ENERGY SCALABILITY IN MULTIMEDIA CODEC USING NQDCT

Ming-Yan Chan and Chi-Wah Kok

Dept. Electrical Engineering, Hong Kong University of Science and Technology, HONG KONG

ABSTRACT

This paper discusses a novel algorithm that combines the DCT with the quantizer to achieve energy scalability in DCT based multimedia codec. A throughout analysis of the algorithm is presented. Simulation results showed that a 15% reduction in computational energy can be obtained in an implementation of MPEG coder over a wide range of bit-rate when compared to that of the standard MPEG coder test model.

1. INTRODUCTION

The multimedia codec system consumes a significant amount of energy in wireless multimedia applications. It would be desirable, in terms of energy conservation, to use a low complexity algorithm in multimedia coder that can obtain energy scalability. The energy scalability is similar to bitrate scalability that trades quality for a corresponding level of energy consumption will allow the application to configure the multimedia coder's energy spending budget according to the system needs. The amount of energy that can be spent on the multimedia codec depends on variables such as the user requirement, the multimedia content, and the state of the power supply. Since the quantization system in the multimedia codec system directly affects the quality of the system and at the same time consumes a significant percentage of the energy, which makes it an important target for enabling energy scalability.

As an example, the computational complex of MPEG-4 compressed video sequence at 30 frames per second (fps) and 720×480 resolution on an MMX-enable 233-MHz mobile Pentium is listed in Table 1. According to the specifications published by Intel, a Tillamook-class mobile Pentium MMX (1.8V, 233MHz) is capable of executing 932 16-bit media MOPS. The quantization process alone account for 14.1% of the machine cycles in MPEG-4 requires 128 megaoperations per second (MOPS) which accounts for less than 14% of the processor resources, and therefore real-time performance is more than feasible. Yet, the power dissipation of such a software implementation can be substantial. The mobile Pentium MMX (1.8V, 233MHz) exhibits 7nF of switched capacitance per machine cycle, which corresponds to 9mF of switched capacitance (1.3M machine cycles at 7nF per cycle), and a software applications supporting real-time video at 30fps will require 0.88W just for the quantization process.

Various strategy has been considered. The most noticeable technique is the approximated DCT algorithm [2]. Depends on the property of the encoding frames (whether it is a I, P or B frames), DCTs with different precision are used to encode the video frame, and thus achieve energy saving. However, there is a significant quality penalty associated with this approach that may not always be acceptable. In this work, we presented the design of an NQDCT. The NQDCT is an approximated DCT algorithm with the consideration of the quantizer. Noticed that the quality of the encoder depends on both the DCT precision and the quantizer being used, therefore, the NQDCT will be able to control the quality/energy tradeoff and achieve better quality than that of [2].

The NQDCT strategy for reducing the computation power was twofold. First, we select a DCT algorithm such that the quantizer can be embedded into it. [3] successfully merged DCT/Q to reduce the computational complexity. However QDCT has restriction on the scalar quantization process that can be merged with the DCT. The QDCT cannot be applied together with variable quantization step size. A new quantized DCT (NQDCT) is proposed in this paper to remedy the problem, such that the quantization process is merged with the DCT, where variable quantization step size can be applied in the proposed algorithm.

Second, we optimize the bit-width especially the bit width required in the computation of the NQDCT algorithm. Though appropriate scaling with the consideration of the quantization DCT coefficients, the NQDCT can be computed with shorter bit quantity with almost no loss in precision. Further gain in the reduction in computational complexity can be obtained by reducing the bit quantity to enable packing multiple data into a single SIMD instruction. Such a trade-off design will be presented in later section together with the error analysis of such implementation on INTEL processor using SSE2 instructions.

