A 2D Addressing Mode for Multimedia Applications

Georgi Kuzmanov[1], Stamatis Vassiliadis[1], and Jos van Eijndhoven[2]

[1] Delft University of Technology - Electrical Engineering Dept.,
P.O. Box 5031, 2600 GA Delft,The Netherlands
{G.Kuzmanov, S.Vassiliadis}@ET.TUDelft.NL
[2] PHILIPS Research - Dept. of Information and Software Technology,
Eindhoven,The Netherlands
jos.van.eijndhoven@philips.com

Abstract. This paper discusses architectural solutions that deal with the high data throughput and the high computational power - two crucial performance requirements of MPEG standards. To increase the data throughput, we define a new data storage facility with a specific data organization and a new addressing mode. More specifically, we introduce an addressing function and refer to it as two-dimensional block addressing. Furthermore, we propose such an addressing approach, as an architectural feature and we believe it has useful properties that may position it as a basic addressing mode in future multimedia architectures. In addition, we propose an instruction set extension, utilizing the advantages of this addressing mode, as means of improving the computational power of a general-purpose super-scalar processor. To illustrate this concept, we have implemented a new instruction "ACcepted Quality" as a dedicated systolic structure. This instruction supports the corresponding function "ACQ" as defined in the Verification Model of MPEG-4. Its FPGA realization suggests 62 ns operating latency. Utilizing this result, we have made performance evaluations with a benchmark software (MPEG-4 shape encoder) using a cycle-accurate simulator. The simulation results indicate that the performance is increased by up to 10%. The introduced approach can be utilized by data encoding tools, which are based on block division of data. These tools are an essential part of many recent and up coming visual data compression standards like MPEG-4.

1 Introduction

The recent development of multimedia applications made them one of the most demanding types of workloads. Their new performance requirements already exceed the capabilities of current general-purpose architectures[1]. Therefore, the need for architectures, dedicated for the new multimedia applications, provokes the nontrivial problem to define such architectures. On the other hand, the fast development rate of the new visual data compression standards, like MPEG,

[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]

dramatically shortens the time-to-market constraints and increases the flexibility requirements. Furthermore, the enormous memory throughput and high computational power, required by the recent MPEG standards, become crucial in solving the problems with their real-time implementation.

Most of the algorithms in MPEG applications are *data intensive* and have two very important features: *data locality* and *data reusability*. These two features require a very intensive data transfer over a restricted location of data. In MPEG this transfer is non-symmetrical - memory loads are much more in number than memory stores and data are processed identically. These properties can be exploited to achieve a better performance of an MPEG architectural implementation. In addition, the support of many functionalities found in multimedia standards (e.g., MPEG-4) is optional. In such cases, it is not effective to make a hardwired architectural 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 this paper, with architectural solutions, we enable implementations that would easily meet the performance requirements of the new multimedia applications. The reported work represents parts of the research, involved into the development of a reconfigurable microcoded processor within the MOLEN project [20]. In this project, a new processor architecture is proposed that supports reconfiguration at the 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. More specifically, this paper proposes the following solutions. To obtain *higher data bandwidth* we define at the *architectural level* a new addressing mode, referred to as *two-dimensional block addressing*. This addressing mode involves three architectural features:

- *Two-dimensional data storage* displayed to the immediate user of the architecture;
- *Block data type* as a basic addressable unit;
- *Two-dimensional addressing function* for random access of blocks of visual data.

To make the benefits of the defined addressing mode stronger and to *improve the computational power* of the system, we also propose an *instruction set extension*. We have implemented the new instruction "ACcepted Quality" (ACQ), which supports the identically named function in MPEG-4. This instruction utilizes the two-dimensional block addressing and is an essential part of the shape encoding process. The ACQ has been implemented as a scalable systolic structure, described in VHDL. The VHDL source has been synthesized for an FPGA chip, and netlist simulations have been run. The data, reported from the FPGA netlist simulator have been used into a cycle-accurate simulator of an out-of-order superscalar microarchitecture. Assuming Altera FPGAs and the

SimpleScalar toolset [5] for microarchitectural simulations, we reduce the calculation of the ACQ function to 62ns, allowing performance gains of the shape encoder by up to 10%.

The discussion in this paper is organized as follows. Section 2 gives some background information about data processing and organization in visual data compression standards. In Section 3 the problem with visual data alignment in conventional, linearly addressable memories is discussed. Some related work is reported in Section 4. Section 5 proposes the new addressing mode, gives a formal definition of it and suggests its possible utilization. In Section 6 a new function, utilizing the newly defined addressing, is proposed and its implementation is discussed. An evaluation of the proposed structure is performed and the results are reported in Section 7. The conclusions of this paper are included in Section 8.

