# Use of Redundant Arithmetic on Architecture and Design of a High Performance DCT Macro-bloc Generator

**Roselyne CHOTIN, Yannick DUMONTEIX, Habib MEHREZ**

**University Pierre & Marie Curie**
**LIP6/ASIM Laboratory**
**Tour 55-65 2e etage**
**4, place Jussieu**
**F75252 Paris cedex 05 - France**

**Roselyne.Chotin@lip6.fr, Yannick.Dumonteix@lip6.fr, Habib.Mehrez@lip6.fr**

## Abstract

In view of the performance obtained through the use of redundant operators, it would appear interesting to generalize the redundant arithmetic in high computational digital circuits. To illustrate this, this paper presents the implementation of a high performance Discrete Cosine Transform (DCT) macrogenerator.
The proposed architectures are applicable to the real-time processing of image compression standards (MPEG, HDTV...)

## 1. Introduction

The redundant arithmetic is not very often used for high computational digital circuits considering the excellent performances of the fundamental operators which are adder [1] [2] and multiplier [3]. To show the real interest of redundant arithmetic, a new hardware implementation of the Discrete Cosine Transform is proposed.
The Discrete Cosine Transform (DCT) was first proposed by Ahmed et al in 1974 [4]. It is commonly used in various standards for image compression (H.261, JPEG, MPEG...). Some DCT processors have already been implemented ([5], [6], [7]), but not recently. A new implementation of the DCT with better performance and targeting a better technology is proposed.
The purpose is to be compatible with the different standards and in particular HDTV. The worst case is to process an image with 1 pixel/cycle rate regarding the maximal bandwidth imposed by the HDTV standard. The processor have to compute 60 images of 1280x720 pixels in one cycle with 8x8 pixels DCT blocks. This leads to a frequency of 83 MHz.
This paper presents the development of a macrogenerator which is independent of the different standards and portable on different technologies. Low power consumption is a mandatory requirement for portable devices such as digital video cameras, videoconferencing with cellulars... So most of the efforts have been devoted to reduce the power and complexity by using redundant arithmetic instead of classical one. Section 2 describes the main advantage of the redundant arithmetic and the various redundant notations. Section 3 presents our algorithmic choices after a short review on the DCT. Section 4 describes the design of the DCT architecture. Section 5 details the implementation method and the section 6 presents the implemented DCT performances.

## 2. Redundant arithmetic

Both main number representations are used in redundant arithmetic : the Carry Save (CS) notation where a digit is defined by $cs_i = cs_i^0 + cs_i^1$ with $cs_i \in \{0, 1, 2\}$ and the Borrow Save (BS) notation where a digit is defined by $cs_i = cs_i^+ - cs_i^-$ with $cs_i \in \{-1, 0, 1\}$.
The principal reason to use redundant arithmetic is the excellent performances of the fundamental operator architecture which are adder and multiplier.
The table 1 presents internal performances of the ope-

rators used in the DCT in 0.35 $\mu M$. The abbreviation NR and CS are respectively used for classical (non redundant) and redundant arithmetic.

| **operator**[1] | delay ns | | area $\mu M^2$ | | power cons.[2] $\mu W/Mhz$ | |
|---|---|---|---|---|---|---|
| 8 bits NR+NR=>NR | 2.8 | ref | 12127 | ref | 7.2 | ref |
| 8 bits CS+CS=>CS | 2.5 | -11% | 11760 | -3% | 7.4 | 3% |
| 16 bits NR+NR=>NR | 3.7 | ref | 27011 | ref | 16.3 | ref |
| 16 bits CS+CS=>CS | 2.6 | -30% | 25908 | -4% | 16.5 | 1% |
| 32 bits NR+NR=>NR | 4.7 | ref | 61740 | ref | 35.4 | ref |
| 32 bits CS+CS=>CS | 2.7 | -43% | 53348 | -14% | 36.2 | 2% |
| 8 bits NR*NR=>NR | 8.4 | ref | 68906 | ref | 55.7 | ref |
| 8 bits NR*CS=>CS | 6.1 | -27% | 54941 | -20% | 29.9 | -46% |
| 16 bits NR*NR=>NR | 12.5 | ref | 262885 | ref | 188 | ref |
| 16 bits NR*CS=>CS | 9.2 | -26% | 257556 | -2% | 141.1 | -25% |
| 24 bits NR*NR=>NR | 14.5 | ref | 679385 | ref | 381.3 | ref |
| 24 bits NR*CS=>CS | 10.8 | -25% | 622912 | -8% | 351.4 | -8% |