2. ALGORITHMIC DESIGN

The DCT of a $N \times 1$ real signal vector $\mathbf{X} = [x_0, x_1, \cdots, x_{N-1}]^T$ is defined as

$$y_k = \frac{c_k}{2} \sum_{n=0}^{N-1} x_n \cos \frac{(2n+1)k\pi}{2N}, \qquad k = 0, 1, \cdots N-1,$$

$$c_k = \begin{cases} \frac{1}{\sqrt{2}}, & k = 0, \\ 1, & \text{otherwise.} \end{cases}$$

In matrix vector representation,

$$\mathbf{Y} = \mathbf{C}\mathbf{X}^T, \tag{1}$$

with $\mathbf{Y} = [y_0, y_1, \cdots, y_{N-1}]^T$, and $[\mathbf{C}]_{i,j} = \cos\left(\frac{(2j+1)*i*\pi}{2N}\right)$ is the cosine transformation matrix. As an example, the cosine transformation matrix $\mathbf{C}$ for $N = 8$ is given by

$$\mathbf{C} = \begin{bmatrix} C_4 & C_4 & C_4 & C_4 & C_4 & C_4 & C_4 & C_4 \\ C_1 & C_3 & C_5 & C7 & -C_7 & -C_5 & -C_3 & -C_1 \\ C_2 & C_6 & -C_6 & -C_2 & -C_2 & -C_6 & C_6 & C_2 \\ C_3 & -C_7 & -C_1 & -C_5 & C_5 & C_1 & C_7 & -C_3 \\ C_4 & -C_4 & -C_4 & C_4 & C_4 & -C_4 & -C_4 & C_4 \\ C_5 & -C_1 & C_7 & C_3 & -C_3 & -C_7 & C_1 & -C_5 \\ C_6 & -C_2 & C_2 & -C_6 & -C_6 & C_2 & -C_2 & C_6 \\ C_7 & -C_5 & C_3 & -C_1 & C_1 & -C_3 & C_5 & -C_7 \end{bmatrix},$$

with $C_k = \cos\left(\frac{n*\pi}{16}\right)$. For image and video compression, these 1D DCT algorithms are applied to the codec using the Row-Column 2D DCT, which can be defined in a similar manner as

$$\mathbf{Y} = \mathbf{C}(\mathbf{C}\mathbf{X}^T)^T, \tag{2}$$

where $\mathbf{X}$, $\mathbf{Y} \in \mathbb{R}_{N \times N}$ are the input and output signal blocks. The scalar quantization of the DCT output $y_{i,j} = [\mathbf{Y}]_{i,j}$

$$y_{i,j} = \lfloor y_{i,j}/q_{i,j} \rfloor, \tag{3}$$

This work described in this paper has been supported by the Research Grants Council of Hong Kong, China (Project no. CERG HKUST6236/01E and DAG03/04.EG37).

Table 1. Computation loading of different MPEG-4 processes

	DCT/Q	NQDCT
DCT	14.5%	20.5%
Quantization	14.1%	0%
MC	23.5%	26.2%
ME	9.3%	10.4%
Others	38.6%	43%

Table 2. NQDCT coefficients for each quantizer q.

q	$C1$	$C2$	$C3$	$C4$	$C5$	$C6$	$C7$
1	0.4904	0.4619	0.4157	0.3536	0.2778	0.1913	0.0975
2	0.2452	0.2310	0.2079	0.1768	0.1389	0.0957	0.0488
3	0.1635	0.1540	0.1386	0.1179	0.0926	0.0638	0.0325
.	...	...	...	...	...	...	...
31	0.0158	0.0149	0.0134	0.0114	0.0090	0.0062	0.0031

where $q_{i,j} = [\mathbf{Q}]_{i,j}$ are the quantization step sizes for the DCT output at location (i, j). In MEPG-4 each $q_{i,j}$ must takes a value in $[1, 2, \cdots, 31]$, and each $q_{i,j}$ can takes a different values, which is known as variable scalar quantization. In general the quantization process can be defined as

$$\mathbf{Y}^q = \lfloor \mathbf{Y}./\mathbf{Q} \rfloor, \quad (4)$$

where $\mathbf{Q}$ is the quantization step size matrix, e.g. $\mathbf{Q} \in \mathbb{R}_{8\times 8}$ in MPEG-4 video coder, and $./$ represents the element to element division between the two matrices in the operand. Such transform and quantization processes are abbreviation as *DCT/Q*.

