Ku
DSP/IC Design Lab, Dept. of Electrical Engineering
National Taiwan University
Taipei, Taiwan, R.O.C.

Abstract - We describe a novel 8×8 2-D DCT/IDCT architecture based on the direct 2-D approach and the rotation technique. The computational complexity is reduced by taking advantage of the special attribute of complex number. Unlike other direct approach, the proposed architecture is regular, hence, it is suitable for VLSI implementation.

INTRODUCTION

Among various transform techniques for image compression, the discrete cosine transform (DCT) is the most popular and effective one in practical applications because it gives an almost optimal performance and can be implemented at an acceptable cost. There are three methods to realize $N \times N$ 2-D DCT: (1) indirect method by the row-column decomposition[1],[2]; (2) direct method [3],[4], such as using polynomial transform [3]; (3) using other transforms, such as DFT and DHT. The indirect method has the advantage of regularity for VLSI implementation. Therefore, most chips for 2-D DCT had been implemented by indirect method[1],[2]. However, the computation amount of the indirect method is more than that of the direct method. The direct method requires less computations, but it incurs the irregularity. Thus, the direct method is not suitable for chip implementation. Nevertheless, the feature of low computation complexity is still attractive. The fact motivates that a low-computation and regular 2-D DCT structure is researched recently.

In this paper, we propose a cost-effective architecture for 8×8 2-D DCT architecture which bears both the advantages of high regularity and less computation amount. At first, the real number input is mapped into complex number in the $N \times N$ 2-D DCT[3]. Then the computation complexity can be reduced by the rotation techniques in the complex number system. For 8×8 2-D DCT/IDCT, further modification is required to make the architecture more regular and this results in the fact that the architecture can be folded to an economically-allowable size for VLSI implementation. The finite wordlength analysis demonstrates that the proposed architecture requires less internal bits than other methods. In the following section, we illustrate that an $N \times N$ 2-D DCT/IDCT can be realized by only N N-point 1-D D-

0-7803-3134-6/96 $5.00 © 1996 IEEE

CT/IDCTs and some additional summations. With some modifications, we can obtain a more regular architecture for 8×8 2-D DCT/IDCTs. Finally, we analyze the internal wordlength problem.

METHODOLOGY

The Mapping of Input Data

The 2-D DCT of an $N \times N$ real signal x_{n_1,n_2} is defined as

$$X_{k_1,k_2} = \frac{2}{N} c(n_1)c(n_2) \sum_{n_1=0}^{N-1} \sum_{n_2=0}^{N-1} x_{n_1,n_2} \cos\left[\frac{2\pi(2n_1+1)k_1}{4N}\right] \cos\left[\frac{2\pi(2n_2+1)k_2}{4N}\right], \tag{1}$$

$$n_1, n_2, k_1, k_2 = 0, 1, \cdots, N-1,$$
$$c(0) = \frac{1}{\sqrt{2}}, \quad \text{and} \quad c(n) = 1 \text{ for } n \neq 0,$$

For convenience, we introduce Y_{k_1,k_2} by neglecting the kernal factor $2c(n_1) \cdot c(n_2)/N$ so that

$$Y_{k_1,k_2} = \sum_{n_1=0}^{N-1} \sum_{n_2=0}^{N-1} x_{n_1,n_2} \cos\left[\frac{2\pi(2n_1+1)k_1}{4N}\right] \cos\left[\frac{2\pi(2n_2+1)k_2}{4N}\right], \tag{2a}$$

and

$$X_{k_1,k_2} = \frac{2}{N} c(n_1)c(n_2) Y_{k_1,k_2}. \tag{2b}$$

In the following, we will assume N to be a power of 2. Using the permutation[3], signal x_{n_1,n_2} can be permuted as:

$$\begin{aligned}
y_{n_1,n_2} &= x_{2n_1,2n_2} && n_1 = 0, \cdots, N/2-1, && n_2 = 0, \cdots, N/2-1 \\
&= x_{2N-2n_1-1,2n_2} && n_1 = N/2, \cdots, N-1, && n_2 = 0, \cdots, N/2-1 \\
&= x_{2n_1,2N-2n_2-1} && n_1 = 0, \cdots, N/2-1, && n_2 = N/2, \cdots, N-1 \\
&= x_{2N-2n_1-1,2N-2n_2-1} && n_1 = N/2, \cdots, N-1, && n_2 = N/2, \cdots, N-1.
\end{aligned}$$

