MPEG-4 Addressing and ACQ Function

Georgi Kuzmanov, Stamatis Vassiliadis, Jos van Eindhoven*

Delft University of Technology - Electrical Engineering Dept.,
P.O. Box 5031, 2600 GA Delft,The Netherlands
*PHILIPS Research - Dept. of Information and Software Technology,
Eindhoven,The Netherlands

Phone:+31-(0)15-27-86249 E-mail:G.Kuzmanov@ET.TUDelft.NL

Abstract— **This paper proposes architectural solutions that deal with two of the most crucial performance requirements imposed by the MPEG-4 standard, namely high data throughput and high computational power. To obtain higher data throughput, we define a new data storage facility with a specific data organization and a new addressing mode called *two-dimensional block addressing*. To achieve higher computational power, we propose an instruction set extension, benefitting from this addressing mode. This is illustrated by the implementation of a new instruction called "ACcepted Quality" (ACQ). This instruction supports the identically named ACQ function, which is an essential part of the shape encoding process in MPEG-4. We have implemented the ACQ function on a dedicated systolic structure and its FPGA realization suggests 62 ns operating latency. Utilizing this result, we have made performance evaluations with a benchmark software (MPEG-4 shape encoder) by a cycle-accurate simulator. The simulation results indicate that the performance is increased by up to 10%.**

Keywords—**Multimedia Architecture, Data Bandwidth, Two-Dimensional Block Addressing, MPEG-4, Visual Object Plane, Accepted Quality Function, Systolic Structure**

I. Introduction

The fast development pace of new visual data compression standards (e.g.,MPEG-4) is dramatically increasing the system performance requirements, which already exceed the capabilities of current general-purpose architectures[1]. In addition, the increasingly popular media applications are becoming one of the most demanding types of workloads. Therefore, there is a growing need for new architectures or architectural extensions, optimized for multimedia algorithms. Such architectures must enable implementations that would easily meet the performance requirements imposed by MPEG-4 [1][10]. The most crucial of these requirements are high computational power and high data throughput. This paper proposes a new data storage facility for MPEG-4 systems. This facility is linked to multimedia processing units through a high bandwidth interconnection and we propose a new addressing approach to allow a faster data access. The addressing is defined as an architectural issue and utilizes visual data processing algorithms based on block division of data. We define an addressing function over a two-dimensional visual data plane and refer to the corresponding addressing mode as a *two-dimensional block addressing*. To increase the computational power of an MPEG-4 architecture, we propose an instruction that utilizes the advantages of the new addressing approach. The proposed instruction supports the *ACcepted Quality (ACQ)* function, which is defined in [1]. The ACQ function is an essential part of the shape encoding process in MPEG-4 and shows whether the encoding gives an accepted quality results according some specified lossy coding conditions. We have implemented the ACQ function as a scalable systolic array structure and have mapped it onto an Altera FPGA.

The remainder of the paper is organized as follows. Section II gives a short introduction into content representation and shape processing as adopted in MPEG-4. Some background information about visual data compression architectures and about dedicated memory organizations is also discussed. Section III gives a formal definition of the new addressing mode and discusses its possible utilization. Section IV proposes an implementation of the ACQ function, that utilizes the newly defined addressing. In Section V, an evaluation of the implemented structure is presented and its results are reported. Finally, the conclusions of this paper with some future research directions are discussed in Section VI.

[1]In this paper by architecture of any computer system we mean the conceptual structure and functional behavior as seen by its immediate user [2].

II. Background

An important part of the multimedia applications domain is formed by the visual data compression standards (e.g., MPEG). All these standards aim to preserve best possible visual quality at a given bitrate range. MPEG-1 and MPEG-2 are dedicated for data rates in the order of 1.5 Mbit/s and 10 Mbit/s, respectively. The latest complete visual coding standard is MPEG-4. Next to supporting data transmission at very low bit rates (64 kbit/s) it is the first content-based visual data compression standard. While in MPEG-1,2 a whole frame of a video sequence is processed, in MPEG-4 a frame is first decomposed with respect to its content and each decomposed part is then processed independently. In all multimedia video and still-picture applications, data is physically displayed as a two-dimensional array of picture elements (pels). Every MPEG picture is divided into basic building blocks called macroblocks. Each macroblock consists of one 16x16 array of luminance (grayscale) pels and two 8x8 arrays of chrominance (color) pels, which represent the full-color of the corresponding 16x16 picture area.