2.1. NQDCT

Consider the last stage of 1D DCT signal flow graph in Figure 1(a), which has the general form of

$$y_k = C_n a + C_m b, \quad (5)$$

where a and b are intermediate DCT results, and C_k is the twiddle factor at the last stage of the DCT signal flow graph for the output signal y_k. The DCT output signal y_k is quantized according to

$$y_k^q = (C_n * a + C_m * b)/q = \left(\frac{C_n}{q}\right) * a + \left(\frac{C_m}{q}\right) * b. \quad (6)$$

As a result, the quantizer can be embedded into the DCT algorithm by using a quantized twiddle factor at the last stage of the DCT signal flow graph. Noted that the quantized DCT coefficients at the last stage of the DCT signal flow graph will has the same values of those list in Table 1 for the MPEG-4 video coder.

There are several advantages to use NQDCT when compared to QDCT. Firstly, NQDCT solved the problem of using variable quantizaton matrix in MPEG-4 video coder associated with QDCT. The NQDCT computes the quantized DCT coefficients by first perform the row transformation using the normal DCT with the signal flow graph in Fig.1(a), and then perform the column transformation using the modified DCT with the signal flow graph in Fig.1(b). Since only the twiddle factors of the last stage of the DCT is required to be modified by the quantizer, as a result, a simple indexing scheme can be used to acquire the quantized DCT coefficients from Table 1, which in turn allows each DCT outputs to has its own quantizer step size and thus achieve higher compression quality. Secondly, since quantized coefficients are only used in last stage before the generation of the quantized DCT coefficients to be the input of the entropy coder in the video encoder. As a result, the NQDCT will not suffer from rounding error accumulation problem when compared to that of QDCT. Therefore better visual quality is obtained as observed in the simulation results.

3. BUS WIDTH OPTIMIZATION

The datapath width of the arithmetic units has been optimized to meet IEEE Standard 1180-1990 for DCT/IDCT precision by a comfortable margin but at the same time discard unnecessary precision bits that would increase the power dissipation. As an example, the pixels in MPEG-4 video are represented using 8-bit quantity. In the 8×8 block DCT, each one of the 64(2^6) elements contributes to every transformed coefficients, so at least 14-bit (8+6) precision is needed. Existed DCT implementation in Intel processor use 32-bit precision because registers are 32-bit long in Intel architecture. However, to get more performance gain from using SSE2 technology, shorter bit quantity is preferred (because one register could contain more individual processing data). As a result, the designers are facing a trade-off between the calculation error and the processing power.

Table 3. Scaled NQDCT Coefficients for Each Quantizer q.

q	C_1^f	C_2^f	C_3^f	C_4^f	C_5^f	C_6^f	C_7^f	r
1	251	213	142	50	237	98	181	128
2	125	105	71	25	117	49	90	64
3	83	71	47	16	79	32	60	43
...	...	...	...	...	...	...	...	...
31	8	7	5	2	8	3	6	4

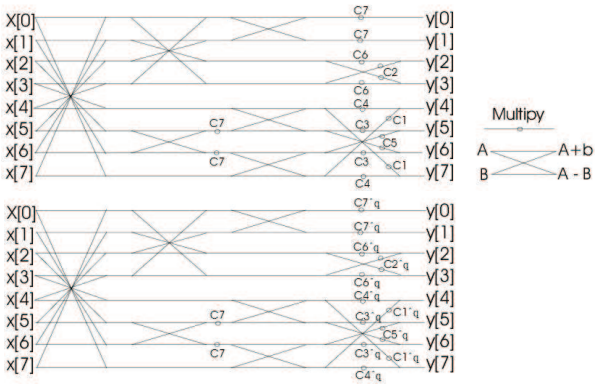

Fig. 1. (a) The 1D Chen DCT signal flow graph, and (b) the NQDCT signal flow graph. Noticed that the quantized DCT coefficients are indicated by C_i^q.

The NQDCT approach this trade-off problem by appropriate scaling similar to that used in traditional fixed-point FFT computation [6]. The only difference is the coefficients in the last stage of the NQDCT computation, where the scaled NQCT coefficients C_n^f are given by

$$C_n^f = \lfloor (C_n 2^n)/q \rfloor, \quad (7)$$

with n being the number of right shifts involved. The scaled NQDCT outputs are rounded to obtain the final output of the quantized DCT output. The rounding factor r is given by

$$r = 2^{(n-1)}/q, \quad (8)$$

which is the same as that in [6]. The scaled NQDCT coefficients (C_n^f and the rounder (r) for each q used in MPEG-4 video coder is listed in Table 3 for reference.