(2a) can be rewritten as:

$$Y_{k_1,k_2} = \sum_{n_1=0}^{N-1} \sum_{n_2=0}^{N-1} y_{n_1,n_2} \cos\left[\frac{2\pi(4n_1+1)k_1}{4N}\right] \cos\left[\frac{2\pi(4n_2+1)k_2}{4N}\right]. \tag{3}$$

Now consider the following expression:

$$U_{k_1,k_2} = \sum_{n_1=0}^{N-1} \sum_{n_2=0}^{N-1} y_{n_1,n_2} \mathrm{W}_{4N}^{(4n_1+1)k_1+(4n_2+1)k_2}, \quad \text{where} \quad \mathrm{W}_{4N} = \exp(-j\frac{2\pi}{4N}). \tag{4}$$

We can easily compute Y_{k_1,k_2} from U_{k_1,k_2} by the following set of expressions:

$$\begin{aligned}
Y_{k_1,k_2} &= \frac{1}{2}[\mathrm{Re}(U_{k_1,k_2}) - \mathrm{Im}(U_{N-k_1,k_2})], \\
Y_{k_1,N-k_2} &= \frac{1}{2}[-\mathrm{Im}(U_{k_1,k_2}) - \mathrm{Re}(U_{N-k_1,k_2})].
\end{aligned} \tag{5}$$

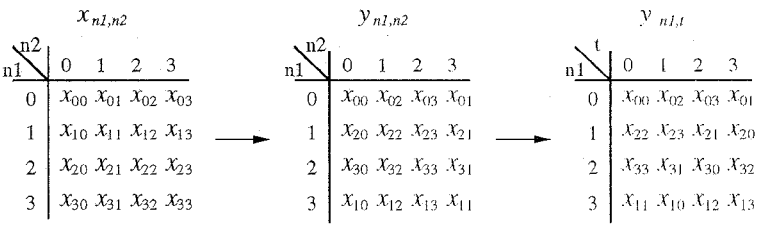


Figure 1: The mapping from x_{n_1,n_2} to $y_{n_1,t}$ when N=4.

Note that (5) requires U_{k_1,k_2} in (4) to be computed for all k_1 and only a sufficient subset of k_2 such that $\{k_2, N-k_2\}$ cover all possible values of k_2[3].

The Proposed 2-D DCT Algorithm

The signal y_{n_1,n_2} is mapped as $y_{n_1,t}$ through the following relation

$$4n_2+1 = (4t+1)(4n_1+1) \bmod 4N, \quad \text{where} \quad 0 \le t, n_1, n_2 \le N-1, \tag{6}$$

The mapping from n_2 to t is one-to-one. However, with different n_1, the mapping order is not the same. Fig. 1 shows the mapping of inputs from x_{n_1,n_2} to $y_{n_1,t}$ when N=4.

By substituting $y_{n_1,t}$ for y_{n_1,n_2} and then (4) can be rewritten as:

$$U_{k_1,k_2} = \sum_{n_1=0}^{N-1}\sum_{t=0}^{N-1} y_{n_1,t} W_{4N}^{(4n_1+1)[k_1+(4t+1)k_2]} \tag{7a}$$

$$= \sum_{t=0}^{N-1}\left[\sum_{n_1=0}^{N-1} y_{n_1,t} W_{4N}^{(4n_1+1)[k_1+(4t+1)k_2]}\right]. \tag{7b}$$

In (7b), $[k_1+(4t+1)k_2]$ is no longer in the range from 0 to $N-1$, which is a common attribute of the ordinary transform. Consider the following relation

$$k_1+(4t+1)k_2 = aN+b, \quad \text{where} \quad a \in \text{integer} \quad \text{and} \quad 0 \le b \le N-1.$$

By substituting the above relation into (7b), we can obtain

$$U_{k_1,k_2} = \sum_{t=0}^{N-1}(-j)^a\left[\sum_{n_1=0}^{N-1} y_{n_1,t} W_{4N}^{(4n_1+1)b}\right]. \tag{8}$$

