

LOW POWER DCT IMPLEMENTATION APPROACH FOR VLSI DSP PROCESSORS

S. Masupe and T. Arslan

The University of Edinburgh
Department of Electronics and Electrical Engineering
Mayfield Rd., Edinburgh EH9 3JL, United Kingdom

ABSTRACT

This paper presents an algorithm for the low power implementation of the Discrete Cosine Transform on Single multiplier CMOS DSPs. The algorithm reduces power by a combination of using shift operations, where possible, and manipulating bit-correlation between successive cosine coefficients applied to the input of the multiplier section such that the effective switched capacitance is reduced. This reduces the switching activity in the multiplication of a Discrete Cosine Transform processor. The paper describes the algorithm, the evaluation procedure and presents results with a number of example images illustrating upto 50% power savings.

1. INTRODUCTION

Currently there is considerable interest in the low power implementation of the Discrete Cosine Transform (DCT). This is mainly due to the DCT being the computational bottleneck of standards such as JPEG and MPEG [1]. Most research work considering low power implementation of the DCT have targeted reducing the computational complexity of the design or modifying it for operation under a lower supply voltage [1, 2]. Both these techniques have a limited effect on power reduction. Another major contribution to power consumption is due to the effective switched capacitance [3, 4]. Only a few researchers have targeted reducing power of a DCT implementation through a reduction in the amount of switched capacitance. This reduction has been achieved through techniques such as the detection of zero-valued DCT coefficients and lookup table partitioning [5]. This paper presents a technique for reducing power dissipation of the DCT by targeting the multiplier section of a DCT processor. The

pivot of this technique is a multiplication algorithm for the low power implementation of the DCT on CMOS based signal processing systems. The algorithm reduces power consumption by reducing the effective switched capacitance of the multiplier through effective manipulation of the multiplication process between the cosine and data matrices. Our results indicate that the effective capacitance can be reduced significantly by performing the multiplications of the DCT in an order dictated by the amount of correlation between subsequent cosine matrix elements applied to one of the inputs of the multiplier section. In addition we propose a modified processor architecture that accommodates this multiplication scheme and is open to exploitation by other ordering algorithms. We demonstrate our scheme by an example of ordering the cosine coefficients according to minimum hamming distance, revealing up to 50% power savings with a number of image examples.

2. IMPLEMENTATION

The computational bottleneck for the implementation of the DCT is the multiplication of the cosine coefficients matrix $[E]$, by the pixel matrix $[D]$, in order to obtain the DCT coefficients $[C]$, i.e.

$$[C] = [E] * [D] \quad (1)$$

Where each element C_{ij} in $[C]$ matrix of order n is given by:

$$C_{ij} = \sum_{k=1}^n E_{ik} D_{kj} \quad (2)$$

Traditionally, the multiplication process is performed in a row-by-column fashion [6, 7]. See Equation(2). In

some applications where the number of multipliers is limited, this implies new data at both multiplier inputs. This in turn implies a large switching activity inside the multiplier circuit.

We carried out a number of experiments with various cosine and pixel matrices. Careful analysis of the multiplication procedure revealed two distinct categories of cosine elements: (1) elements that are powers of 2, and (2) those which are non-powers of 2 numerics. For this reason, our algorithm processes the elements in (1) by performing a simple shift operation. It is well known that the switched capacitance of a shift is significantly less than that of a multiplication [2].