3.1. Error Analysis

The scaled NQDCT computation reduces computational complexity by using fixed-point arithmetic but at the same time, induced rounding and truncation error in the quantized DCT output.

3.2. Rounding Error

The rounding error is the result of scaled NQDCT output being not divisible by the rounder q. An example of the rounding error:

	Non-divisible coefficients
DCT/Q	$(251 * 5 + 213 * 5)/2 = 1160$
NQDCT	$(251/2 * 5 + 213/2 * 5) = 1155$

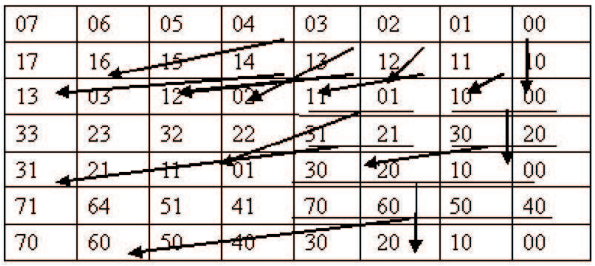

Fig. 2. Transposition signal flow implemented using SSE2

where a difference of 5 in magnitude is observed from the output of DCT/Q and QDCT, and 251 and 213 are the intermediate signal entering the last stage of the DCT process.

3.3. Truncation Error

The truncation error is the result of the scaled NQDCT output being down shifted and thus losses it precision. An example of truncation error:

DCT/Q	$1160 >> 8 = 4$
QDCT	$1155 + 128 >> 8 = 5$

Although the difference between scaled NQDCT and DCT/Q outputs are very large, the maximum difference is 1 after a right shift operation, and in most cases, zero differences are observed. Extensive simulations using random sequences and MPEG-4 test sequences revealed the following

Random data	$0-5$ unit difference per block
Testingsequence	$0-2$ unit difference per block

which showed that the difference between DCT/Q and NQDCT is very small. In deed, for MPEG-4 test sequence, 99.5% DCT/Q and NQDCT are identical. As a result, the encoded PSNR are almost the same for DCT/Q and NQDCT results, and hence no visible difference is observed.

4. SIMD MULTIMEDIA PROCESSOR IMPLEMENTATION

Further performance gain can be obtained by considering the implementation of the codec on SIMD machine. In this paper, the implementation of NQDCT on Intel processors with Intel SIMD extensions known as the SSE2 was considered. In SSE2 implementation, the NQDCT computational cost is not dominated by the number of multiplications and additions, rather it is dominated by the parallelism within the processing. Parallelism can be applied in two ways: 1) within a single block, and 2) between blocks. In MPEG-4 video coding, each iterations will store 6 blocks (4-luminous, 2-chrominous) into the cache before being processed. As a result, it is more convenient to adapt the first approach, as it requires less temporal register and memory. Further noticed that the NQDCT is appropriately scaled, such as to increase the amount of parallelism achievable for each register usage.

4.1. Parallelism within a single block

The overhead of this approach is that it requires two transpositions of a single block. Fig 2 shows a typical way using SSE2 instructions to transpose a row. The transposition is done by mainly using `punpcklwd`, `puncklqdq` and `punpcklqdq` instructions to intervene the data. A complete matrix transposition requires more instructions to perform. As a result, the CPU time spend on transposition become significant in SSE2 implementation.

Table 4. Number of each instruction type

	add	mul	shift	mem read	mem write
1D-DCT	28	16	8	8	8
8x8 Q	0	64	65	65	64
8x8 DCT/Q	448	320	193	193	192
8x8 NQDCT	448	256	128	136	128

The row column DCT computation is the reason of the 2 transpositions. The first transposition will first the `xmm` registers with 8 indices of each row. After the 1D NQDCT, another transposition to fill `xamm` with 8 indices of each column. We found that by changing the order of passing row and column can save one transposition from the tradition order

$$\mathbf{Y} = \mathbf{C}(\mathbf{C}\mathbf{X}^T)^T, \qquad \mathbf{X} = \mathbf{C}^{-1}(\mathbf{C}^{-1}\mathbf{Y}^T)^T, \tag{9}$$

to the proposed order

$$\mathbf{Y} = (\mathbf{C}\mathbf{X})\mathbf{C}^T, \qquad \mathbf{X} = (\mathbf{C}^{-1}\mathbf{Y})\mathbf{C}^{-T}. \tag{10}$$

It is easy to show that the about two computations will produce the same results. Although the proposed DCT computation order can reduce computational costs, changes are required to be made in the quantization matrix, zig-zag scanning order, etc.