A. Content representation in MPEG-4

For content-based coding, MPEG-4 uses the concept of a video object plane (VOP). A VOP is an arbitrarily shaped region of a frame, which usually corresponds to a semantic object in the visual scene. A sequence of VOPs in the time domain is referred to as a Video Object (VO). This means that we can view a VOP as a "frame" of a VO. Each VOP has its *shape* - a feature that does not exist in MPEG1/2 standards and was proposed for the first time in MPEG-4 [1][10]. A possible representation of the shape information is the binary format. This format represents the shape as a bitmap, referred to as *binary alpha plane*. Each pel in this plane takes one of two possible values, which indicate whether a pel belongs to the object or not. Each 16x16 pel macro block has a corresponding 16x16 bitmap block referred to as *binary alpha block* (BAB). Previous research efforts show that the most computationally expensive algorithms in MPEG-4 [10] are motion estimation and shape encoding [5][6][9]. These two parts of the standard constitute more than 95% of the computations in the MPEG-4 video encoder and represent the most critical performance bottlenecks.

The MPEG-4 shape encoder is responsible for three basic procedures preceding the actual encoding process, which are: *mode decision*, binary *motion estimation and compensation* and *rate control*. In all these procedures the ACQ function is intensively used.

Mode decision. A mode decision procedure is performed over each BAB [1] and there exist seven modes to code the shape information of each macroblock.

Motion estimation and compensation for shape. This new functionality is adopted by MPEG-4 and is the most demanding part of shape encoding in terms of performance. The motion estimation and compensation for shape is similar to the traditional motion estimation and compensation for video frames, but it is now performed over the binary alpha map.

Rate control is obtained through block level size conversion of all BABs. The conversion ratio (CR) is 1/4, 1/2 or 1 the original size. Each 16x16 BAB is down-sampled to (16xCR)x(16xCR) and then up-sampled back to 16x16 by means of filters described in [1].

B. Visual data processing architectures

In literature [5][6][7][8][9], it is agreed that *systolic array processors*, SIMD and VLIW processors match many of the new multimedia algorithms. In [7] it is advocated that the combination of Simultaneous Multithreaded Processors and streaming vector μ-SIMD instructions can achieve the performance required for future media workloads. The same paper proposes two important architectural concepts as appropriated for the new workload demands - Chip Multi-Processors and Simultaneous Multithreaded Processors.

The support of many functionalities found in multimedia standards (e.g., MPEG-4) is optional. In such cases, it is expensive and not effective to make a hardwired implementation that supports all functionalities in the standards. To keep the implementation of such a complex multimedia architecture at a reasonable cost-performance ratio, a *reconfigurable approach* can be used. In [16] a new processor architecture is proposed that supports reconfiguration at architectural level and achieves high flexibility in tuning a system for a specific application. The reconfiguration and execution processes are controlled by only three new instructions, allowing instructions, entire pieces of code, or their combination to execute in a reconfigurable manner.

C. Visual data and memory

In digital video applications, visual data is scanned and transformed into a sequence of color pels. The full color (luminance+chrominance) values of these pels

are usually stored into linearly addressable memory at subsequent locations. This linear data alignment, however, does not appear to be optimal with respect to visual data processing in MPEG.

Most of the algorithms in the MPEG standards are *data intensive* and have two important features: *data locality* and *data reusability*. These two features mean that intensive data transfers from only a few memory locations are constantly required. In such memory locations (say search area for motion estimation algorithms) data are not processed in the order they are aligned in the linearly byte or word addressable memory. Furthermore, in order to access the right piece of data (Figure 1) in such memories, some time expensive address manipulations and algorithm-dependent data reordering are required. This fact advocates for the need of a different data access approach aimed at achieving a considerable data throughput speed-up.

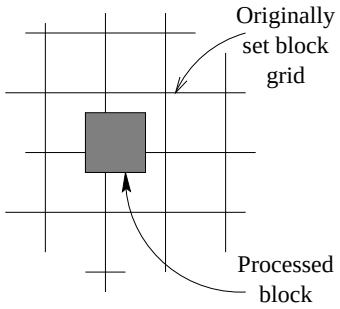

Fig. 1. *Block Data Overlap*

In [12], Park develops the ideas from [4][11][13][15] for two-dimensional data alignment into multiple memory modules. He proposes a faster buffer memory system by separating the address calculation from the address routing and solving the complex control problem of the latter. This concept for data allocation has been used in the design of graphical display systems where it is referred to as a *block subarray access*. However, it is not defined as an *architectural* issue and is not implemented within visual data compression standards.