Let the summation of n_1 to be represented by $U'_{t,b}$. Then we can find

$$U'_{t,b} = \sum_{n_1=0}^{N-1} y_{n_1,t} W_{4N}^{(4n_1+1)b}, \quad 0 \le b \le N-1$$

$$= \sum_{n_1=0}^{N-1} y_{n_1,t}\left(\cos\frac{2\pi(4n_1+1)b}{4N} - j\sin\frac{2\pi(4n_1+1)b}{4N}\right)$$

$$= \sum_{n_1=0}^{N-1} y_{n_1,t}\left(\cos\frac{2\pi(4n_1+1)b}{4N} - j\cos\frac{2\pi(4n_1+1)(N-b)}{4N}\right).$$

It is clear that $U'_{t,b}$ is a complex number. However, its real part is indeed an N-point 1-D DCT, and its imaginary part is relative to the real part by the relation:

$$\begin{aligned} \mathrm{Im}\{U'_{0,b}\} &= 0 \\ \mathrm{Im}\{U'_{t,b}\} &= -\mathrm{Re}\{U'_{N-t,b}\}, \quad 1 \le t, b \le N-1. \end{aligned}$$

This reveals that $U'_{t,b}$ can be achieved by calculating N-point 1-D DCT. Since the term $(-j)^a$ belongs sign operation, only additions and subtractions are needed. Therefore, an $N \times N$ 2-D DCT can be realized by N N-point 1-D DCT's with some additions. Nevertheless, the row-column method needs $2N$ N-point 1-D DCT's. Similar result has been deduced in [4] with different approach, but its structure is not regular, and so is unsuitable for VLSI implementation. To overcome this problem, the proposed algorithm develops a regular architecture as illustrated in the following section.

ARCHITECTURE OF AN 8×8 2-D DCT/IDCT

To realize the 2-D DCT, the additions are always irregular, especially when N is large. However, most proposed video compression standards, such as H.261, JPEG[6], MPEG-1 and MPEG-2, need only 8×8 2-D DCT/IDCT. In this section, we present a regular structure for 8×8 2-D DCT/IDCT and further can fold the architecture to one forth of original size. The following subsection analyzes the internal wordlength problem.

The Parallel 8×8 2-D DCT Architecture

As mentioned in [3], when N=8, it is only to compute U_{k_1,k_2} for all k_1 but k_2=0, 1, 2, 4, and 5. Then the summation of t in (7a) with different k_2 can be expanded as follows:

$$U_{k_1,k_2} = \sum_{n_1=0}^{7} u_{n_1,k_2} \mathrm{W}_{32}^{(4n_1+1)(k_1+k_2)}, \quad k_2 = 0, 4, 1, 2, 5. \tag{9}$$

where

$$\begin{aligned} u_{n_1,0} &= \sum_{t=0}^{7} y_{n_1,t}, \qquad u_{n_1,4} = \sum_{t=0}^{7} (-1)^t y_{n_1,t}, \\ u_{n_1,2} &= (y_{n_1,0} + y_{n_1,4} - y_{n_1,2} - y_{n_1,6}) - j(y_{n_1,1} + y_{n_1,5} - y_{n_1,3} - y_{n_1,7}), \\ u_{n_1,1} &= (y_{n_1,0} - y_{n_1,4}) - j(y_{n_1,2} - y_{n_1,6}) \\ &\quad + \mathrm{W}_8[(y_{n_1,1} - y_{n_1,5}) - j(y_{n_1,3} - y_{n_1,7})](-1)^{n_1}, \\ u_{n_1,5} &= (y_{n_1,0} - y_{n_1,4}) - j(y_{n_1,2} - y_{n_1,6}) \\ &\quad - \mathrm{W}_8[(y_{n_1,1} - y_{n_1,5}) - j(y_{n_1,3} - y_{n_1,7})](-1)^{n_1}. \end{aligned}$$

For implementation, we partition the computation of (9) together with (5) into two stages: Stage 1 - Pre-addition: computing the values of $u_{n_1,0}$, $u_{n_1,4}$,

