A 2-DIMENSIONAL 16 POINT DISCRETE COSINE TRANSFORM CHIP FOR REAL TIME VIDEO APPLICATIONS

I. Defilippis, U. Sjöström, M. Ansorge, F. Pellandini

Institut de Microtechnique, Université de Neuchâtel, Rue A.-L. Breguet 2, CH-2000 NEUCHATEL, Switzerland

RESUME

Cet article présente une décomposition particulière de l'algorithme de la Transformée en Cosinus Numérique (TCN) permettant une implantation VLSI régulière et modulaire. Cette décomposition est ensuite appliquée à une TCN bidimensionnelle de 16 x 16 points, dont la réalisation sous la forme d'un circuit intégré spécifique (ASIC) est performante du point de vue de la puissance de calcul et se prête au traitement d'images vidéo en temps réel.

SUMMARY

This paper presents a particular decomposition of the Discrete Cosine Transform algorithm (DCT), allowing regular and modular VLSI implementations. This decomposition is then used for the realization of a 2-Dimensional 16 x 16 point DCT implemented on an Application Specific Integrated Circuit (ASIC) with high through-put performances suitable for real time video applications.

1. Introduction

The use of the *Discrete Cosine Transform* (DCT) [Ahme74] for image processing has become very popular. For typical image data, the DCT has very close performances to the Karhunen-Loève Transform (KLT). The KLT is well known to be optimal, but it does not exist any fast computational method for it. The DCT is better than other orthogonal transforms (DFT, DWHT, DST, etc.) considering several important criteria: rate distortion, data decorrelation, energy compression, etc. [Mats85]. Therefore, it is widely used for image coding.

Using the inherent properties of the transform, many algorithmic methods have been developed over the past years to allow a faster computation of the DCT. These efforts lead to *fast transforms* like the FFCT [Vett84]. Almost all the research efforts made for reducing the transform processing time have been made in the context of microprocessor based systems. Consequently, the optimization criteria for various algorithms have been mainly focused on the reduction of the number of multiplications. However, once VLSI implementations are considered, this reduction is no longer a necessity. The principle objective is then to optimize the decomposition of the algorithms by taking into account factors such as processing speed, chip surface, and most of all the global regularity of the obtained structure. In this paper it is shown by an example that the use of distributed arithmetic for such implementations offers very interesting perspectives.

2. The discrete cosine transform

Because of the separability of the 2-dimensional DCT algorithm, the problem of computing an NxN-point 2–dimensional DCT is reduced to the problem of computing 2N N-point 1–dimensional DCT's [Jaya84]. If nothing else is specified, all the discussions of this paper refer to 1-D transforms.

A very promising version of the DCT has been chosen for implementation, namely the *Modified Symmetric DCT* (MSDCT), first proposed in [Mats84]. The MSDCT is a variant of the *Symmetric DCT* (SDCT) [Kita80] offering some additional advantages. It is important to mention that both the SDCT and the MSDC have identical forward and inverse transform matrices. This is very important for VLSI implementation, since only one chip is required. Given an N point data sequence $\{x(n)\} = x(0), x(1),..., x(N-1)$, the forward and the inverse MSDCT have the following form:

$$X(k) = \sqrt{\frac{2}{N-1}} \sum_{n=0}^{N-1} c_n \, x(n) \cos\left(\frac{nk\pi}{N-1}\right) \quad ; k = 0,1,...,N-1 \qquad (1)$$

$$x(n) = \sqrt{\frac{2}{N-1}} \sum_{k=0}^{N-1} c_k \, X(k) \cos\left(\frac{nk\pi}{N-1}\right) \quad ; n = 0,1,...,N-1 \qquad (2)$$

where $$c_i = \begin{cases} 1/2 & ; i=0 \text{ and } i=N-1 \\ 1 & ; \text{else} \end{cases}$$