Table 1: Adder and multiplier performances

# 3. DCT algorithms

## 3.1. 2-D DCT

The 2-D Discrete Cosine Transform (DCT) is defined as follows :

$$DCT_{i,j} = \frac{1}{\sqrt{2N}} c_i c_j \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} p_{x,y} \cos\frac{(2x+1)i\pi}{2N} \cos\frac{(2y+1)j\pi}{2N}$$

$$c_i = \frac{1}{\sqrt{2}} \text{ if } i = 0 \text{ and } c_i = 1 \text{ if } i \neq 0$$

To reduce the complexity, it is possible to calculate a 2-D DCT by performing a 1-D DCT on the rows and then a 1-D DCT on the columns (see table 2).

## 3.2. 1-D DCT

The 1-D DCT is defined as follows :

$$DCT_i = \frac{2}{\sqrt{2N}} c_i \sum_{x=0}^{N-1} p_x \cos\frac{(2x+1)i\pi}{2N}$$

$$c_i = \frac{1}{\sqrt{2}} \text{ if } i = 0 \text{ and } c_i = 1 \text{ if } i \neq 0$$

Complexity can be reduced using DCT Signal Flow Graph (see table 2).

| | Block LxL | | Block 8x8 | |
|---|---|---|---|---|
| | Mul | Add | Mul | Add |
| 2D-DCT | $L^4$ | $L^4$ | 4096 | 4096 |
| 1D-DCT | $2L^3$ | $2L^3$ | 1024 | 1024 |
| Graphs | $L^2 log_2 L$ | $2L^2 log_2 L$ | 192 | 384 |

Table 2: DCT complexity

[2] All operators presented have been placed and routed with the Cadence CAD system using the Alliance Standard Cell library [8] in 0.35$\mu M$.

[2] The adders have not good area and power consumption performances because the full-adder standard cell used is not very effective

## 3.3. 1-D DCT Signal Flow Graphs

Many different algorithms have been proposed to compute the 1-D DCT. The Signal Flow Graph has been widely used for DCT implementation due to its smaller complexity. The complete implementation of a graph permits to compute the 1-D DCT of 8 pixels in only one cycle. The required rate is of 1 pixel/cycle, so a graph partitioning is necessary. To reach this rate, the partitioning will be at the most of 8 sub-blocks to compute the 8 results of the DCT.

| Graph | Add | Mul | critical path | blocks | loopback |
|---|---|---|---|---|---|
| Chen[9] | 4 | 4 | 2 add + 1 mul | 8 | 1 |
| B. J. Lee[10] | 6 | 2 | 1 add + 1 mul | 8 | 3 |
| Loeffler[11] | 5 | 3 | 3 add + 1 mul | 6 | 1 |

Table 3: Graph properties

The table 3 presents the properties of the graph sub-blocks : number of adders and multipliers, critical path, number of sub-blocks to compute the complete 1D-DCT, and number of required loop-backs.

## 3.4. Requirements and choices

The Chen's graph is discarded because of the number of multipliers and sub-blocks required. The critical path is long and no cycles are available to add pipeline stages.