2 Visual Data Presentation in MPEG

The industrial impact of the new digital technology urged the development of standards for digital video compression. All these standards aim to preserve best possible visual quality at a given bitrate range. In this paper we focus on the MPEG standards and their basic requirements. The first generation video coding standard, MPEG-1, is dedicated for data rates on the order of 1.5 Mbit/s and is intended for storing digital audio-visual information in a storage medium such as CD-ROM. MPEG-2 extends the bitrate to the range of over 10Mbit/s and is currently used as basic coding standard for digital TV broadcasting and High Definition Television (HDTV). The latest complete visual coding standard, MPEG-4 [12][13], enables data transmission at very low bit rates (64 kbit/s). The inclusion of entirely new *content-oriented* functionalities, however, makes most of the specialists refer to MPEG-4 as a new standard generation rather than the next MPEG version. While in MPEG-1,2 a whole frame of a video sequence is processed, in MPEG-4 the frame is decomposed with respect to its content and each decomposed part is processed separately.

In all MPEG standards, visual data is physically displayed as a two-dimensional plane of picture elements (pixels). The basic building block of an MPEG picture is the macroblock (MB) depicted in Figure 1. Each macroblock consists of a 16x16 array of luminance (grayscale) pixels and two 8x8-pixel chrominance (color) blocks. These three blocks actually cover the same picture area to represent its full-color and each 16x16 luminance block is processed as four 8x8-pixel blocks.

For content-based coding, MPEG-4 uses the concept of a video object plane (VOP). 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 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 of the video objects is transmitted by a separate bitstream of arbitrary-shaped VOPs. Once the VOPs, required for a visual scene composition are available in the receiver, the corresponding frame

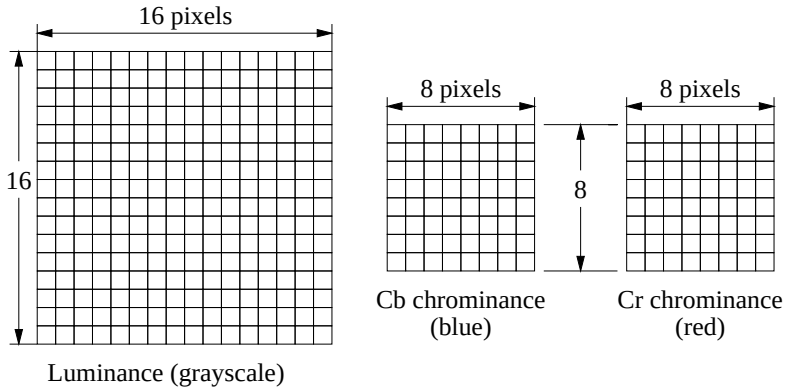

Fig. 1. The MPEG Macro Block

is reconstructed. The concept of the frame composition in MPEG-4 is sketched in Figure 2.

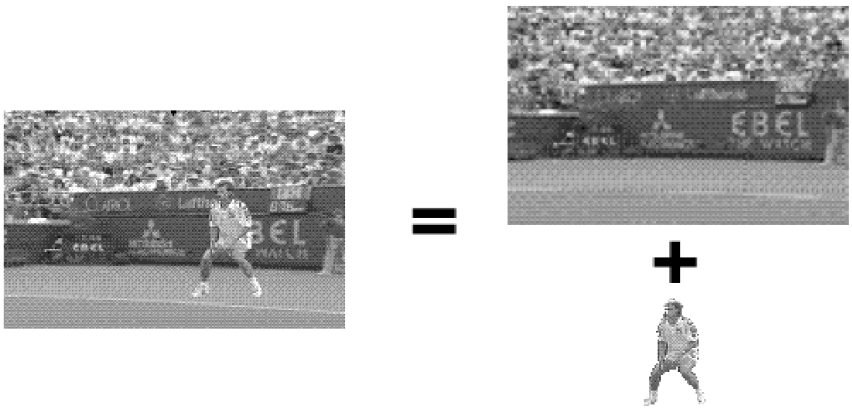

Fig. 2. Frame composition in MPEG-4

To distinguish an object from the background and to identify the borders of a VOP, MPEG-4 defines *shape* of an object. Shape information is provided in binary or grayscale format. The binary format represents the object shape as a pixel map, which has the same size as the bounding rectangular box of the VOP. Each pixel from this bitmap takes one of two possible values, which indicate whether a pixel belongs to the object or not. The binary shape representation of a VOP is referred to as *binary alpha plane*. This plane is partitioned into 16x16 *binary alpha blocks* and each binary alpha block is associated with

the macroblock, which covers the same picture area. In the grayscale shape format, each pixel can take a range of values, which indicate its transparency. The transparency value can be used for different shape effects (e.g.,blending of two images).

3 The Addressing Problem in MPEG

Video information is represented as a scanned sequence of pixels from a two dimensional visual plane. In digital video systems, this information is usually stored into linearly addressable memories and displayed later as two-dimensional frames. In MPEG standards, this information is processed and modified between the scan and display phases. Most of data processing in these standards, however, is not performed over pixel sequences, but over certain regions (blocks of pixels) from a frame, and this arises some problems with data alignment and accessibility into systems memory. To illustrate these problems, let us take the following example. Let us assume a linearly addressable memory and a pixel plane divided into blocks with dimensions 2x2, where each pixel is represented by a byte (see Figure 3). In linear addressing spaces the basic addressable units are bytes and words. We store video information in a conventional scan-line manner and we want to access the pixel block containing pixels 1, 2, 11 and 12. This pixel block (32 bits of information) is not aligned into consecutive memory locations (see Figure 3b) and we can not access it by a single memory transfer even if we can transfer a 32-bit word per memory cycle. This may lead to delays in data processing since the processor would have to wait for the whole data delivery. For MPEG-1,2 we may tune the memory system to pack the right bytes into a word, since the scan line length is a constant, equal to the width of the frame. In MPEG-4, however, this would not be so simple. The reason is that the scan line length is equal to the width of the VOP, which in turn may take any arbitrary value.

