Parallel Implementation of 2D-Discrete Cosine Transform Using EPLDs

D.V.R. Murthy, S. Ramachandran and S. Srinivasan
Department of Electrical Engineering
Indian Institute of Technology, Chennai - 600 036

Abstract

A novel implementation of Two Dimensional Discrete Cosine Transform (2D-DCT) using Embedded Programmable Logic Devices (EPLDs) has been proposed in this paper. The key feature of this scheme is that it's architecture is regular, linear, pipelined and it fits into just four numbers of commercially available EPLDs. It is capable of processing images of size 512 X 512 pixels at rates of 25 frames per second. The chip set offers device independent design and can be used in conjunction with other processors. The algorithm implemented can be easily modified and remapped as per needs with a minimum of effort since the architecture is realized using modular Hardware Description Language (HDL). The hardware complexity and accuracy of the proposed DCT processor compare favourably with those of other known implementation techniques.

1. Introduction

Most video standards such as HDTV video coding, JPEG and MPEG use Two Dimensional Discrete Cosine Transform (2D-DCT) as a transform coding scheme for effective image compression. The DCT is, however, computationally very intensive. To realize high speed and cost effective DCT computation for video coding, one needs an efficient and a VLSI implementation. In the recent past, many attempts have been made at efficient mapping of DCT algorithms into VLSI chips.

Cho and Lee [1] have implemented 2D-DCT using only six bit precision for cos coefficients. With the increase of precision, the processing speed decreases drastically. Direct mapping of Fast Cosine Transform algorithm [4] on an SIMD is presented in [2]. It uses a large number of switching elements and lacks in terms of modularity and regularity. In the architecture proposed by Sun-Chen et.al. [3], all multipliers are replaced by a large number of ROMs (more than forty of size 1k x10 bits) and emphasises a very stringent utilization of VLSI technology for design of ROMs to meet the required speed criteria.

A linear, pipelined, parallel architecture has been proposed in this present work for the implementation of 2D-DCT. The scheme proposes a modular DCT processor and is implemented with EPLDs. Since it has a pipelined architecture, the subsequent processes (Quantization, Huffman coding etc.) do not require extra time for computation as they can also be pipelined. While Huffman coding for n' th block (8X8 pixels) is being processed, one can process DCT and Quantization for (n+1)' th block in parallel. Therefore, there is a considerable reduction in the overall processing time. Furthermore, this architecture, while it is modular and regular, suffers from none of the limitations cited in the above references.

Section 2 proposes a new algorithm for parallel multiplication of 2D-DCT matrix, Section 3, an architecture for realizing the same, while Section 4 deals with its implementation using EPLDs. Results and comparisons made with other implementations are discussed in Section 5 and conclusions derived in Sec. 6.

2. Algorithm for Parallel Matrix Multiplication

DCT is an orthogonal transform consisting of a set of vectors that are sampled cosine functions [4]. 2D-DCT of a block of size 8 x 8 pixels is defined as

$$Z = C \, X \, C^T \qquad (2.1)$$

where X is the input matrix, C, the cosine matrix and C^T, its transpose. A direct implementation of the DCT is computationally intensive and requires butterfly structures to reduce the number of computations. The butterfly approach often results in irregular architecture, complicated routing and long design time. In order to achieve a regular and efficient method, a parallel matrix multiplication is proposed in the following section.

2.1. Decomposition and Parallelization

Eq. 2.1 can be implemented by parallel architecture wherein partial products (column vectors of matrix XC^T)

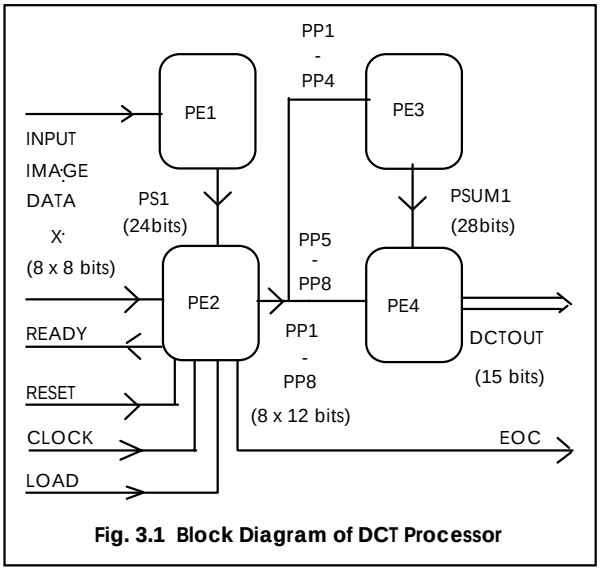

Fig. 3.1 Block Diagram of DCT Processor

generated in the first stage are fed to the second stage in a pipeline. After eight, first stage partial products are generated (in 8 clock cycles) and stored in intermediate registers, the DCT coefficient is generated (one every clock cycle after the eigth clock pulse) by multiplying row vectors of C and column vectors of XC^T matrix. While computing the '(i + 1) th' partial products of XC^T, 'i th' DCT coefficient can be computed simultaneously since 'i th' partial products of XC^T are already available. Thus the 64 DCT coefficient outputs are computed in 72 clock cycles after a latency of 8 clock cycles.