3. The arithmetic

The presented approach is based on *Distributed Arithmetic* (DA) [Pele74]. This technique is very suitable for calculating short length discrete transforms. The basic operation of every discrete transform is the *innerproduct*, (vector multiplication $\mathbf{a}^T \cdot \mathbf{x}$):

$$y = \mathbf{a}^T \cdot \mathbf{x} = \sum_{i=0}^{N-1} a_i \, x_i \qquad (3)$$

The input signals (vector **x**) and the coefficients (vector **a**) can be expressed in two's complement form, in which case the summation expression becomes:

$$y = \sum_{i=0}^{N-1} a_i \left[\sum_{j=1}^{W_d-1} x_{ij} \, 2^{-j} - x_{i0} \right] \qquad (4)$$

Where W_d is the signal or data wordlength. The summation order is then interchanged,i.e. the summation is first made over the terms between the j:th bit of x_n times a_n , and then over the bits. A final subtraction is made for the sign bit (bit 0).

$$y = \sum_{j=1}^{W_d-1} \left[\sum_{i=0}^{N-1} a_i x_{ij} \right] 2^{-j} - \sum_{i=0}^{N-1} a_i x_{i0} \qquad (5)$$

Defining F_j as a function of the jth bit of each x_n :

$$F_j = a_0 x_{0j} + a_1 x_{1j} + a_2 x_{2j} + ... + a_{(N-1)} x_{(N-1)j} \qquad (6)$$

The summations can be written as:

$$y = \sum_{j=1}^{W_d-1} F_j 2^{-j} - F_0 \qquad (7)$$

F_j can only take 2^N different values, which can be precalculated and stored in a *look-up-table*, for example a *ROM*. The j-th bit of x_0 to x_{N-1} are used to address this table. *Horner's* form can be used to compute (7):

$$y = (...((0 + F_{W_d-1}) 2^{-1} + F_{W_d-2}) 2^{-1} + ... + F_1) 2^{-1} - F_0 \qquad (8)$$

The resulting hardware is very simple: to implement the innerproduct, only an *adder/subtractor with accumulator register* (ALU), and a *ROM* table are required (Fig. 1).

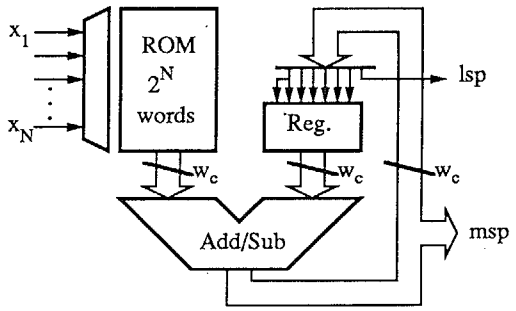

Fig. 1 *Distributed Arithmetic Processor.*

It is not necessary to implement a full w_d bit wide ALU. In fact only w_c bits have to be processed (where w_c is the ROM coefficient wordlength). This saves a lot of hardware since $w_c<w_d$ in most applications. The division by two is implemented as a shift at the input of the accumulator register. The inputs x_1 to x_N are shifted in with the least significant bit first. This generates the address of the ROM where F_j is stored. F_j is added to the contents of the accumulator register and the result is stored back into this register after a shift,i.e., a division by two. This is repeated for all bits of the inputs with exception for the sign-bit, for which a subtraction is performed instead.

4. Implementing the DCT using DA

To realize an N point DCT N DA–processors, (each of them computes an N-terms innerproduct) are required. However, for a 16 point DCT the ROM of each DA-processor contains 2^{16} coefficients: most of the resulting chip area would consist of ROM. Fortunately, it is possible to reduce the ROM size using the symmetries of the DCT base-functions [Mats85]. For instance, it is simple to verify that the rows of the MSDCT transform matrix are alternately symmetrical or antisymmetrical around the central column:

$$X(k) = x(0)\, a_{k0} + x(1)\, a_{k1} + ... \pm x(N-2)\, a_{k1} \pm x(N-1)\, a_{k0} \qquad (9)$$

By simple pre-addition or pre–subtraction of the inputs, the DCT is computable using (N/2)-terms innerproducts:

$$X(k) = [x(0) \pm x(N-1)]\, a_{k0} + [x(1) \pm x(N-2)]\, a_{k1} + ... \qquad (10)$$

The DCT computation still needs N DA-processors,but the ROM of each processor only contains $2^{N/2}$ coefficients. For a 16 point DCT this results in 256 coefficients instead of 65536.