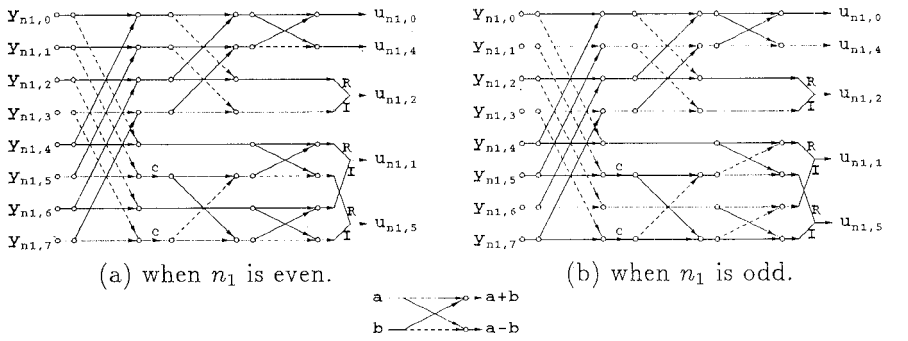


Figure 2: Stage 1 - Pre-addition.

$u_{n_1,2}$, $u_{n_1,1}$, and $u_{n_1,5}$; Stage 2 - Complex DCT and post-addition: computing the summation of n_1 and (5).

The Stage 1 for realizing pre-addition is shown in Fig. 2, where the term W_8 ($=1/\sqrt{2}$) requires two multiplications for each n_1 and both architectures for even n and odd n are somewhat different. In Stage 2, firstly consider $U_{k_1,2}$, $U_{k_1,1}$, and $U_{k_1,5}$. In the three cases, the inputs belong complex number, and the computations can be concluded as:

$$U_{k_1,k_2} = \sum_{n_1=0}^{7} u_{n_1,k_2} W_{32}^{(4n_1+1)(k_1+k_2)}, \quad \text{where } k_2 = 1, 2, 5. \tag{10}$$

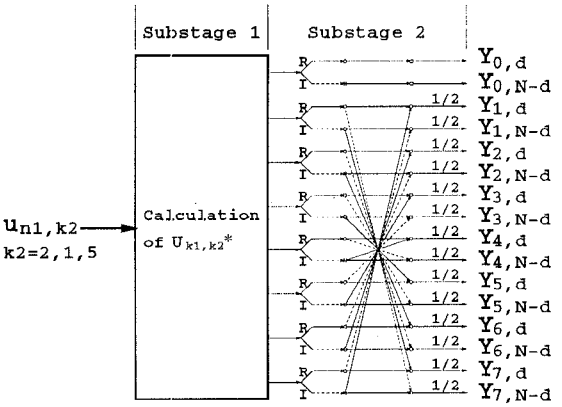
This equation is like 1-D DCT, except the input is complex number and the term $(k_1 + k_2)$ is not in the range from 0 to $N - 1$. Fig. 3(a) implies the computations of U_{k_1,k_2} with k_2=1, 2, 5, and which leads to Y_{k_1,k_2} for all k_1 and k_2= 2, 6, 1, 7, 5, and 3 from (5). In this figure, the first substage is to realize U_{k_1,k_2}, and the second substage is to realize the post-adder.

Since $u_{n_1,0}$ and $u_{n_1,4}$ are real number, we can use the same architecture in Fig. 3(a) together with the third substage, as shown in Fig. 3(b), to derive $Y_{k_1,0}$ and $Y_{k_1,4}$ for all k_1 directly by setting the input to be $u_{n_1,0} - ju_{n_1,4}$.

Between the Stage 1 and Stage 2, the interconnection is not local, but it is still regular. This feature of regularity will be utilized to fold the architecture in the next subsection. The following Stage 2 needs four 8-point complex DCT's with different k_2, and four same butterfly modules. Only the computation of Y_{k_1,k_2} with k_2=0,4 requires an additional butterfly stage. This reveals that the structure is more regular than [4].

