LOW POWER DCT USING HIGHLY SCALABLE MULTIPLIERS

Ricardo Castellanos, Hari Kalva, and Ravi Shankar

Department of Computer Science and Engineering, Florida Atlantic University, Boca Raton, FL 33431

ABSTRACT

Low power consumption in computing systems is a key requirement for devices such as cell phones and cameras. In this paper we present a low power DCT implementation using a highly scalable multiplier. This paper focuses on IDCT with playback applications such as digital photo displays. The proposed solution exploits the fact that the size of the multiplications varies per stage in a multistage IDCT implementation and configuring multipliers to match the needs of each stage saves power. Results are compared with Wallace and Array multipliers. We show that using a scalable multiplier and dynamically reconfiguring the width of the multiplier leads to significant power savings (over 72%) with negligible degradation in decoded image quality.

1. INTRODUCTION

Many applications today use digital signal processing in a wide variety of areas due to the increasing advances in technology. Multimedia applications, for example, use algorithms to code video and images to reduce the large amount of storage and satisfy transmission requirements. DCT is widely used in image and video compression to de-correlate the signal and increase compression efficiency. DCT is a complex operation and uses significant amount of computing resources. Thus fast transforms such as the fast discrete cosine transform (DCT) are often used to meet realtime constraints [1],[2],[5].

Due to the use of multipliers, a significant amount of power and computations are required in image and video coding for direct and inverse transform operations (DCT/IDCT). A multiplication is computation intensive and utilizes large amounts of power. Over the years many fast algorithms have been proposed for the computation of the DCT focusing on different approaches (optimizing speed, throughput, latency, turnaround time), but the design for low power consumption was not an issue until the late eighties and it has become increasingly a hot topic as the demands for mobile computation power, portable devices and portable multimedia applications increase.

2. THE DISCRETE COSINE TRANSFORM

2.1. One Dimensional DCT-IDCT

A one dimensional N-point DCT of a given data X, is defined by:

$$\tilde{X}_k = E_k \sum_{n=0}^{N-1} X_n \cos\left(\frac{\pi(2n+1)k}{2N}\right),\ k = 0,1,\dots,N-1 \qquad (1)$$

The inverse DCT is given by:

$$X_k = E_k \sum_{n=0}^{N-1} \tilde{X}_n \cos\left(\frac{\pi(2n+1)k}{2N}\right),\ k = 0,1,\dots,N-1 \qquad (2)$$

Where $E_k = \begin{cases} \frac{1}{\sqrt{2}}, & k = 0 \\ 1, & k \neq 0 \end{cases}$

2.2 Transform Matrix

The 1-D DCT can also be expressed as matrix-vector product:

$$[F]_{Nx1} = [T]_{NxN}[X]_{Nx1} \qquad (3)$$

Where [T] is known as the transform matrix. The 1-D IDCT can be written as:

$$[X]_{Nx1} = [T]^{-1}_{NxN}[F]_{Nx1} \qquad (4)$$

The 2-D DCT is obtained by row-column decomposition. A row wise 1-D DCT is applied followed by the column wise 1-D DCT:

$$[F]_{NxN} = [T]_{NxN}[X]_{NxN}[T]^{T}_{NxN} \qquad (5)$$

The same operation is applied for the 2-D IDCT:

$$[X]_{NxN} = [T]^{T}_{NxN}[F]_{NxN}[T]_{NxN} \qquad (6)$$

3. MULTISTAGE REPRESENTATION OF DCT

Figure 1 shows the flow graph (butterfly diagram) for the DCT algorithm using 29 additions and 13 multiplications, with each stage corresponding to one single pass in the butterfly diagram and it is represented with one of the 4 different sparse matrices. Fast DCT implementations partition the 8x8 transform matrix into four different stages [3],[4],[6]. The transform matrix can then be expressed as the product of 4 sparse matrices:

$$[T]_{8x8} = [T_4]_{8x8}[T_3]_{8x8}[T_2]_{8x8}[T_1]_{8x8} \qquad (7)$$

Each single pass results in intermediate values of varying magnitude. The magnitude of these values depends on the input data size. For an n-bit wide input, the output after each stage is going to have a specific length. This bitwidth can be determined based on the transform matrix coefficients C_i. The values in the transform matrix are not shown due to space constraints.