To implement the new DCT computation order, the input data is first transposed and pointed by pointers such that the registers `xmm0`, `xmm1`, `xmm2`, and `xmm3` are filled with 8 same row elements $X[0]$, $X[1]$, $X[2]$, $X[3]$. Parallel additions and subtractions are operated for 8 elements. Intervene data in alternate order is used for multiply and add instruction `pmaddwd`. Since each register size is 128 bit wide, we can only perform 4 data multiplications in each instruction. As a result, 2 multiply and add instructions are required to compute 8 data output. The final NQDCT results are obtained by an addition with the corresponding rounder r for the specific quantizer following the 8 bit shift down operations.

The implementation can be further speeded up with extra memory. Since SSE2 only provides multiply and add instructions but no multiply and subtraction instruction is provided. As a result, when performing the butterfly computation $g \cdot (b_0 - b_1)$, two instructions are required, where one instruction is required for subtraction, and one for multiplication. On the other hand, if we make use of the extra memory to store the seven $-g$ in the memory and loaded into register. A single multiply and add instruction can be used to compute the above butterfly process.

5. ENERGY COMPARISONS

The computational energy consumed by the algorithm is compared in two ways. First, we will consider the energy saving resulted from algorithmic changes. The number of instruction required to implement a 8×8 DCT/Q and NQDCT are presented in Table 4, which clearly showed a significant amount of saving on the number of multiplications and shifts in NQDCT. Furthermore, the numbers of memory access (both read and write functions) are reduced. All those reduction in the instruction counts contributes significantly to the reduction in computation power of the NQDCT algorithm. Noticed that this evaluation does not consider the instructions incurred in the control functions of the actual implementation. To complement the algorithmic complexity comparison, the actual power consumption of an MPEG-4 video coder implemented with DCT/Q and NQDCT are presented in Figure 4 under various bit rates.

Fig. 3. Akyio and Funfair testing sequence.

5.1. Simulation Results

MPEG-4 video coder is implemented to demonstrate the performance of the proposed NQDCT. Video coding results are collected with MPEG-4 test sequence in 30 frames per second CIF (352×288, i.e. 44550kB/frame) video. The small video size is chosen, such that we can simulate a large number of encoding bit rates. Furthermore, Intel Pentinum 4 processor graded computer running at 1.8GHz is used to generate the simulation results. Such that the high performance system will allows us to compress the video in real time under various bit rates. Furthermore, both MPEG-4 coder implemented with DCT/Q and NQDCT algorithms are optimized with SIMD extension whenever possible for fair comparison. In particular, Fig.3 showed the screen capture of the decoded video of Akyio and Funfair compressed at 32kb/s and 127kb/s respectively. Noted that the Funfair video encoded at 127kb/s has an encoding rate of 25frames/sec in our system. Akyio and Funfair is chosen to report in this paper, because Akyio has a static background and little motion, while Funfair contains a lot of moving objects. As a result, the two test sequence represents a large variety of real world video. The graph in Fig.4(a) showing the proposed NQDCT algorithm can obtain similar PSNR when compared to that of DCT/Q. Actually, the PSNR difference when compared between NQDCT and DCT/Q is in the range of 0.01 to 0.5dB for Akyio and 0.09 to 2dB for funfair. At the same time, a large reduction in power can be achieved by NQDCT. Showing in Fig.4(b) is the estimated power of the MPEG-4 video coder implemented with NQDCT and DCT/Q (where power dissipated by memory storage and other external interface are not considered). It can be observed that NQDCT based algorithm requires 15% less power in average than that of DCT/Q based video coder.