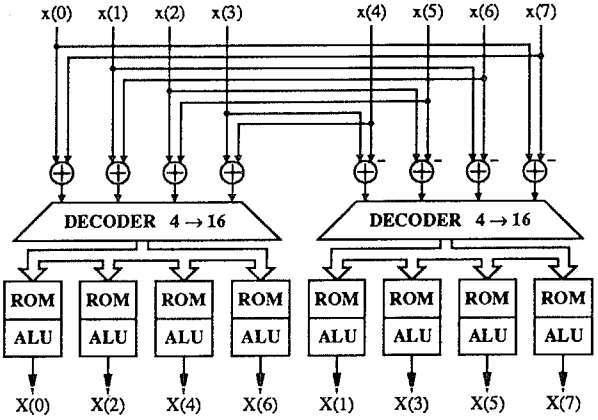

Fig. 2 *8 point DCT using 8 parallel DA-processors. The ROM size is reduced using row symmetry.*

5. The 2-D 16 point DCT chip

The *specifications* of the developed chip have been chosen in order to allow digitized video signal processing: the input and output data wordlength is maximum 16 bit and the coefficient wordlength is 10 bit. All measures (areas, densities, speed limits, etc.) reported here refer to the used technology, CMN20a, which is a double metal, poly-gate, *2μm CMOS technology* supplied by VLSI Technology Inc. (VTI).

5.1 Global system architecture

The global architecture of the operative part of the chip is shown in Fig. 3.

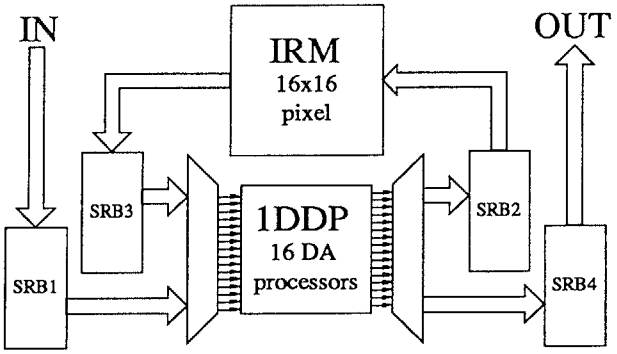

Fig. 3 *Global architecture of the 2-D DCT chip.*

The chip mainly consists of three parts: the *1-D DCT Processor* (1DDP), the *Intermediate Result Memory* (IRM) and the *Shift Register Banks* (SRB).

The core of the chip is the 1DDP. It contains 16 DA–processors, which allow the concurrent computation of 16 innerproducts (in other words the computation of a full 1-D 16 point DCT) according to the scheme of Fig. 2.

The IRM is a relatively high speed single access RAM, which is used to store the results after the first dimension has been transformed. On chip storage of intermediate results allows a reduction of the I/O dataflow with a factor of two.

Furthermore, the IRM implicitly performs the required matrix transposition (rows are written in while columns are read out).

There are 4 shift register banks, each of them contains 16 registers of 16 bit each. The input registers SRB1 and SRB3 are of parallel load and serial out kind; they accept the input data respectively from outside the chip and from the IRM. Once an input register bank is full, its content is shifted out bit-by-bit to the 1DDP. The output registers SRB2 and SRB4 are of serial in and parallel out kind; they are loaded alternatively with the computed innerproducts coming in a serial form from the 1DDP. Once an output register bank is full, its content is stored into the IRM (SRB2) or is transferred to the outside of the chip (SRB4).