3. Architecture

The basic architecture of the proposed DCT processor is as shown in Fig. 3.1. It consists of four processing elements, PE1 and PE2 constituting the first stage of DCT while PE3 and PE4, the second stage.

The incoming image data (row vectors of X) is stored in PE1 and PE2 and can be loaded row by row by the control signal LOAD only after ascertaining that READY is set each time the row is stored. After the first eight CLOCK cycles, the partial products, PP1 - PP8 are generated and transferred to input latches of PE3 and PE4 on every 8th clock cycle. Corresponding cos coefficients (C) are multiplied with the partial products during the next clock cycle to generate the DCT output. While this DCT output computation is under progress, computation of the next set of first stage partial products can go on concurrently. Once the pipeline is full, for every clock cycle, a DCT

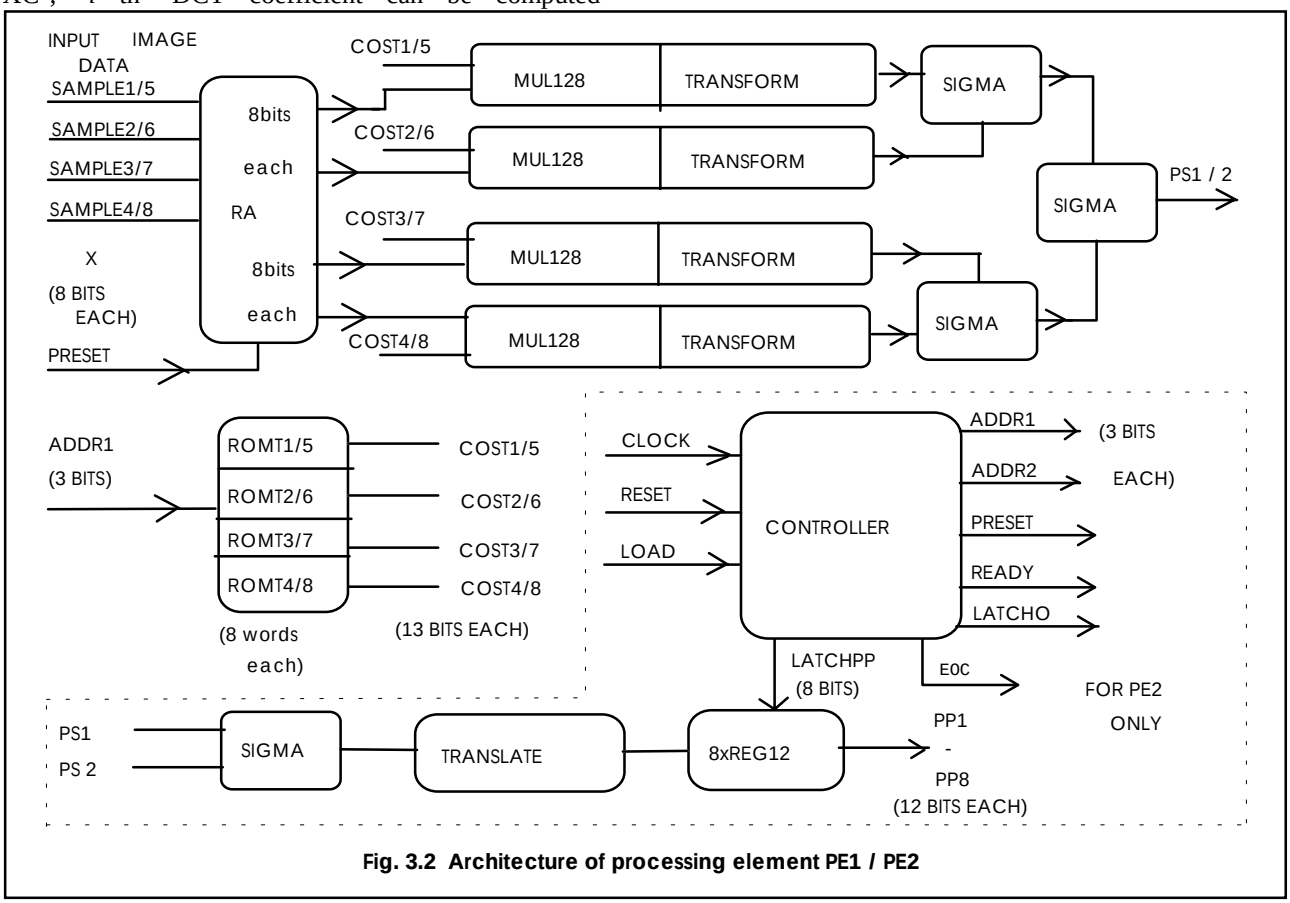

Fig. 3.2 Architecture of processing element PE1 / PE2

coefficient is generated. Thus, computation of DCT of an 8x8 image block requires a total of 72 clock cycles. The normalizing factors are absorbed within the cos coefficients and hence no extra scaling is required. The 64 coefficients of the DCT outputs are available at DCTOUT, one at a time, and can be latched at the rising edge of EOC output signal.

The DCT processor can be cleared by the external signal RESET.

3.1. Description of PE1 / PE2

As depicted in Fig. 3.2, PE1/PE2 consists of a 4 x 8 bit Register Array (RA), four 8x12 unsigned parallel multipliers (MUL128) for multiplying input image data (X) and cos coefficients (C^T), four two's complement, converters (TRANSFORM) and three adders (SIGMA). Four 8x13 bit ROMs are used to store cos coefficients (C^T). These ROMs are accessed by the Controller using ADDR1 as address. Transform components convert the products into 