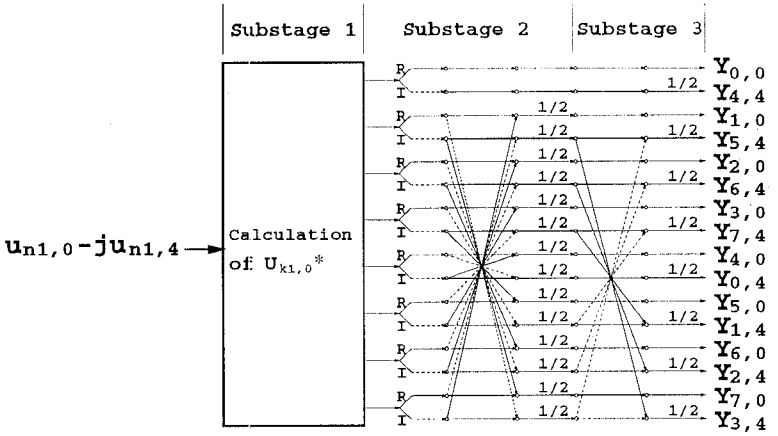
Folding the Parallel Architecture of 8×8 2-D IDCT

The previous subsection proposed a parallel architecture with 64 input data and 64 out data. To be suitable for VLSI implementation, the inherent problems of bandwidth limitation and hardware capacity should be solved. Folding technique is utilized to cope with the problem in the architecture.



$$* \; U_{k_1,k_2} = \sum_{n_1=0}^{7} u_{n_1,k_2} \mathrm{W}_{32}^{(4n_1+1)(k_1+k_2)}, \quad k_2 = 1, 2, 5.$$

(a) Computation of Y_{k_1,k_2} for all k_1 and k_2= 2, 6, 1, 7, 5, and 3.



$$* \; U_{k_1,0} = \sum_{n_1=0}^{7} (u_{n_1,0} - j u_{n_1,4}) \mathrm{W}_{32}^{(4n_1+1)k_1}.$$

(b) Computation of Y_{k_1,k_2} for all k_1 and k_2= 0 and 4.

Figure 3: Stage 2 - Computation of Y_{k_1,k_2}.

Table 1: Relation of k_1, k_2, a, and b.

	$k_2=0$			$k_2=1$			$k_2=2$			$k_2=5$		
k_1	k_1+k_2	a	b	k_1+k_2	a	b	k_1+k_2	a	b	k_1+k_2	a	b
0	0	0	0	1	0	1	2	0	2	5	0	5
1	1	0	1	2	0	2	3	0	3	6	0	6
2	2	0	2	3	0	3	4	0	4	7	0	7
3	3	0	3	4	0	4	5	0	5	8	1	0
4	4	0	4	5	0	5	6	0	6	9	1	1
5	5	0	5	6	0	6	7	0	7	10	1	2
6	6	0	6	7	0	7	8	1	0	11	1	3
7	7	0	7	8	1	0	9	1	1	12	1	4

Before folding the architecture, let's consider the computation of U_{k_1,k_2}, k_2= 0, 1, 2, 5. If the index k_1+k_2 is replaced by $8a+b$, both a and b are integer, and $0 \le b \le 7$, then we can derive

$$\begin{aligned} U_{k_1,k_2} &= \sum_{n_1=0}^{7} u_{n_1} W_{32}^{(4n_1+1)(k_1+k_2)}, \\ &\quad \text{where } u_{n_1} \text{ represents } u_{n_1,1},\ u_{n_1,2},\ u_{n_1,5}, \text{ or } u_{n_1,0} - j u_{n_1,4} \\ &= (-j)^a \sum_{n_1=0}^{7} u_{n_1} W_{32}^{(4n_1+1)b}, \quad 0 \le b \le 7. \end{aligned} \tag{11}$$

The relation among k_1, k_2, a, and b is illustrated in Table 1. Therefore, the computation of U_{k_1,k_2} can be achieved by the following three steps:
Step 1: Calculate the summation of n_1 in (11) for the values of b from 0 to 7. The output is denoted by U_b $(=\sum_{n_1=0}^{7} u_{n_1} W_{32}^{(4n_1+1)b})$.
Step 2: The output U_b from step 1 is multiplied by 1 or $-j$ according to the value of a. For example, in the case of k_2=2, the output needs to be multiplied by $-j$ when b=0, 1.
Step 3: The output from step 2 is rotated to make the output order to be the increasing order of k_1. This step can be implemented by a barrel shifter.