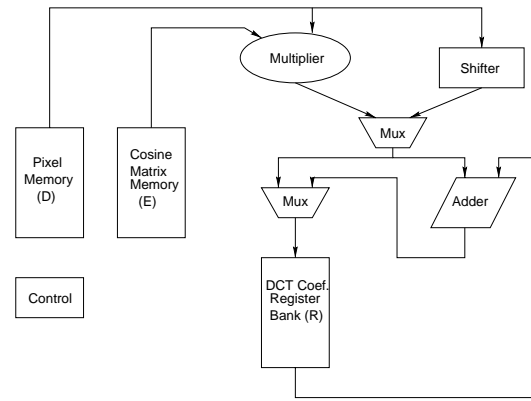


Figure 2: Simplified architecture of the processor

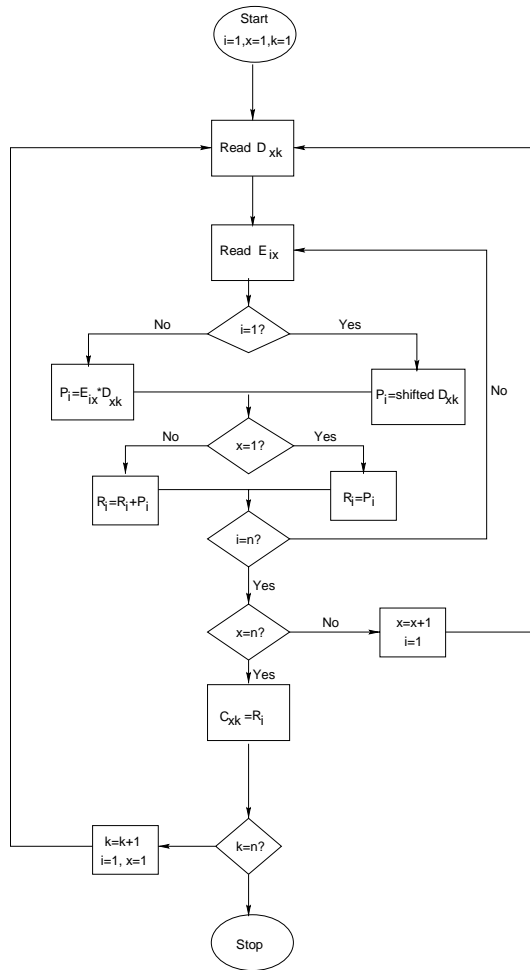


Figure 1: Flowchart of the algorithm

For each column being processed, the multiplication procedure outlined above has the product form (m *constant), where m are the cosine matrix elements

in the column being processed. The *constant* remains valid for only these m , if the column changes, a new constant is activated. The multiplication results are unchanged irrespective of the order in which the constant is multiplied by the sequence of cosine coefficients in the column. For this reason, our algorithm performs the multiplication of elements in (2) in a column-by-column fashion and orders the elements, in the column being processed, according to some criteria. In our case the criteria being minimum switching activity (hamming distance) between consecutive coefficients multiplied by the constant. This guarantees that similar elements are subsequently applied to the inputs of the multiplier causing minimum switching activity within the internal circuit of the multiplier. Although, our investigations were carried out using the example of ordering elements according to minimum hamming distance, in practice, any ordering algorithm can be used and *the amount of power saving is determined by the power of the algorithm used*. The steps in Figure 1 outline the algorithm, which commences with the following initialisation steps: (1) Process entries E_{1x} with shift operation. (2) Order remaining coefficients according to some ordering algorithm. (3) Save entries E_{ix} in (E) memory, see Figure 2, with elements of the same column being adjacent to each other, followed by the next column, and so on

To illustrate the above algorithm, consider an example where;

$$\mathbf{E} = \begin{pmatrix} 2.00 & 2.00 & 2.00 & 2.00 \\ 1.85 & 0.77 & -0.77 & -1.85 \\ 1.41 & -1.41 & -1.41 & 1.41 \\ 0.77 & -1.85 & 1.85 & -0.77 \end{pmatrix}$$

and

$$\mathbf{D} = \begin{pmatrix} 35 & 32 & 31 & 34 \\ 32 & 37 & 31 & 31 \\ 29 & 28 & 27 & 31 \\ 26 & 28 & 31 & 34 \end{pmatrix}$$

After the first iteration ($i=1$, $x=1$, and $k=1$), the first column, E_{i1} , is ordered according to minimum hamming distance to produce the ordered sequence (2.00, 0.77, 1.41, 1.85). The original locations of these elements are stored. Next the above sequence is multiplied by the first entry in D_{x1} , 35. The entries in registers R_i will be (70, 64.75, 49.35, 26.95), where the first entry (70) is saved in R_1 and the second entry is saved in R_2 and so on. Similarly at the end of iteration 2 ($i=2$, $x=1$, and $k=1$), R_i will contain (134, 89.39, 4.23, -32.25). The last iteration ($i=4$, $x=1$, and $k=1$) for this particular column, results in R_i containing the first column of the DCT coefficient matrix [C], i.e. $R_i = C_{x1} = (244, 18.96, 0, 1.35)^T$. This procedure is carried out until $x=k=n$, at which case R_i will contain C_{44} .

Although our algorithm can be implemented on traditional DSPs, for high throughput applications, a modified processor architecture is required. The architecture, a simplified version of which is shown in Figure 2, requires an internal register bank in order to store the partial products, (R), which eventually result in the DCT coefficients, [C]. In addition, a special memory unit is allocated for both the cosine and the pixel matrix elements, E_{ix} and D_{xk} respectively. A shifter is included to cope with the additional shift operations. The multiplier and the adder units are included to perform the normal multiply-add DSP operations. Since the outputs of both the multiplier and the shifter have to share the same input of the register bank, a multiplexer is needed to resolve which one output can use the register bank input. Another multiplexer is required to handle outputs from multiplier/shifter and adder units since some of the inputs to the register bank proceed directly to the bank without passing through the adder.