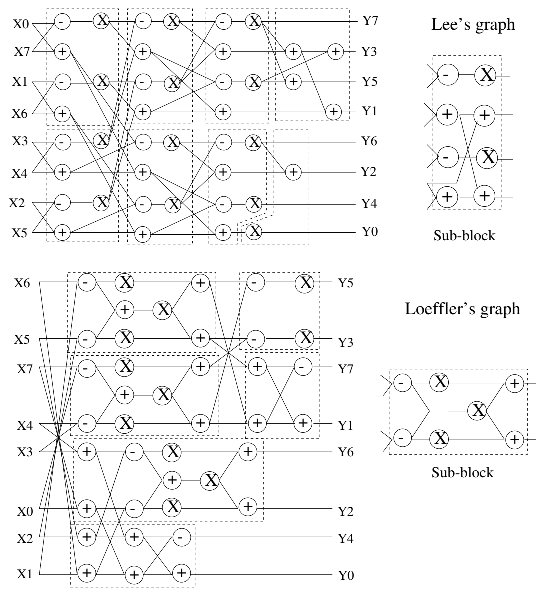


Figure 1: Graph partitioning

At first sight, the Lee's graph seems to be better than Loeffler's (less multipliers and better critical path). But with the Loeffler's graph, two cycles are available to add pipeline stages and reduce the critical path.
Furthermore the Lee's graph requires four sub-blocks

in serial to compute the complete 1-D DCT which leads to a precision problem. In fact, at the output of the sub-block, the bit-dynamic is higher than in the input. Then the input bit-dynamic have to be extended. The length of the extension is determined by the number of sub-blocks used in serial to compute the complete 1-D DCT (three in this case). So only two multipliers are required but they are slower and their bit-dynamic are bigger, otherwise the results are less precise. Then our choice is to implement the Loeffler's graph (see figure 1).

# 4. Architectures

## 4.1. Global system

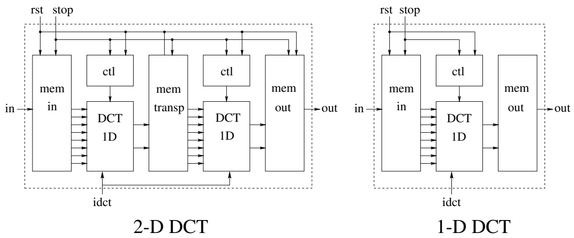


Figure 2: System architecture

The system implementing the 2-D DCT includes : two operative parts which implement the 1D-DCT calculation, input/output synchronizer which are the memories to store the data and a memory which transpose data between the two passes of 1-D DCT (see figure 2). A study of the processor's architecture is done with both redundant and classical arithmetic to estimate the impact of redundant architecture.
If only 1-D DCT have to be realized, one operative part is used with no transposition memory. The figure 2 presents the 1-D DCT and 2-D DCT architectures. The different parts of this system are described in the next subsections.

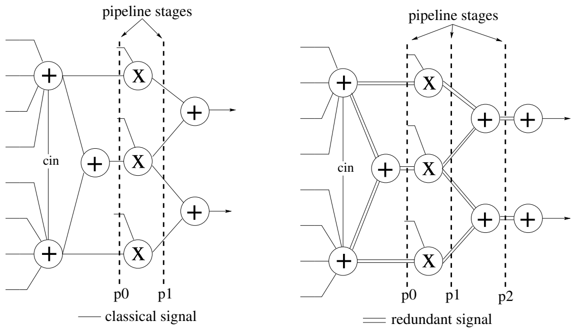


Figure 3: Classical/redundant operative part

## 4.2. Operative part

### 4.2.1. *Classical and redundant arithmetic*

Regarding Loeffler's graph, a 6 blocks partitioning can be performed according to the required rate of 1 pixel/cycle. Two supplement cycles are available to add two pipeline stages without changing the rate and latency. The figure 3 presents the possible architectures of a partitioning block with classical or redundant arithmetic where $p_i$ represent the different possible pipeline stages.