We then fold the structure and reverse the data flow to obtain the architecture of 8×8 2-D IDCT. The folded architecture is described in the Fig. 4. Among the four sets of the input, only the first set needs to pass through the Substage 3, so the multiplexers at the end of this substage can select the correct data to the next substage. The Substage 2 is similar to Fig. 3, but the direction of the data flow is reversed. The Substage 1 includes a barrel shifter, a set of multiplexers to select whether the data multiplying $-j$ or not, and a complex IDCT: $u_{n_1} = \frac{1}{8}\sum_{b=0}^{7} U_b W_{32}^{-(4n_1+1)b}$. For the sake of the regularity of the interconnection, the interconnection can be represented by four 4-by-4 transpose memories. Finally, the Stage 1 is obtained by reversing the data flow in the Fig. 2. The folded size is now reasonable for chip implementation.

To realize u_{n_1} in the Fig. 4, we rewrite U_b=p+jq, and then obtain

$$u_{n_1} = \frac{1}{8}\sum_{b=0}^{7} U_b W_{32}^{-(4n_1+1)b} = \frac{1}{8}\sum_{b=0}^{7}(p+jq)W_{32}^{-(4n_1+1)b}$$
$$= \frac{1}{8}\left[\sum_{b=0}^{7} pW_{32}^{-(4n_1+1)b} + j\sum_{b=0}^{7} qW_{32}^{-(4n_1+1)b}\right].$$

This computation requires two 1-D IDCTs together with 16 additions. Therefore, according to the Fig. 4, the proposed design can be implemented using two 1-D IDCTs and one transpose memory, which is just required by row-column design method together with 76 extra adders and 4 extra constant multipliers.

Finite Wordlength Analysis

When implementing an IDCT architecture, there are two inherent errors which will reduce the accuracy of outputs; one is quantization error of coefficients and another is finite internal wordlength. Therefore, the Joint CCITT/ISO committee has established a specification to evaluate the errors caused by finite wordlength in IDCT[6]. According to this specification, the proposed IDCT architecture requires 10,000 8×8 blocks of random numbers in the range from -256 to 255 as the input. The plots of overall mean square error versus internal wordlength with different coefficient wordlengths are shown in Fig. 5(a). The horizontal dashed line is the upper bound for overall mean square error. Fig. 5(b), (c), and (d) describe the analysis of peak mean square error, overall mean error, and peak mean error, respectively. The above analysis implies that 11-bit coefficient wordlength and 17-bit internal wordlength will be enough to satisfy the four error requirements.

Based on the same analysis, Table 2 shows the results for three different input ranges ([-256:+255], [-5:+5], and [-300:+300], which are also mentioned by Joint CCITT/ISO) with 12-bit coefficient wordlength and 18-bit internal wordlength. It is clear that all the values are much smaller than the specifications except for the overall mean error at the range of [-300:+300].

CONCLUSIONS

This research analyzes the $N \times N$ 2-D DCT/IDCT using direct method and develops a regular architecture. After folding the architecture, it will be very suitable for VLSI implementation. Traditionly, direct method has less computation complexity but irregularity; on the other hand, the row-column method is more regular with the penalty of requiring more computations. However, the proposed architecture has both the advantages of low computation complexity and high regularity. According to the specification by Joint CCITT/ISO committee on the IDCT, the proposed design needs only coefficient wordlength of 12 bits and internal wordlength of 18 bits.

Table 2: Accuracy Analysis for Three Difference Input Ranges

		Input range -L to +H		
	Spec.	L=256, H=255	L=H=5	L=H=300
Peak Pixel Error	≤ 1	1	1	1
Overall Mean Square Error	≤ 0.02	0.0089	0.0014	0.0103
Peak Mean Square Error	≤ 0.06	0.0117	0.0025	0.0135
Overall Mean Error	≤ 0.0015	0.0006	0.0005	0.0013
Peak Mean Error	≤ 0.015	0.0027	0.0016	0.0033

References

[1] 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.

[2] A. Madisetti and A. N. Willson, "A 100 MHz 2-D 8×8 DCT/IDCT processor for HDTV applications," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 5, pp. 158-165, Apr. 1995.