2's complement representation to take care of sign during addition. The partial sum (PS1) generated by the PE1 is fed to PE2 where the TRANSLATE converts two's complement number back to sign-magnitude number. The generated partial products are truncated and stored in eight, 12-bit registers, REG12, in sign-magnitude form. With every 8th clock pulse, the stored partial products are transferred to the second stage.

The Controller generates different control signals as shown in Fig. 3.2. These are used to latch the incoming data (PRESET), storing the data in pipeline registers (LATCHPP, LATCHO), clearing the data, etc. It basically consists of a program counter and a control ROM of size 72 x 18 bits. The program counter is a 72-state counter that increments on the rising edge of CLOCK and generates address to the control ROM where the control signals are stored. The Program Counter is reset after transmitting all the 64 DCT coefficients when the DCT processor will be ready to accept the next 8x8 block of image data.

3.2. Description of PE3 / PE4

As shown in Fig. 3.3, the Processing Element PE3 / PE4 consists of four 12x11, unsigned, parallel multipliers, MUL1211 to multiply cos coefficients (C) and PP1 - PP8, four two's complement converters, TRANSF and three adders, PLUS. Four on-chip, 8x13 bit ROMs are used to store the cos coefficients. These ROMs are addressed by ADDR2 generated by the Controller Unit. TRANSL converts two's complement number back to sign-magnitude number. The partial sum, PSUM1 from PE3 is fed to PE4, where the DCT coefficient is generated at the 15 bit output, DCTOUT.

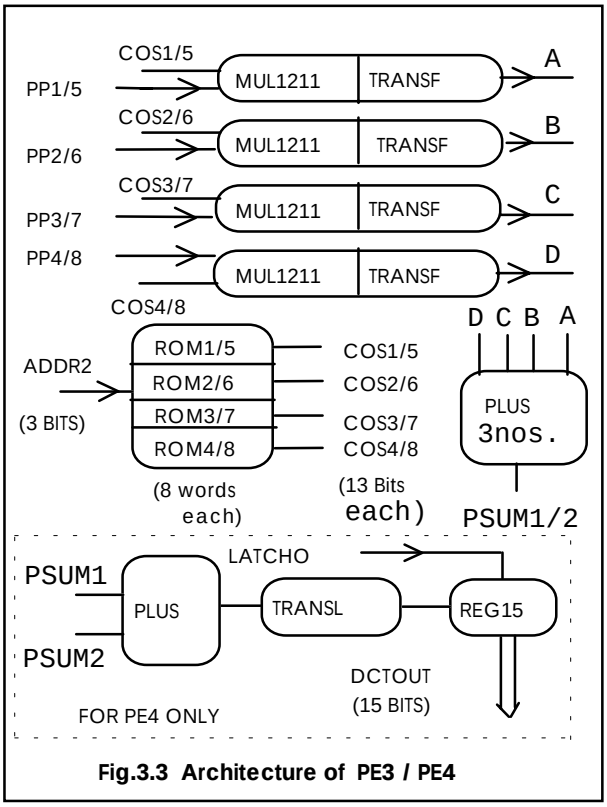

Fig.3.3 Architecture of PE3 / PE4

3.3. Sequence of Operations

Timing of different signals and events mentioned in figures 3.2 and 3.3 are as follows. The host processor and the DCT processor sequences run concurrently.