The principal advantages of this architecture are:

- Maximum possible parallelism (16 parallel processors).
- Optimal use of the hardware, i.e. all parts of the chip are active all the time.
- Every part of the chip works synchronously (a single clock is required).
- Completely pipelined structure allowing really high speed and throughput.
- Innerproduct operations yields minimal quantization noise.

The use of 16 bit registers is advantageous even for data of smaller wordlength (images are typ. coded on 8 bits) since there are enough guard bits to avoid overflow detection and correction logic. Furthermore the SRB's are completely synchronous with the rest of the chip. This simplifies the Control Unit and saves a lot of area for clock generators in comparison with the small area increase due to the use of 16 bit registers.

It should be noted that SRB1 and SRB4 can communicate with the external world but not with the IRM; on the other hand, SRB2 and SRB3 can communicate with the IRM but not with the external world. This leads to an area efficient hard-wiring of most of the chip components. Anyway, the time scheduling and the dataflow insure an optimal utilization of all the chip components without any supplementary interconnection channel (see section 5.5).

5.2 The Distributed Arithmetic Processors

As seen in section 3, a DA-processor basically consists of two parts: the ALU and the ROM. Clearly the most important part is the ALU. It will determine most of the characteristics of the DA-processor (speed, area, power consumption, etc.), and it will influence the design of the ROM.

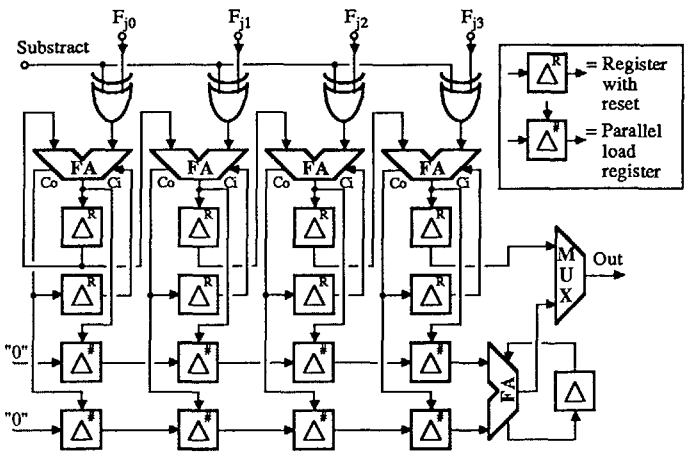

Fig. 4 *Carry-Save Adder with pipeline registers.*

An efficient implementation of the ALU is the so-called *Carry-Save Adder* (CSA) which is shown in Fig. 4 [Siks81]. The main feature of this adder is the elimination of the carry-propagate time; the computational path between the delay-elements is minimal. Another useful property of the CSA is the built-in add-shift-accumulate operation, which exactly corresponds to the needs of DA.

In DA, the result is not available directly after the last bit of the inputs has been shifted into the ALU, this because of the pipelining of the structure. All the contents of the registers must be shifted out (at the cost of W_c extra clock cycles) before the computation of the successive innerproduct can start. To avoid this problem and increase the troughput, *pipelining registers* (the registers with parallel load in Fig. 4) have been added. When the last bit of an innerproduct is computed, the contents of the sum- and carry-registers are downloaded into the pipelining registers. Thus, the computing of the successive innerproduct can start immediately. The pipelining registers are of parallel in, serial out kind. A full-adder is used at the output to sum up the partial sums and carries in the proper way. To complete the processor, a multiplexer is used to select the desired part of the result.

Due to speed and area constraints, *dynamic logic* has been chosen for the ALU registers. A compact *pass-transistor full-adder* with excellent speed properties (only two gate delays) is used [Reus83]. The ALU design has been done very carefully; in particular the layout is fully hand-crafted. The obtained bit–slice contains 71 transistors and is very compact and fast. The resulting area (abutment box) is only 0.02135 mm^2, corresponding to a density of 3300 MOS/mm^2. Simulations and test integrations have shown that the ALU works at clock rates up to 100 MHz.