Table 1 and table 2 show the bitwidth output of values in each stage for row wise and column wise operations for n-bit length input values. In this case, the input values are the mmost critical set, forcing the output to take the maximum or minimum values.

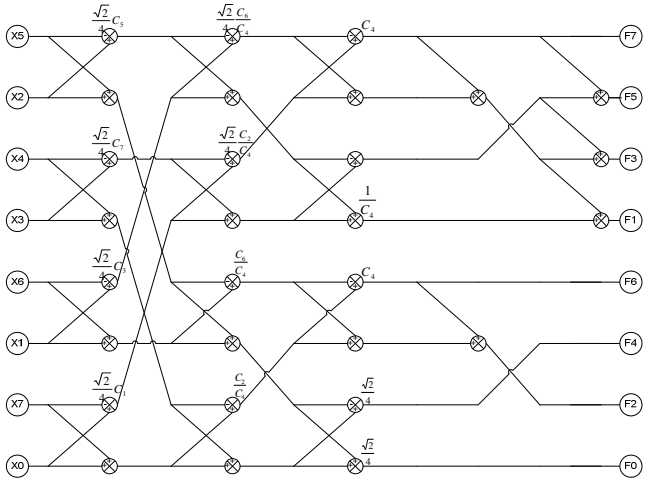


Figure 1. Flow graph of DCT

Table 1: DCT Bitwidth in each stage for row operations

Stage	Output	Bitwidth
1	[T1]**X**	n+1
2	[T2][T1]**X**	n+2
3	[T3][T2][T1]**X**	n+3
4	[T4][T3][T2][T1]**X**	n+2

Table 2: DCT Bitwidth in each stage for column operations

Stage	Output	Bitwidth
1	[T4][T3][T2][T1]**X**$[T1]^T$	n+3
2	[T4][T3][T2][T1]**X**$[T1]^T[T2]^T$	n+4
3	[T4][T3][T2][T1]**X**$[T1]^T[T2]^T[T3]^T$	n+3
4	[T4][T3][T2][T1]**X**$[T1]^T[T2]^T[T3]^T[T4]^T$	n+3

Based on the previous analysis, the output bitwidth for the 2-D inverse DCT can be determined the same way as the forward DCT. Tables 3 and 4 show the bitwidth of values in each stage for row wise and column wise operations.

Table 3: IDCT Bitwidth in each stage for row operations

Stage	Output	Bitwidth
1	$[T4]^T$**F**	n+2
2	$[T3]^T[T4]^T$**F**	n+3
3	$[T2]^T[T3]^T[T4]^T$**F**	n+3
4	$[T1]^T[T2]^T[T3]^T[T4]^T$**F**	n+2

Table 4: IDCT Bitwidth in each stage for column operations

Stage	Output	Bitwidth
1	$[T1]^T[T2]^T[T3]^T[T4]^T$**F**[T4]	n+4
2	$[T1]^T[T2]^T[T3]^T[T4]^T$**F**[T4][T3]	n+4
3	$[T1]^T[T2]^T[T3]^T[T4]^T$**F**[T4][T3][T2]	n+5
4	$[T1]^T[T2]^T[T3]^T[T4]^T$**F**[T4][T3][T2][T1]	n+3

From tables 1-4, it is clear that each stage results in numbers of different magnitude depending on the input. The magnitude of the coefficients in the input depends on the content being encoded and the quality at which it is encoded. Power consumed by multipliers depends on the width of the multipliers. Traditional multipliers such as Wallace and Array multipliers cannot be scaled dynamically and hence a 16 bit multiplier is used even with 8-bit operands. A low power multiplier based on operand truncation was proposed in [8]. Operand truncation reduces switching activity and hence reduces power consumption. The downside is that truncation leads to quality loss due to reduce precision. Power consumption can thus be reduced if different multipliers can be used for each stage. A Highly Scalable Multiplier (HSM) is well suited for these applications [8]. An HSM allows dynamic configuration of the multipliers for each block and each stage of IDCT.