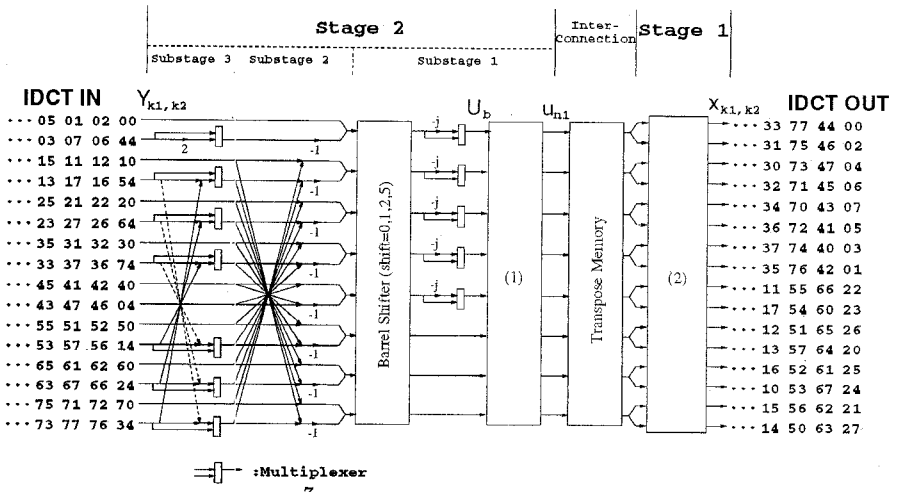
[3] P. Duhamel and C. Guillemot, "Polynomial transform computation of 2-D DCT," in *Proc. ICASSP'90*, pp. 1515-1518, Apr. 1990.

[4] N. I. Cho and S. U. Lee, "Fast algorithm and implementation of 2-D discrete cosine transform," *IEEE Trans. Circuits Syst.*, vol. CAS-38, pp. 297-305, Mar. 1991.

[5] B. G. Lee, "A new algorithm to compute the discrete cosine transform," *IEEE Trans. Acust., Speech, Signal processing*, vol. ASSP-32, pp. 1243-1245, Dec. 1984.

[6] ISO/IEC JTC1/SC29/WG10, JPEG Committee Draft CD 10918, 1991.

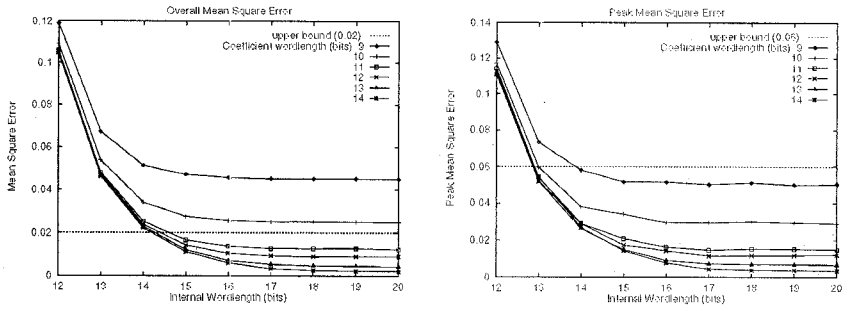
[7] N. Weste and K. Eshraghian, *Principle of CMOS VLSI Design, A Systems Prespective*, Addison-Wesley, 1992.



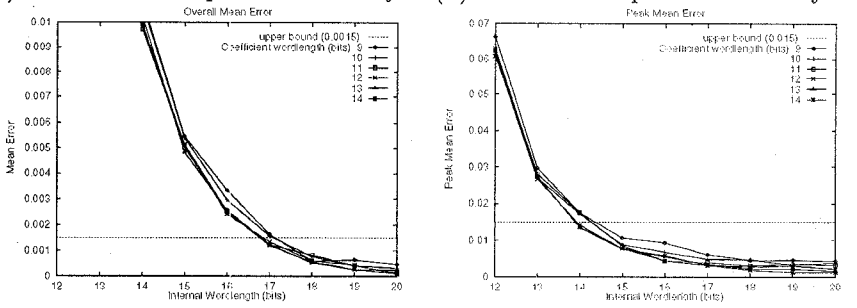
(1) $u_{n_1} = \frac{1}{8}\sum_{b=0}^{7} U_b W_{32}^{-(4n_1+1)b}, 0 \le n_1 \le 7.$

(2) The same as Fig. 2, but the direction of data flow is reversed.

Figure 4: Folded architecture for 8×8 2-D IDCT.



(a) Overall mean square error analysis (b) Peak mean square error analysis



(c) Overall mean error analysis (d) Peak mean square error analysis

Figure 5: Finite wordlength analysis