5.3 The shift register banks

The 4 shift register banks are all based on the same basic cell (Fig. 5). This cell is also implemented using *dynamic logic*, it contains 10 transistors, and has an area of 2562 μm^2 (3900 MOS/mm^2).

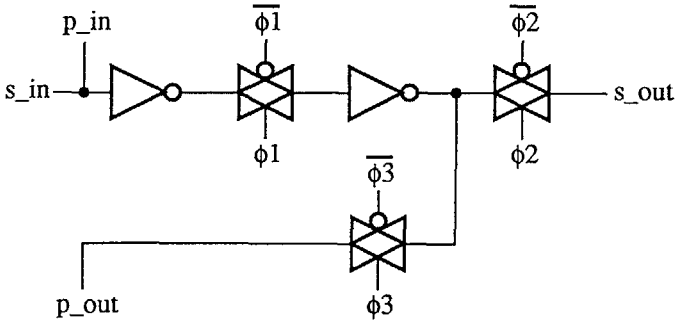

Fig. 5 *Basic shift register cell.*

Shift register banks of any size can be composed by abutment of this cell. The different combinations of $\phi 1$, $\phi 2$ and $\phi 3$ allow serial or parallel propagation of the data.

5.4 The Intermediate Result Memory

For the IRM the standard VTI *static RAM* compiler has been used. No special speed or area optimizations have been performed. This is principally due to the fact that the IRM represents a minor part of the total chip area. For future developments where higher speed or density are required, the design of a special RAM could be necessary.

5.5 Time scheduling

The time scheduling of the chip is simple and intuitively readable from the chip architecture:

1) During 16 clock cycles a row of the original frame is loaded from outside the chip into SRB1.
2) During the subsequent 16 cycles the content of SRB1 is shifted into 1DDP (the innerproducts calculation starts).
3) During the subsequent 16 clock cycles the innerproduct is terminated (pipelining) and shifted into SRB2.
4) During the subsequent 16 clock cycles the content of SRB2 (transformed row) is loaded into IRM.

This procedure is repeated 16 times until the whole frame has been loaded on the chip, transformed and stored into IRM. The same frame is then transformed in the vertical direction:

1) During 16 clock cycles, one column of the intermediate data is read from IRM and written into SRB3.
2) During the subsequent 16 cycles the content of SRB3 is shifted into 1DDP (the innerproducts calculation starts).
3) During the subsequent 16 clock cycles the innerproduct is terminated (pipelining) and shifted into SRB4.
4) During the subsequent 16 clock cycles the transformed column is read outside the chip.

Of course, the register banks work in a symmetrical way, i.e. during SRB1 is loading data form the external world, SRB2 is shifting data into the 1DDP; during SRB1 is shifting data into the 1DDP, SRB2 is loading data from the RAM.

This implies that the chip *works always concurrently on two frames*. When the n:th frame starts to be loaded and transformed row-by-row on the chip, the (n–1):th frame is fully stored in the IRM and is ready to be transformed in the vertical direction. With the time, the transformed rows of the n:th frames will replace the columns of the (n-1):frame in the IRM.