4. TRANSFORM COEFFICIENT PRECISION

The DCT/IDCT implementations use floating point representation of the transform matrix at each stage. As the multiplier is scaled for each block/stage, the transform matrix coefficients should also be scaled down to match the multiplier size used. For example, if an 8-bit multiplier is used, an 8-bit floating point representation is used for the coefficients. This reduced precision of the transform coefficients affects the quality of the decoded images. The accuracy of a number in floating point representation depends on the bit length used and how many of those bits are used for the exponent and mantissa. A reduced precision floating point representation was determined by evaluating the configuration for mantissa that minimizes the loss in precision. The test was made for 8-bits, 10-bits, 12-bits and 14-bits of number length precision and 3, 4, 5 and 6 bits to represent the exponential part. For all different bit-length floating point precision of matrix coefficients, the best option for the minimum distance to the original value occurs when the exponent field is using 3-bits precision.

5. SIMULATION AND RESULTS

The performance of the proposed method was evaluated by simulating JPEG encoding/decoding on images: performing DCT, Quantization, Inverse Quantization, and then IDCT. The focus of the experiments was calculating the IDCT estimated power consumption, PSNR and SSIM on a set of images based on the use of scalable multipliers. A scalable multiplier adapts its size (bit length) according to the input resolution; therefore the transform matrix coefficients are truncated numbers that depend on the floating point resolution used by the multiplier. The IDCT algorithm use the product of 4 sparse matrices to represent the DCT transform matrix using 8x8 block size. The 2D-IDCT using the transform matrix is expressed as: $[X]_{NxN} = [T]^T_{NxN}[F]_{NxN}[T]_{NxN}$ thus, the 2D-IDCT using the 4 spare matrices is: $[T1]^T[T2]^T[T3]^T[T4]^T$**F**[T4][T3][T2][T1]. Each matrix implies a different number of multiplications as follows: [T1] and [T2] use 4 multiplications each one, [T3] uses 5 multiplications and [T4] use zero(0) multiplications; the total number of multiplications to perform the 2D-IDCT for each 8x8 block is 26.

Four different quantization modes were used in the simulation. First value is Q=1 which means no quantization, second and third value are Q=8 and Q=16

respectively, which mean the constant value used to quantize the DCT block and de-quantize the IDCT block and finally Q=M which means the Quantization Matrix defined by the JPEG standard.

Three different cases are considered to perform the IDCT algorithm explained before:

- Fixed Size Multiplier (FSM)
- Scalable Multiplier with Fixed Size for block (SFM)
- Scalable Multiplier with Variable Size for block (SVM)

The power consumption for every case is analyzed based on three different algorithms for multiplication: A Highly Scalable Multiplier (HSM) [8], Wallace Multiplier (WM) [8],[9],[10] and Array Multiplier (AM)[8],[9],[10]. Power consumption in CMOS digital circuits is proportional to switching activity in logic gates; the total number of gates that switch is used to calculate the approximate switching power consumption. In Table 5 we can see the Average Toggle Count (ATC) used to calculate the switching power with the three different multipliers. ATC for 10 and 14 bits was calculated using 12 and 16 bit multipliers with the 2 LSB set to zero. HSM scales better than WM and AM. Increase in switching activity of an HSM is proportional to the square of the rate at which bit-width increases (going from 16 to 32 bits increases bits by 2 and hence ATC (power consumption) by 4).

Table 5: ATC for Estimated Power Consumption

Bits	HSM	WM	AM
8	692	626	808
10	1097	1263	2041
12	1580	1819	2939
14	2167	2966	5626
16	2831	3874	7348
32	11449	19548	99102

5.1. Fixed Size Multiplier (FSM)

The FSM uses always the same size of bits to perform operations. Even if the input bitwidth is short, the FSM uses all of its resources as if multiplying larger numbers. The power consumption in this case is a number showing a considerable amount of energy because the use of circuitry not needed to perform the operations, but the output quality is always better because the use of full precision for the floating point representation. The power consumption will be proportional to the number of multiplications for each stage (26) and the ATC of the fixed multiplier (X) (Table 5): $PC \approx 26 \cdot ATC(X)$. Values for Power Consumption are shown in table 6.