### 4.2.2. *Sequencing*

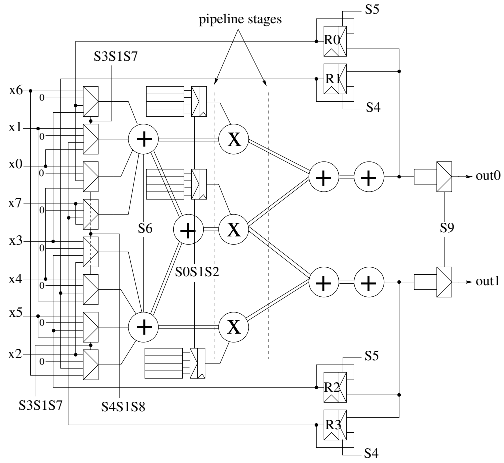


Figure 4: Redundant block architecture

As only sub-block is used to perform all calculations, then some sequencing is necessary (see figure 4). Multiplexers are used to give the correct input data to each operator. Three ROMs memories or register banks are used to store the coefficient values. On the output of the operator, a multiplexer is used to adjust correctly the data. Results are not store in a register, because there is a register after in the transposition memory for the 2-D DCT or output memory for the 1-D DCT (see figure 2). Control signals for the

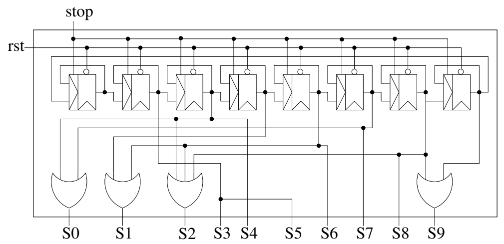


Figure 5: Sequencer of the operative part

different multiplexers come from a small sequencer presented in figure 5. It is a shift register with one set

| | Op1 | Op2 | Mul0 | Mul1 | Mul2 |
|---|---|---|---|---|---|
| 1 | (x6-x1)+(0-0) | (x5-x2)+(0-0) | $\cos\frac{\pi}{16} - \sin\frac{\pi}{16}$ | $-\cos\frac{\pi}{16}$ | $\sin\frac{\pi}{16} + \cos\frac{\pi}{16}$ |
| 2 | (x0-x7)+(0-0) | (x3-x4)+(0-0) | $-\cos\frac{3\pi}{16} + \sin\frac{3\pi}{16}$ | $\cos\frac{3\pi}{16}$ | $-\sin\frac{3\pi}{16} - \cos\frac{3\pi}{16}$ |
| 3 | (x3-x0)+(x4-x7) | (x2-x1)-(x5-x6) | $\frac{1}{2}(\cos\frac{6\pi}{16} + \sin\frac{6\pi}{16})$ | $-\frac{1}{2}\cos\frac{6\pi}{16}$ | $\frac{1}{2}(\sin\frac{6\pi}{16} + \cos\frac{6\pi}{16})$ |
| 4 | (x3+x0)+(x4+x7) | (x2+x1)+(x5+x6) | 0 | $\frac{1}{2\sqrt{2}}$ | $-\frac{1}{\sqrt{2}}$ |
| 5 | (R0-R3)+(0-0) | (R1-R2)+(0-0) | $-\frac{1}{2}$ | 0 | $-\frac{1}{2}$ |
| 6 | (R2-0)+(R1-0) | (R0-0)+(R3-0) | 0 | $\frac{1}{2\sqrt{2}}$ | $-\frac{1}{\sqrt{2}}$ |

Table 4: Operation scheduling

and 7 reset registers for the begin of the calculation. The sequencer can be frozen in the current state with a stop signal. The table 4 explains which operations have to be executed during the current state.

### 4.3. Input/output synchronizer

#### *4.3.1. Freeze*

All input and output data are not used at the rate of 1 data/cycle and that can result in synchronization problems with the outside world. Since the stop signal can frozen the processor in the current state. When the stop signal returns to zero, the calculations resume from the frozen state.