5.6 Chip floorplan

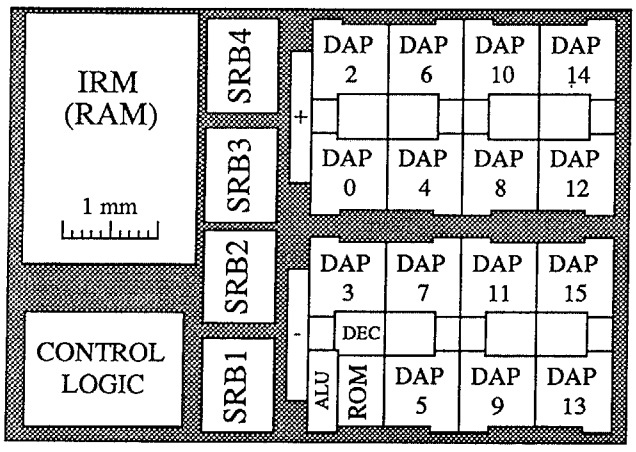

Fig. 6 *Floorplan of the chip core.*

The floorplan of the chip core is shown in Fig. 6. The 16 DA-processors are grouped two by two. This allows an area efficient sharing of the ROM decoder. The chip core occupies 29.25 mm^2 (4.5 mm x 6.5 mm). The total number of MOS is approximately 95'000 subdivided as follows: 44'000 for the IDDP (including pre-adders and -subtractors), 11'000 for the SRB's, 30'000 for the IRM, 10'000 for the control logic. The core density (3250 MOS/mm^2) is very high for a 2 µm technology.

6. Chip performance and achieved results

It can be simply verified that real time processing of 512x512 pixel images at a rate of 32 images/second can be reached using a basic clock speed of 16 MHz. Tests and simulations of the basic cells show that this clock frequency can easily be achieved. The complete layout of the chip is terminated and will be fabricated in the next university multiproject integration.

Without taking into account the control problem, for the computation of the same DCT with a straightforward algorithm, real time processing would require 480 * 10^6 operations (additions or multiplications) per second. Naturally, the computing requirements can be drastically reduced using an FFT-like algorithm [Vett84]. With this improvement, the computation of an 1-D N point DCT, is reduced to $N(\log_2 N)$ additions and $N/2(\log_2 N)$ multiplications. Moreover, a specialized digital signal processor chip, allowing pipelined operations (such as multiply-accumulation), can be successfully employed. Still in this case, the real time computation requires 64 to 128 * 10^6 operations per second.

The basic cells of the chip are able to run with clock rates up to 100 MHz. For this reason it is beleaved that an accurate redesign of some parts of the chip (IRM, I/O buffers and pads) could provide either for high resolution video image processing (1024 x 1024 pixels) or for channel multiplexing.

7. Acknowledgements

This work was supported by FSRM (*Fondation Suisse pour la Recherche en Microtechnique*) under Grant 85/7.

8. References

[Ahme74] N. Ahmed, T. Natarjan, K. R. Rao: "Discrete Cosine Transform", *IEEE Trans. Comput.*, Vol. C-23, pp. 90-93, January 1974.

[Jaya84] N. S. Jayant, P.Noll: *Digital Coding of Waveforms*, Prentice-Hall, Englewood Cliffs, N.J., USA, 1984.

[Kita80] H. Kitajima: "A Symmetric Discrete Cosine Transform", *IEEE Trans. Comput.*, Vol. C-29, No. 4, pp. 317-323, April 1980.

[Mats84] S. Matsumura, B. Sikström, U. Sjöström, L. Wanhammar: "LSI Implementation of an 8 Point Discrete Cosine Transform", *Int. Conf. on Computers, Systems & Signal Processing, Bangalore*, India, December 1984.

[Mats85] S. Matsumura: "Discrete Cosine Transforms - Theory and LSI Implementation", *Linköping Studies in Science and Technology*, Thesis No. 43, Linköping University, Sweden, August 1985.

[Pele74] A. Peled, B. Liu: "A New Hardware Realisation of Digital Filters", *IEEE Trans. Acoust., Speech, Signal Processing*, Vol. ASSP-22, No. 6, pp. 456-462, December 1974.

[Reus83] P. P. Reusens: "High Performance VLSI Digital Signal Processing Architectures and Chip Design", *Ph. D. Dissertation*, Cornell University,USA, 1983.

[Siks81] B. Sikström, L. Wanhammar: "A Shift-Accumulator for Signal Processing Applications", *Proc. European Conf. on Circuit Theory and Design, ECCTD'81*, The Hague, The Netherlands, 1981.

[Vett84] M. Vetterli, H. J. Nussbaumer: "Simple FFT and DCT Algorithms with Reduced Number of Operations", *Signal Processing*, No. 6, pp. 267-278, 1984.