A flexible architecture adapted to conventional motion-estimation algorithms is proposed in [8]. Some ideas for a specific data-memory organization and access are discussed and a trial-and-error data reordering is proposed for algorithm independent and optimal performance solutions.

An extensive exploration in memory management and organization for MPEG-4 encoders is reported by the researchers from Katholieke University Leuven and Dresden University of Technology [3][14]. However their focus was in the field of low-power consumption. These authors propose combining background and foreground memory in a low-power optimized hierarchy and an approach to design a processor array within the context of the derived memory organization. The power consumption is minimized by dramatically decreasing the number of background memory accesses without sacrificing speed.

III. The Two-Dimensional Addressing Mode

A video frame has a two-dimensional structure, but traditionally the information it contains has been stored in a linearly addressable memory. Since the basic unit being processed in MPEG-4 is the block of pels, we can assume a new storage facility with a two-dimensional logical organization, where the basic addressable units are blocks. This allows the implementation of a memory system with a higher data throughput. This resulted in the definition of the *two-dimensional block addressing* (2DBA).

Definition 1 *We define the Two-Dimensional Block Addressing as the following address function:*

$$A^B_{k,l}(i,j) =$$

$$= \begin{vmatrix} p_{i,j} & p_{i,j+1} & \cdots & p_{i,j+k-1} \\ p_{i+1,j} & p_{i+1,j+1} & \cdots & p_{i+1,j+k-1} \\ \cdots & \cdots & \cdots & \cdots \\ p_{i+l-1,j} & p_{i+l-1,j+1} & \cdots & p_{i+l-1,j+k-1} \end{vmatrix}$$

,where k,l are block dimensions;

$p_{i,j}$ represents pel with coordinates i,j in the addressable area;

$0 \le i,k < M, 0 \le j,l < N$*;*

M,N are the dimensions of the 2D addressable area.

If k=l, the address function can be denoted as $A^B_k(i,j)$, so $A^B_{16}(i,j)$ denotes the 2D address i,j of a 16x16 block.

The definition includes three architectural issues:

- **Two-dimensional data storage** displayed to the immediate user (programmer) of the architecture.
- **Block data type** as a basic addressable data unit.
- **Addressing function** to access blocks of visual data.

The definition states that the 2D address of a block is the same as the 2D-coordinates of its upper-left-most pel in the addressable area. The graphical representation of the 2DBA is depicted in Figure 2.

Figure 3 depicts an abstract design model of the proposed addressing mode. The interconnection network is responsible for routing the right data block

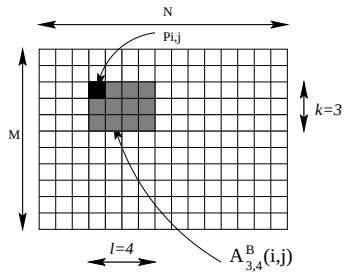

Fig. 2. *2D Block Addressing*

from the memory array to the processing units. We can also refer to the above proposed addressing scheme as a two-dimensional cutting or two-dimensional barrel shifting, performed by the access network block in Figure 3.

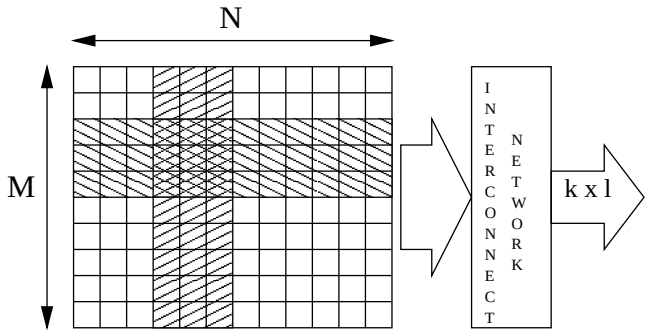

Fig. 3. *An Abstract Design Model*

It is important to note that we define the 2D addressing at the architectural level so it is up to the designer to propose its implementation. The memory can be implemented as an on-chip buffer with dedicated organization. Read and write operations are not symmetrical in MPEG-4 where random block read is the most frequent memory access type while write operations are relatively seldom. Simpler implementations and higher speed-ups are achievable by exploiting the asymmetry between data read and write memory accesses.

IV. Addressing utilization

Besides the data throughput, the computational power of a processor can also be improved by defining instructions that utilize the proposed (block) data type and addressing mode. These instructions should support program kernels or functions that are consistent with three basic preconditions for the processed data: *block-organized visual data*, *data locality* and *data reusability*. A good candidate to utilize the proposed addressing mode that meets these three preconditions is the binary shape encoder in MPEG-4. Among the most important shape manipulations is the verification of the accepted quality of a block encoding - *the accepted quality* (ACQ) function.