Table 6: Power Consumption (ATC) for FSM

FSM	Power Consumption (ATC)		
Bit Precision(X)	HSM	WM	AM
8	17992	16276	21008
16	73606	100724	191048
32	297674	508248	2576652

5.2. Scalable Multiplier - Fixed Size for Block (SFM)

The SFM uses a scalable multiplier that adapts its size to the maximum length needed in the most critical stage, it means that for each 2D-IDCT block the multiplier has a fixed size for all stages (sparse matrices). For a specific input bitwidth "n" as the maximum precision number in the input block, the size of the multiplier will be "n+5". The power consumption is proportional to the number of multiplications for each stage (26) and the ATC determined by the scalable multiplier (X=n+5) (Table 5).

5.3 Scalable Multiplier - Variable Size for Block (SVM)

The SVM adapts its size for every single stage (sparse matrix) based on the input bitwidth of each stage. The first stage adapts the multiplier to the input block and then the next stage adapts its multiplier to the output of the previous block. The power consumption is proportional to the ATC sum of the multiplications for each stage considering the different multiplier's resolution used in every single stage.

A set of 5 Images with 512x512 resolution is processed to analyze the power consumption and quality for the FSM, SFM and SVM.

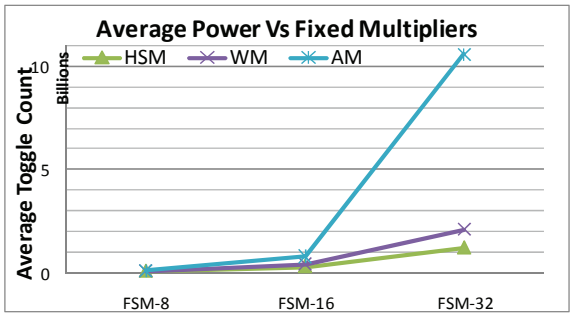
Baboon **Barbara** **Goldhill** **Lena** **Peppers**

Figure 2: Set of Images

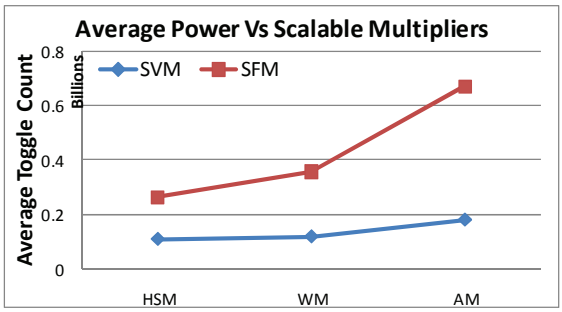
The power consumption is estimated calculating the number of times a specific multiplier is used (8-bits, 10-bits, 12-bits, 14-bits and 16-bits) when performing the 2D-IDCT. The approximate power consumption will be equivalent to the sum of the products of every single multiplier resolution multiplied by its corresponding ATC. Figure 3(a) shows the average power for FSM using 8, 16 and 32 bits. Figure 3(b) shows the average power for SVM and SFM.

The original input image is compared with the output image after the whole process (DCT-Quantization-Inverse Quantization-IDCT) to evaluate the quality using the PSNR and SSIM [7] algorithms. The performance of the HSM with fixed size, fixed-size per block, and variable-size per block is summarized in Tables 7-9. The results show that a significant amount of power has been reduced when the size of a multiplier is adjusted per block or per stage. The overhead in reconfiguring the multipliers is negligible. The results show that the power consumed by a HSM can be further reduced by changing the width of the multiplier for each stage of the IDCT. The reduced precision of floating point operations has minimal impact on the quality of decoded images. For video coding applications, such reduced precision causes a drift in the decoded video and distortion can accumulate for long GOPs. Video decoding

will be addressed as a follow up to this work. Image decoding using this approach can improve the battery life of portable devices used for image browsing.