#### *4.3.2. Input/output management and transposition memory*

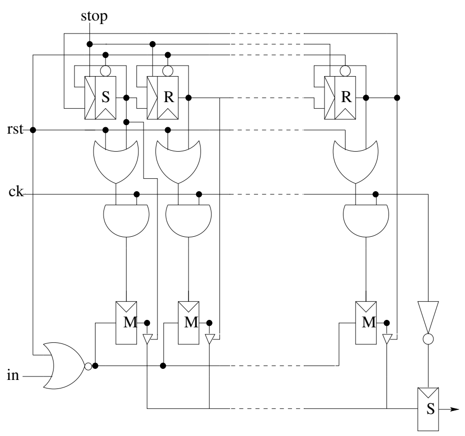


Figure 6: Storage principle

To store a pixel at the input of the system, a storage unit is used as show in the figure 6. A small sequencer (as described before) permits to load and store the pixel in the current state without the need of defining a R/W control input and a storage address. At each cycle, one of the sequencer's register is set to one. That activates the write input of a master register and the tristate gate at the output of another register. Then the input pixel is written in the register addressed by the sequencer and a previously stored pixel is read from another register also addressed by he sequencer through the tristate gate. The same unit is used on output to schedule the data. A memory allows to transpose data between the two passes of 1-D DCT for the 2-D DCT. It is a two blocks of 8x8 pixels memory which uses the storage principle to store and load data.

#### *4.3.3. System interfaces*

The global system can be compatible with the bi-buff interface (see figure 7). That is a synchronous communication interface producer/consumer which uses two switch buffers of 8 or 64 pixels on input. Buffers are on consumer side and the producer have write access to the buffers. The producer writes in the first buffer while the consumer (DCT) reads in the second one. The $OK_i$ signals inform the consumer that the buffers have been written and are available and the $RDY_i$ signals that the buffers have been consumed.

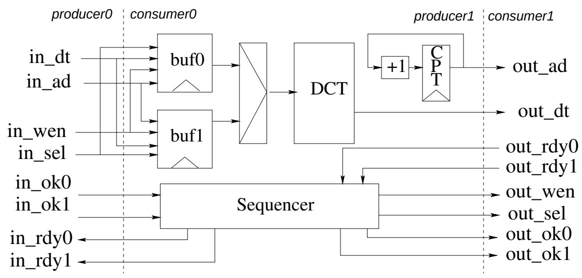


Figure 7: Bi-buff interface

## 5. VLSI implementation methodology

To implement the DCT presented in this paper, we have designed a parameterizable generator under GenOptim [12].

### 5.1. GenOptim CAD environment

GenOptim allows the designer to concentrate exclusively on the algorithm to be implemented. Furthermore, it makes it possible to perform modifications without intervention on the source code of the generator. Consequently, the developer of a generator is never bothered by electrical, placement and timing constraints. GenOptim allows the user to quickly design efficient macro-cell generators independent of

| | Frequency MHz | | Area $mm^2$ | | Power cons. $\mu W/MHz$ | | Transistors number | |
|---|---|---|---|---|---|---|---|---|
| Ours (0.35$\mu m$) | 96 | ref | 1.85 | ref | 2661 | ref | 242K | ref |
| [5] (2$\mu m$) | 14.3 | 571% | 50.4 | 2624% | 25200 | 847% | 73K | -70% |
| [6] (0.8$\mu m$) | 18 | 433% | 10 | 440% | 41700 | 1467% | 72K | -70% |
| [7] (0.8$\mu m$) | 100 | -4.2% | 21 | 1035% | | | 102K | -58% |

Table 5: Compare with other DCT processors

the target technology. These macro functions (shifters, multiplexers, adders, etc.) are very common in the VLSI design world and may be used in many different contexts [13],[14]. GenOptim uses a set of C functions which allow the user to describe the netlist, the layout, the test vectors and the behavioral views. As shown in figure 8, the design starts with a virtual description (netlist, and layout of virtual cells) of the circuit. GenOptim then maps the virtual description to the target library and optimizes the circuit. A successfully fabricated image convolution has been reported in [14].