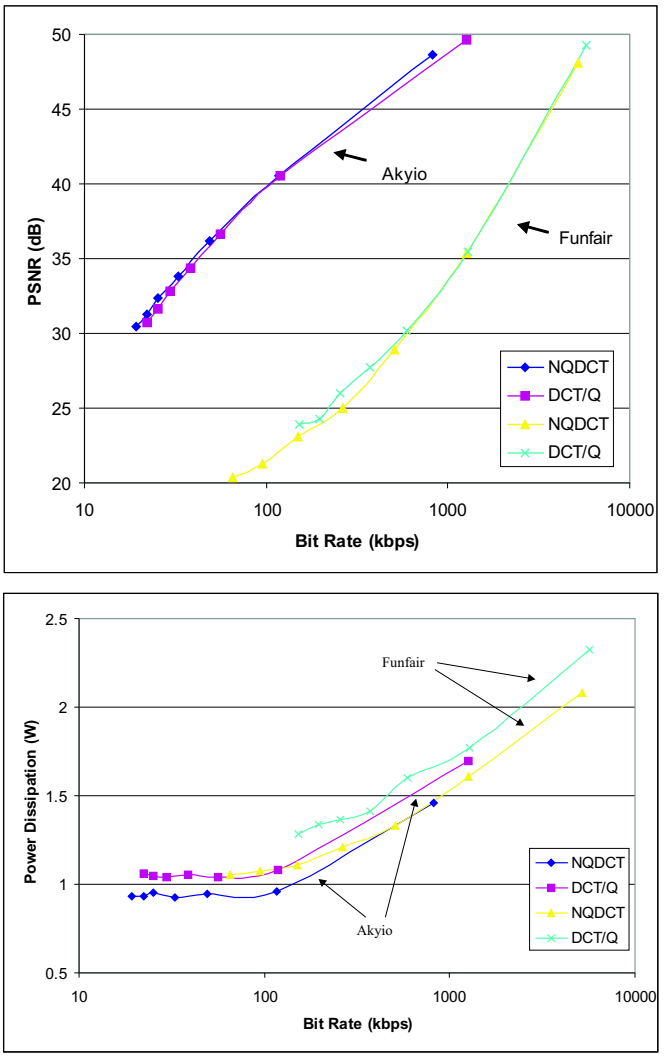

Fig. 4. (a) PSNR vs Bit rate with $q = 1, 5, 10, 15$, and (b) Encoding power dissipation (W/frame) vs Bit Rate with $q = 1, 5, 10, 15$.

6. CONCLUSION

A novel NQDCT algorithm is proposed where the quantization process is merged with the DCT. The proposed algorithm can be used with variable quantization step size. Furthermore, appropriate scaling has been applied to optimize the precision and buswidth to gain further reduction in computational complexity. Simulation results have shown that a 15% reduction in computation power has been obtained with the new algorithm in MPEG-4 video coding when compared to that of the standard MPEG-4 test model.

7. REFERENCES

[1] T.Xanthoponlos and A.P.Chandrakasan, "A lowpower IDCT macrocell for MPEG-2 MPML exploting data distribution properties for minimal activity," *IEEE Journal of Solid-State Circuits*," vol.34, no.5, May 1999, pp.693-701.

[2] R. Henning and C. Chakrabarti "The trading dct/idct quality for energy reduction in MPEG-2 video codecs"

[3] A.Docef, F.Kossentini, K.Nguyen, and I.R.Ismaeil, "The quantized DCT and its application to DCT-based video coding," *IEEE Trans. Image Processing,* Mar. 2002, pp.177-187.

[4] *Draft Recommendation H.263*, ITU Telecom, Standardization Sector Study Group 15. 1995.

[5] T.Sikora, "The MPEG-4 video standard verification model," *IEEE Trans. Circuits and Systems for Video Technology,* Feb. 1997, pp.19-31.

[6] P.Kabal and B.Sayar, "Performance of fixed-point FFT's: Rounding and scaling considerations," *Proc. ICASSP,* vol.11, April 1986, pp.221-224.

[7] J.Han, G.Ren and C.Han, "A novel fixed-point FFT algorithm on embedded digital signal processing systems," *Proc. WCCC-ICSP,* vol.1, Aug. 2000, pp.21-25.