For the Host Processor:

1. Set the pointer to point to the first row of matrix X.
2. The host processor writes the row input image data (as pointed by the pointer) to the DCT processor.
3. If READY is set, assert LOAD to start/continue the DCT operation. Otherwise repeat step 3.
4. Increment pointer by one. If pointer >8, then reset the pointer to point to the first row again. Go to 2.

Note: The host processor must keep track of details regarding the image frame, the block number etc.

For the DCT Processor:

1. Set n = 1 (Clock pulse count).
2. Set READY to accept input row data from the host.
3. If LOAD (internally latched) is reset, repeat step3.
4. Otherwise (LOAD set), reset READY and compute the partial product, PP 'n' (in column of XC^T) in one clock cycle, applicable for n = 1 to 64. In parallel, compute DCT (n-8) coefficient (row of C * column of XC^T), applicable for n=9 to 72.
5. Reset LOAD and increment 'n' by one. If n=73, then set n=1.
6. Go to 2.

4. Implementation

The DCT Processor has been implemented using four numbers of Altera's FLEX 10k EPLDs [5]. Each module is described in VHDL, the size of coding being 1500 lines approximately. The utilisation of logic modules in each chip is about 30,000 logic gates (65 to 75% chip utility), thus providing enough room for incorporation of additional features.

5. Results and Discussions

The individual components were synthesized, functional testing and timing analysis carried out. The proposed architecture was tested with a 135nS clock, based on the worst case delay paths, and the circuit comprising four EPLDs is found to be working satisfactorily with the specified clock. As mentioned towards the end of Sec. 3, DCT computation for an 8X8 pixel block requires 72 clock cycles, that is, 9720nS (72X135nS). Therefore, for a 512X512 pixel image frame (4096 blocks), the processing time will be approximately 40mS (4096X9720nS). Hence, it is capable of processing images of size 512 X 512 pixels at the rate of 25 frames per second.

5.1. Comparison of DCT Processor Speed with Other Implementations

Type of implementations	Execution time in μS
C Pgm. using PC-AT (586, 120 MHz)	153.0
TMS 320C50 (20 MHZ) [6]	98.6
Proposed DCT processor	9.7

Table 5.1. Comparison of Execution Times of Different DCT Implementations for an 8x8 Image Block

Type of Processors	Execution time in μS
Proposed DCT processor	9.7
SIMD processor [2]	8.2
ROM Multiplier based DCT Chip [3]	5.2

Table 5.2. Comparison of Execution Times of Different DCT Processors for an 8x8 Image Block

The execution speed of the DCT processor is compared, against algorithmic implementation of DCT, with a PC-AT and a standard DSP processor as presented in Table 5.1and with two parallel architectures as given in Table 5.2. It is found that the speed of the proposed DCT processor is way ahead of the former and comparable with the latter architectures.

6. Conclusions

From the results obtained and comparisons made with the existing architectures, it can be concluded that the implementation of DCT processor using EPLDs may be an easy and cost effective solution meeting the real time requirements. The cos coefficients represented with 12 bits resolution meet the desired accuracy for the purposes of image reconstruction with negligible distortion. The speed of the pipelined architecture can be improved further by reducing the word length, and optimizing the Controller. This architecture also reduces the time taken by other processing stages like Quantization, Huffman Coding etc. so that the effective overall processing speed is much more than what is specified. The entire finite precision arithmetic performed by the DCT processor is simulated by C coded programs and verified.

In order to take full advantage of the proposed, parallel, pipelined architecture, all other functionality mapping, namely Quantization and Huffman Coding have to be integrated with the well structured DCT so that a total VLSI solution is realized. Work in this direction is currently under progress. The speed can be improved further by deploying fast algorithms and additional pipelining without sacrificing the concept of this novel architecture.

References

1. N.I. Cho, S.U. Lee, "Fast Algorithm and Implementation of 2D-Discrete Cosine Transform", IEEE transactions on Circuits and Systems, Vol. CAS - 38, pp 297 - 305, Mar 95.
2. C. M. Wu and A. Chiou , "A SIMD Systolic Architecture and VLSI chip for the Two-Dimensional DCT and IDCT", IEEE trans.on Consumer Electronics, Vol. 39, No. 4, pp 859 - 869, Nov. 1993.
3. M.T. Sun, T.C. Chen, and A.M. Gottlieb, "VLSI implementation of a 16x16 Discrete Cosine Transform", IEEE transactions on Circuits and Systems, vol. 36, pp 610 - 617, April 1989.
4. Rao and P. Yip, "Discrete Cosine Transforms: Algorithms, Advantages, and Applications", New York Academic, 1990.
5. ALTERA MAX + PLUS II manuals and ALTERA FLEX 10K Data Sheets, Ver. - 1, 1995.
6. S. Venkatesh and S. Srinivasan, "An efficient implementation of a progressive image transmission system using successive pruning algorithm on a parallel architecture", accepted for publication by HIPC 98.