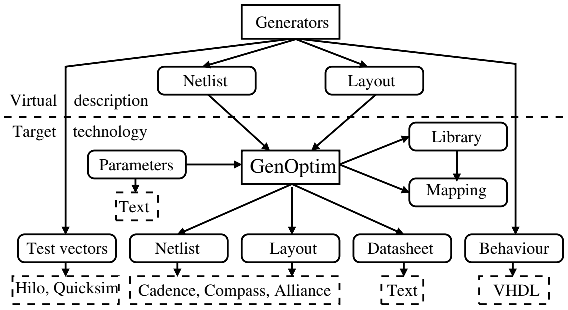


Figure 8: GenOptim description

### 5.2. Generator description

The generator is built from two distinct parts. The first part is different for each generator and describes the algorithm in terms of virtual cells. The other is common to all generators. It calls GenOptim, which maps the virtual cells on the target library and solves automatically the electrical problems by analyzing the circuit and the cell library used.

For our DCT generator, the parameters are data size (pixels, coefficients, output and loopback), the kind of DCT (1D or 2D) and the system interface (bi-buff or not).

## 6. Results

A complete system has been generated for a 2D DCT with 8 bits input pixels, 12 bits coefficients, 16 bits results and 20 bits loop-back. The circuit has been placed and routed with the Cadence CAD system using the Alliance Standard Cell Library [8] in 0.35 $\mu M$. Those standard cells are not specially de-

| Arithmetic | Pipeline | area $mm^2$ | | power cons. $\mu W/Mhz$ | | freq MHz | transistors number | |
|---|---|---|---|---|---|---|---|---|
| Classical | without | 0.37 | ref | 501 | ref | 36 | 44788 | ref |
| | p0, p1 | 0.38 | 2% | 488 | -3% | 75 | 47898 | 7% |
| Redundant | without | 0.36 | -3% | 506 | 1% | 45 | 40401 | -10% |
| | p0, p1 | 0.41 | 11% | 427 | -15% | 116 | 47909 | 7% |
| | p0, p2 | 0.4 | 8% | 372 | -26% | 96 | 46349 | 3% |

Table 6: Operative part results

signed for a low power consumption and has no latches (only M/S registers) that is a problem for area and power consumption of memories. But this library is the only which permits to have a power consumption estimation. The power consumption computation is based on a tool presented in [15]. The design features of the operative part are presented in table 6 using classical and redundant arithmetic.

The table 7 shows the performances of the different blocks. The operative part used is redundant arithmetic with the (p0,p2) pipeline stage because its has the lower power consumption and satisfies the HDTV frequency.

| Block | Area $mm^2$ | | Transistors number | Power cons. $\mu W/MHz$ | Frequency MHz |
|---|---|---|---|---|---|
| Mem_in | 0.03 | 2% | 5048 | 50 | 415 |
| Bi-buff 64 | 0.16 | 9% | 23248 | 318 | 354 |
| Bi-buff 8 | 0.03 | 2% | 4170 | 45 | 419 |
| Mem_transpo | 0.92 | 50% | 129108 | 1684 | 193 |
| Mem_out | 0.1 | 6% | 15266 | 183 | 322 |
| 1st 1-D DCT | 0.4 | 21% | 46349 | 372 | 96 |
| 2nd 1-D DCT | 0.4 | 21% | 46349 | 372 | 96 |
| TOTAL | 1.85 | 100% | 242120 | 2661 | 96 |

Table 7: Blocks results

Regarding the results, the classical arithmetic can not compute the 2-D DCT at the 83 MHz frequency. The use of redundant arithmetic and pipeline stages permits a lower power consumption (-26%) and a higher frequency (+166%) for an area increase of only 8%.