(a)



(b)

Figure 3: Average Power for Fixed and Scalable Multipliers

Table 7: Performance of 16 and 32 bit HSM (Q=M)

HSM FSM 16/32	PSNR	SSIM	Power(million) 16/32-bits
Baboon	28.23	0.8807	301 / 1219
Barbara	32.89	0.9277	301 / 1219
Goldhill	33.58	0.895	301 / 1219
Lena	35.78	0.9181	301 / 1219
Peppers	34.84	0.8865	301 / 1219

Table 8: Performance of HSM with SFM (Q=M)

HSM SFM	8-bit	10-bits	12-bits	14-bits	16-bits	Power (million)	PSNR	SSIM
Baboon	0	104	8762	60814	36816	249	28.23	0.88
Barbara	0	338	3848	50726	51584	262	32.89	0.92
Goldhill	0	78	7722	42302	56394	263	33.58	0.89
Lena	182	3094	7852	42978	52390	257	35.79	0.91
Peppers	0	260	5382	31798	69056	273	34.84	0.88

Table 9: Performance of HSM with SVM (Q=M)

HSM SVM	8-bit	10-bits	12-bits	14-bits	Power (million)	PSNR	SSIM
Baboon	30508	66839	9149	0	108	28.3	0.87
Barbara	39534	56399	10563	0	105	32.9	0.92
Goldhill	41554	53750	11192	0	105	33.5	0.89
Lena	50762	44852	10882	0	101	35.6	0.91
Peppers	40260	51135	15101	0	107	34.7	0.88

Analyzing the power consumption in ATC, the Scalable Multiplier with Variable Size (SVM) reduces power consumption by 64% when compared with fixed 16-bit HSM and 91% when compared with fixed 32-bit HSM. Table 10 shows the reduction in power consumption for SVM and SFM compared with FSM-16 and FSM-32 using WM and AR.

Table 10: Power Consumption Reduction

	FSM-16		FSM-32	
	WM	AM	WM	AM
SVM(HSM)	72%	77.2%	94.5%	98.3%
SFM(HSM)	13.8%	14.3%	82.9%	93.6%

6. CONCLUSION

Low power DCT using a scalable multiplier was presented. Performance analysis for an IDCT implementation related to playback applications was reported. The proposed solution, dependant on scalable multiplier support in hardware, exploits the fact that the sizes of the multiplications vary per stage in a multistage IDCT implementation and configuring multipliers to match the needs of each stage saves power. The proposed solution affects quality because of lower precision floating point representation necessary. We show that this impact is negligible. The power consumed by HSM can be reduced by more than 72% when multistage implementation is used.

7. REFERENCES

[1] M. Kuhlmann, and K. K. Parhi. "Power Comparison of Flow-Graph and Distributed Arithmetic Based DCT Architectures" Conference Record of the Thirty-Second Asilomar Conference on Signals, Systems & Computers, Vol. 2, Nov 1998, Pages: 1214-1219.

[2] K. R. Rao and P. Yip, Discrete Cosine Transform: Algorithms, Advantages, Applications, Academic Press, 1990, ISBN 0-12-580203-X.

[3] X. Wan, Y. Wang, and W. H. Chen. "Dynamic range analysis for the implementation of fast transform," IEEE Transactions on Circuits ans Systems for Video Technology, Vol. 5, Issue 2, Apr 1995, Pages: 178-180.

[4] A. Artieri and O. Colavin, "A chipset core for image compression," IEEE Transactions on Consumer Electronics, Vol. 36, Issue 3, Aug. 1990, Pages: 395-402.

[5] E. Farag, and M. Elmasry. "Low-Power Implementation of Discrete Cosine Transform" Sixth Great Lakes Symposium on VLSI, Mar 1996, Pages: 174-177

[6] T.-S. Chang, C.-S. Kung, and C.-W. Jen. "A simple processor core design for DCT/IDCT," IEEE Transactions on Circuits and Systems for Video Technology, Vol. 10, Issue 3, Apr 2000, Pages: 439-447.

[7] Z. Wang, A Bovik, H Sheikh and E. Simoncelli. "Image Quality Assesment: From error visibility to structural similarity". IEEE Transactions on Image Processing, Vol 13, No 4, Apr 2004. Pages 600-612.

[8] Ajmera, A.M., Highly Scalable Multiplier, MSCE Thesis, Department of Computer Science and Engineering, FAU, December 2003.

[9] Kyungtae Han, Evans, B. L, Swanrtzlander, E. E. "Low power Multipliers with Data Wordlength Rreduction". Asilomar conference on Signals, Systems and Computers, 2005. Oct 28 – Nov 1, 2005. Pages 1615-1619.

[10] Hans Meier, P.C., Analysis and Design of Low Power Digital Multipliers, Electrical and Computer Engineering Dissertation, Carnegie Mellon University, August, 1999.