A. The Accepted Quality function

In MPEG-4 a decision about a suitable coding mode is made for each BAB in the binary alpha map. An essential part of this process is the necessity to ascertain whether this BAB has an accepted quality under some specified lossy coding conditions. Each BAB is divided into 16 4x4 pixel blocks and this data structure is evaluated with the criterion for an accepted quality. A dedicated function called ACQ is defined in [1]:

Definition 2 *Given the current original binary alpha block i.e. BAB and some approximation of it i.e. $BAB^{'}$, it is possible to define a function*

$$ACQ(BAB^{'}) = MIN(acq_0, acq_1, ..., acq_{15}), \qquad (1)$$

where

$$acq_i = \begin{cases} 0 & if\ SAD\_PB_i > 16 * alpha\_th \\ 1, & otherwise. \end{cases} \qquad (2)$$

$SAD\_PB_i(BAB, BAB^{'})$ is defined as the sum of absolute differences for PB_i, where an opaque pel has value of 255 and a transparent pel has value of 0. The parameter alpha_th *may have one of the following values {0,16,32,64,...,256}.*

In this definition $BAB^{'}$ is the encoded BAB and *alpha_th* formally specifies the lossy coding conditions. The higher the *alpha_th* value is, the less quality of the encoding is acceptable. If *alpha_th=0*, then encoding will be lossless.

We can represent SAD_PB_i as follows:

$$SAD\_PB_i = 255 \sum_{j=0}^{15} |P_{i.16+j} - P^{'}_{i.16+j}| \qquad (3)$$

where $P_{i.16+j}$ and $P^{'}_{i.16+j}$ are the *binarized* values of the j^{-th} pels from PB_i and $PB^{'}_i$ respectively and a value of *0 represents a transparent pel* while a value of *1 - an opaque one*. According to these assumptions we can substitute the absolute difference in (3) with a *xor* operation:

$$SAD\_PB_i = 255 \sum_{j=0}^{15} (P_{i.16+j} \oplus P^{'}_{i.16+j}) =$$

$$= 255(PB_i \oplus PB_i^{'}) = 256(PB_i \oplus PB_i^{'}) - (PB_i \oplus PB_i^{'}) \quad (4)$$

where $PB_i \oplus PB_i^{'}$ denotes the bit sum of the bit-by-bit *xor* over the pel blocks.

According to Definition 2 and Equation (4):

$$acq_i = (SAD\_PB_i \leq alpha\_th * 16) =$$

$$= [256(PB_i \oplus PB_i^{'}) \leq alpha\_th * 16 + (PB_i \oplus PB_i^{'})] \quad (5)$$

and

$$ACQ(BAB^{'}) = AND_{16}(acq_0, acq_1, ..., acq_{15}) \quad (6)$$

According to Definition 2, $alpha\_th * 16 = alpha\_th_5 * 256$, where $alpha\_th_5$ denotes the five MSD of *alpha_th*. On the other hand the result of $(PB_i \oplus PB_i^{'})$ is a five-digit number and we can reduce the acq_i computation to the comparison of two 5-digit numbers as follows: $acq_i = [(PB_i \oplus PB_i^{'}) \leq alpha\_th_5.\frac{256}{255}]$ and since $\frac{256}{255} \approx 1$:

$$acq_i \approx [(PB_i \oplus PB_i^{'}) \leq alpha\_th_5] \quad (7)$$

The implementation of Equation (7) is depicted on Figure 4. We can assume the discussed structure as a basic processing element (PE). To maintain all 16 pixel blocks of the BAB in parallel (taking into account Equation (6)) we can build the systolic processor shown on Figure 5.