The transposition memory has been realized with a standard cells library that does not have latches but M/S registers which are twice bigger and have a higher power consumption. So the transposition memory uses about 50% of the chip area and 63% of the power consumption. This can be reduce by placing the transposition memory as an internal RAM or ROM. The table 5 compares our DCT processor with other implementations. Only the processor described in [7] can reach the frequency of 83 MHz imposed by the HDTV standard. But our processor has a lower power consumption and requires a significantly smaller area if compared with the implementation proposed in [7].

## 7. Conclusion

A 1-D/2-D DCT macrogenerator, that is compliant with the standards HDTV, MPEG... has been realized and validated. It is portable on different technologies and configurable. With this macrogenerator a complete 2-D DCT has been realized. This chip has an area of 1.85 $mm^2$, a power consumption of 2.6 $mW/MHz$ (0.8 $mm^2$ and 744 $mW/MHz$ for DCT calculation) and a frequency of 96 MHz. Most of our efforts have been devoted to reduce the power consumption and increase the frequency of the operative part.
Using redundant arithmetic in high computational digital circuits permits to significantly increase the timing performances and reduce the power consumption with a small area increase.

### References


[1] J-M. Muller. *"Elementary Functions, Algorithms and Implementation"*. Birkhauser Boston, 1997.

[2] J.J.F. Cavanagh. *"Digital Computer Arithmetic, Design And Implementation"*. McGraw-Hill, 1984.

[3] Y. Dumonteix and H. Mehrez. "A family of redundant multipliers dedicated to fast computation for signal processing". *ISCAS*, May 2000.

[4] N. Ahmed, T. Natarajan, and K. R. Rao. "Discrete Cosine Transform". *IEEE Transactions on Computer*, C-23, January 1974.

[5] M. T. Sun, T. C. Chen, and A. M. Gotlieb. "Vlsi implementation of a 16x16 discrete cosine transform". *IEEE Trans. Circuits Syst.*, 36, April 1989.

[6] N. Demassieux, G. Concordel, J.-P. Durandeau, and F. Jutand. "An optimized vlsi architecture for a multiformat discrete cosine transform". *ICASSP87*, 1987.

[7] S.-I. Uramoto, Y. Inoue, and al. "A 100 MHz 2-D Discrete Cosine Transform core processor". *IEEE Journal of Solid-State Circuits*, 27, April 1992.

[8] A. Greiner and al ALLIANCE. "A complet set of CAD Tools for teaching VLSI Design". *Third EuroChip Workshop*, 1992. http://www-asim.lip6.fr/alliance/.

[9] W. A. Chen, C. Harrison, and S. C. Fralick. "A Fast computational Algorithm for the Discrete Cosine Transform". *IEEE Transactions on Communications*, COM-25, 1977.

[10] Byeong Lee. "A New Algorithm to Compute the Discret Cosine Transform". *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-32, December 1984.

[11] C. Loeffler, A. Ligtenberg, and G. Moschytz. "Pratical Fast 1-d dct Algorithms with 11 Multiplications". *ICASSP*, 2, 1989.

[12] A. Houelle, H. Mehrez, and N. Vaucher. "On Portable Macro-Cell FPU Generators using the 754-IEEE standard". *ICSPAT*, 1994.

[13] M. Aberbour and al. "Architecture and design methodology of the RBF-DDA neural network". *ISCASS*, 1998.

[14] A. Houelle and al. "A real time image convolver". *ICSPAT*, 1995.

[15] J. Dunoyer. *"Modèles et Méthodes Probabilistes Pour L'évaluation de la Consommation des Circuits Intégrés VLSI"*. PhD thesis, Université Pierre et Marie Curie Paris, 1999. http://asim.lip6.fr/publications/1999/.