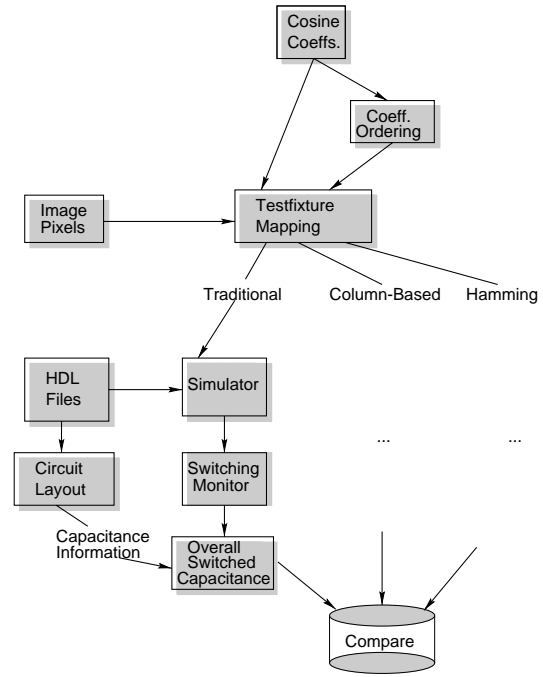


Figure 3: Framework for algorithm evaluation

3. SIMULATION AND RESULTS

The cosine coefficient matrix used was obtained using the MATLAB signal processing toolbox. This was scaled such that entries can be represented by numbers between -128 and 127. The pixels were chosen to represent several types of scenarios including checked and striped images. An 8x8-bit array Multiplier was constructed, using the Cadence VLSI suite with ES2 0.7 CMOS Technology, and processed down to layout level. A C-program based *test-fixture* mapping system was developed to generate input simulation files for the Verilog-XLTM digital simulator [8]. This involved forming the appropriate image-pixel/cosine-coefficient pairs, in the order imposed by the multiplication algorithm, so that they can be applied to the inputs of the multiplier hardware. Figure 3 illustrates the framework for the evaluation of the algorithm. Three types of input simulation files are generated, representing the use of one of the following: (1) Traditional cosine DCT multiplication (Traditional). (2) Column-based multiplication algorithm without ordering (Column-based). (3) Column-based multiplication algorithm with ordering according to minimum hamming distance (Ham-

Image	Ordering	Switched Capacitance (pF) *10 ³	Switched Capacitance reduction (%)
Horizontal Stripes	Traditional	13.77	-
	Column-Based	8.45	38.64
	Hamming	7.06	48.64
Vertical Stripes	Traditional	7.70	-
	Column-Based	7.55	1.97
	Hamming	6.41	16.84
Blocks	Traditional	9.25	-
	Column-Based	7.65	17.27
	Hamming	6.47	29.99
Checked	Traditional	13.81	-
	Column-Based	8.39	39.25
	Hamming	6.83	50.50

Table 1: Typical power savings

ming). In each simulation, the number of signal transitions (switching activity), at the output of each gate, is monitored. Capacitive information for each gate is extracted from the layout of the multiplier circuit. Both of these are used to obtain a figure for the total switched capacitance of the multiplier.

Table 1 illustrates the results obtained with the different scenarios used to test the algorithm. As the table shows, power savings are achieved with all examples, with a maximum of 50.5% power saving for an 8-bit multiplier.

4. CONCLUSIONS

An algorithm for the low power implementation of the Discrete Cosine Transform on CMOS based DSPs has been presented. The algorithm reduces power by manipulating data at multiplier inputs such that switching activity is reduced. This in turn has the effect of reducing the total switched capacitance. It has been shown that the algorithm can provide up to 50% power saving, when used in conjunction with an ordering algorithm which manipulates bit correlation of data and coefficient values at multiplier inputs, and that it provides the potential for more savings in power.

5. REFERENCES

[1] I. Pao and M. Sun, "Computation reduction for discrete cosine transform," in *International Sym-*

posium on Circuits and Systems, vol. 4, pp. 285 – 288, IEEE, 31 May - 3 June 1998.

- [2] L. Chen, J. Jiu, H. Chang, Y. Lee, and C. Ku, "Low power 2D DCT chip design for wireless multimedia terminals," in *International Symposium on Circuits and Systems*, vol. 4, pp. 41 – 44, IEEE, 31 May - 3 June 1998.
- [3] A. P. Chandrakasan and R. W. Brodeserson, *Low Power Digital CMOS Design*. Kluwer Academic Publishers, 1995.
- [4] A. T. Erdogan and T. Arslan, "Data block processing for low power implementation on single multiplier CMOS DSP processors," in *International Symposium on Circuits and Systems*, vol. 5, pp. 441 – 444, IEEE, 31 May - 3 June 1998.
- [5] S. Cho, T. Xanthopoulos, and A. P. Chandrakasan, "Ultra low power variable length decoder for MPEG-2 exploiting codeword distribution," in *Proceedings of the 1998 IEEE Custom Integrated Circuits Conference*, vol. 4, pp. 177 – 180, IEEE, May 1998.
- [6] W. Chen, *The Circuits and Filters Handbook*. CRC Press Inc., 1995.
- [7] V. Bhaskaran and K. Konstantinedes, *Image and Video Compression Standards: Algorithms and Architectures*. Kluwer Academic Publishers, 1995.
- [8] D. E. Thomas and P. R. Morby, *The Verilog Hardware Description Language*. Kluwer Academic Publishers, 1995.