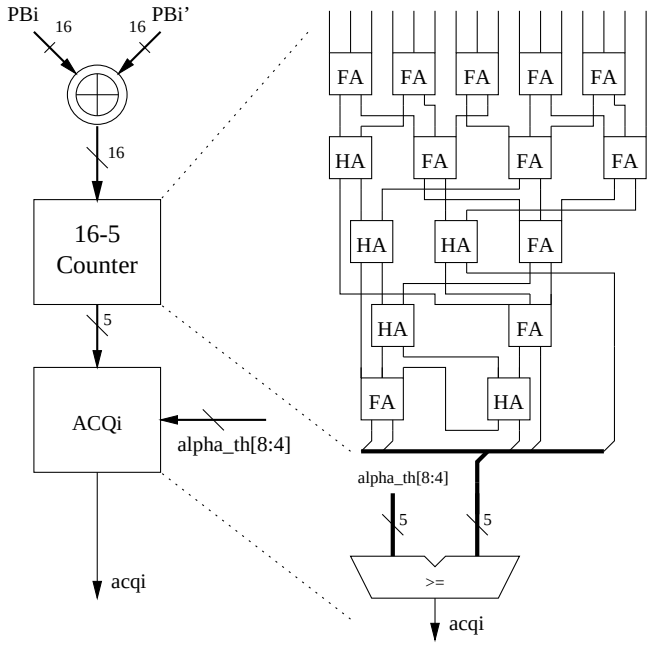

Fig. 4. *Accepted quality single pel-block processing element*

B. Scalability and Data Bandwidth

The proposed circuit would take two cycles for execution in a real implementation[2], and if pipelined it can produce a valid result every cycle, given the data memory bandwidth requirements are met. On the other hand, the structure is scalable and can meet any memory bandwidth restrictions. For its efficiency, however, a multiple of 16 bits per cycle bandwidth is recommended, ranging between 16 and 256 bits/cycle for a single BAB. Figures 4 and 5 show the two extreme cases - a pel block processor and a BAB processor. These two processors differ in the granularity and the throughput of the processed data. If we use the 2D addressing mode over an on-chip memory array for the ACQ engine we can *randomly* fetch the required data amount, thus supplying the optimum data throughput.

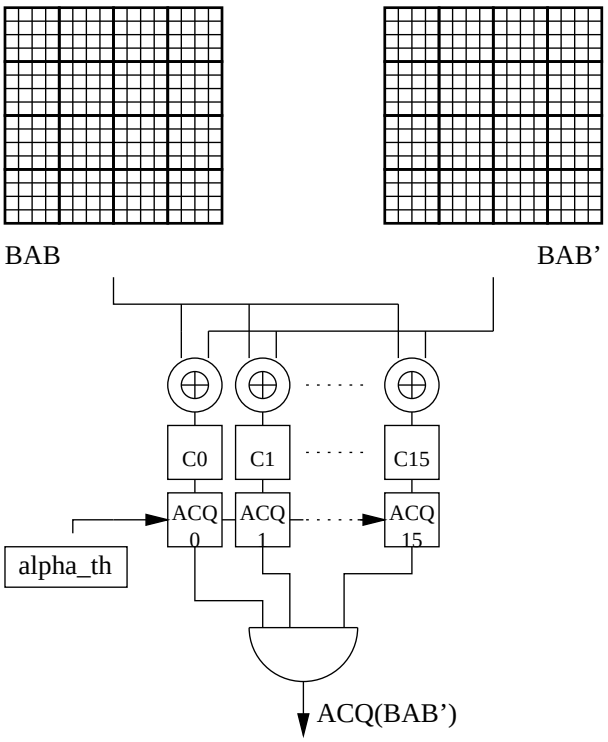

Fig. 5. *The ACcepted Quality processing structure*

V. Evaluation

To evaluate the proposed structure, a single processing element and an array of processing elements have been modeled in VHDL and RTL simulations have been run. The VHDL models have been synthesized for Altera FPGA. The reference software for the evaluation of the structure was Altera Max+Plus II.

[2] A cycle here is considered to be comparable to the cycle of a high speed, 2-cycle multiplier.

The simulation results indicate that each processing element performs the acq_i function within 60 ns. The evaluation of the MIN function takes about 2 ns.

TABLE I

Processing time and required memory bandwidth according to the number of processing elements (for Altera FPGA)

Number of PE	Processing time in ns	BAB/s	Memory bandwidth
1	992	1 008 065	16 bit
2	496	2 016 129	32 bit
8	124	8 064 516	128 bit
16	62	16 129 032	256 bit

Table I suggests the processing latency and memory bandwidth, required for different number of processing elements in an Altera FPGA. The reported data guarantees that the proposed engine can meet the real time constrains of a dedicated MPEG-4 shape processor. To evaluate the structure as an instruction implementation, however, we have to use the reported data into a cycle-accurate simulator of a microarchitecture. We chose the the MPEG-4 shape-encoding algorithm and the SimpleScalar toolset to simulate the ACQ instruction as an instruction set extension of a superscalar MIPS architecture. The simulation results indicated about 10% faster performance for a machine organization adopted from [16].

VI. Conclusions and future work

In this paper we introduced a new addressing mode, referred to as two-dimensional addressing. An instruction, utilizing the new memory access was proposed and its scalable implementation was investigated. We achieved a considerable processing speed-up, due to: a) the two-dimensional addressing is more suitable and faster than conventional addressing schemes when applied over block organized visual data; and b) we defined and implemented a new function that plays a key role in the investigated class of algorithms. Furthermore, the design of the proposed function is highly parallel and at a very low hardware cost. The sum of absolute differences per pixel block basis has been substituted by a pixel block-wise *xor* operation and a parallel counter structure. The thresholding has been performed by a simple, 5-bit comparator.

A combination of the new addressing mode and a set of dedicated instructions utilizing it, appear to be very beneficial for a range of multimedia algorithms. This fact addresses two directions for future research:

Addressing implementation. A fast and cost-effective implementation of the two-dimensional addressing will make the benefits of this approach stronger. Its promising properties may position it as a *basic addressing mode in future MPEG architectures.*

New instructions. Defining a complete set of dedicated instructions with respect to the two-dimensional addressing, forms another research direction for an overall multimedia processing speed-up.

References

[1] ISO/IEC JTC1/SC29/WG11, N3312, MPEG-4 video verification model version 16.0.

[2] G. A. Blaauw and F. P. Brooks. *Computer Architecture: Concepts and Evaluation.* Addison-Wesley, 1997.

[3] E. Brockmeyer, L. Nachtergaele, F. V. Catthoor, J. Bormans, and H. J. D. Man. Low power memory storage and transfer organization for the MPEG-4 full pel motion estimation on a multimedia processor. *IEEE Transactions on Multimedia*, 1(2):202–216, June 1999.

[4] P. Budnik and D. J. Kuck. The organization and use of parallel memories. *IEEE Transactions on Computers*, 20(12):1566–1569, Dec. 1971.

[5] H.-C. Chang, L.-G. Chen, M.-Y. Hsu, and Y.-C. Chang. Performance analysis and architecture evaluation of MPEG-4 video codec system. In *IEEE International Symposium on Circuits and Systems*, volume II, pages 449–452, Geneva,Switzerland, 28-31 May 2000.

[6] H.-C. Chang, Y.-C. Wang, M.-Y. Hsu, and L.-G. Chen. Efficient algorithms and architectures for MPEG-4 object-based video coding. In *IEEE Workshop on Signal Processing Systems*, pages 13–22, 11-13 Oct 2000.

[7] J. Corbal, R. Espasa, and M. Valero. DLP+TLP processors for the next generation of media workloads. In *7-th International Symposium on High Performance Computer Architecture*, pages 219–228, 19-24 Jan. 2001.

[8] S. Dutta and W. Wolf. A flexible parallel architecture adapted to block-matching motion-estimation algorithms. *IEEE Transactions on Circuits and Systems for Video Technology*, 6(1):74–86, Feb. 1996.

[9] H.-J.Stolberg, M.Berekovic, P.Pirsch, H.Runge, H. Moller, and J.Kneip. The M-PIRE MPEG-4 codec DSP and its macroblock engine. In *IEEE International Symposium on Circuits and Systems*, volume II, pages 192–195, Geneva,Switzerland, 28-31 May 2000.

[10] ISO/IEC JTC1/SC29/WG11 N4030. MPEG-4 Overview - (V.18 - Singapore Version), March. 2001.

[11] D. H. Lawrie. Access and alignment of data in an array processor. *IEEE Transactions on Computers*, C-24(12):1145–1155, Dec. 1975.

[12] J. W. Park. An efficient buffer memory system for subarray access. *IEEE Transactions on Parallel and Distributed Systems*, 12(3):316–335, March 2001.

[13] J. W. Park and D. T. Harper. An efficient memory system for the SIMD construction of a gaussian pyramid. *IEEE Transactions on Parallel and Distributed Systems*, 7(8):855–860, Aug. 1996.

[14] R. Schaffer, R. Merker, and F. Catthoor. Combining background memory management and regular array co-partitioning, illustrated on a full motion estimation kernel.

In *13th International Conference on VLSI Design*, pages 104–109, 3-7 Jan. 2000.

[15] D. C. van Voorhis and T. H. Morrin. Memory systems for image processing. *IEEE Transactions on Computers*, C-27(2):113–125, Feb. 1978.

[16] S. Vassiliadis, S. Wong, and S. Cotofana. The MOLEN rm-coded processor. In *11th International Conference on Field Programmable Logic and Applications (FPL)*, 2001.