The other extreme approach to access block-organized data, stored into linearly addressable memory space is to reorder data. If we store each block of pixels into consecutive bytes (Figure 4), we will be able to access the whole required information for these blocks in a single memory cycle (e.g., block with pixels 1, 2, 11 and 12). In MPEG standards, however, some of the most demanding algorithms (e.g., motion estimation) do not process only the blocks, set into the original block grid. These algorithms also require to access block data with arbitrary position in the frame. In such cases the block-oriented reordering would not help for accessing the right piece of data. For example block containing pixels 12, 13, 22 and 23 (Figure 4) can not be accessed in a single memory cycle, not even in two cycles, because its pixels are scattered through the memory.

These two examples show that different memory organizations and addressing modes are vital for data processing speed-up of MPEG architectures and their implementations.

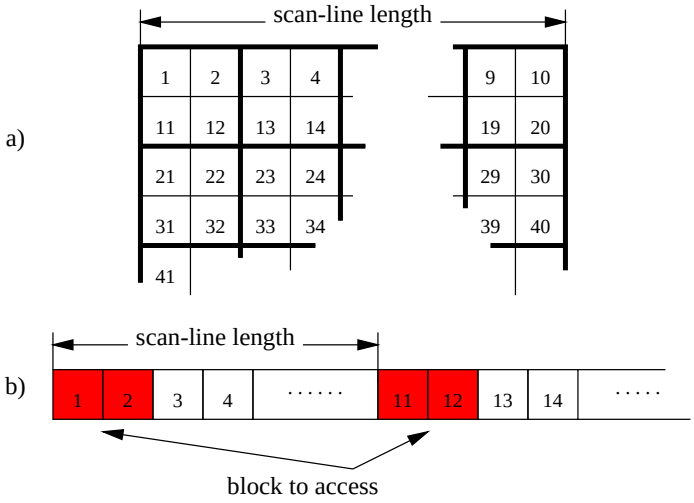

Fig. 3. Video Data Alignment into Linearly Addressable Memory: a) Pixels in a Video Frame; b) Block Access in a Scan-Line Aligned Data Memory.

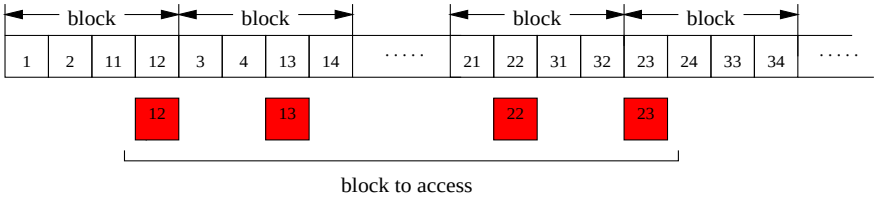

Fig. 4. Block-based Alignment of Visual Data into Linearly Addressable Memory

4 Related work

In [14], the three common addressing paterns for vector processors are described. They are classified as *sequential, regular* and *submatrix* accesses. The submatrix access is, in essence, a two-dimensional addressing in a square vector array with firmly defined dimensions.

In [16], Park develops the ideas from [4][15][17][19] 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 processor, adapted to conventional motion-estimation algorithms is proposed in [6]. 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. This processor is too specialized and requires additional data reordering.

An extensive exploration in memory management and organization for MPEG-4 encoders is reported in [3, 18]. However the focus is in the field of low-power consumption. The proposal combines 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 (e.g., without changing the memory bandwidth).

Multiprocessor video processing systems with distributed embedded DRAM are discussed in [9, 10]. The DRAM and local SRAM of the systems are distributed to multiple processor nodes. The integrated DRAM is primarily used as frame buffer. Loading and storing operations between local SRAM and DRAM are controlled by a DMA controller, capable of addressing rectangular image portions. A mechanism for block oriented data transfer between the processor nodes is also discussed. The memory organization is not designed for co-existence with a general-purpose processor (GPP) and is not intended for an FPGA implementation.

In [1], some instruction set extensions aiming at MPEG-4 video are proposed. New instructions are proposed for block-level processing, bitstream parsing, shape processing and padding. A VLSI MPEG-4 codec, called M-PIRE, was developed and its macroblock engine described in [8]. The same paper emphasizes on the instruction set discussion as well.

In this paper we differentiate with previous proposals in one or more of the following:

- We define the architectural aspects of a universal, reconfigurable data storage, dedicated for block-organized visual data (differentiates from [6, 16]).
- The storage is compatible with any general-purpose architecture and is suitable for an implementation in a custom computing machine - a hybrid between GPP and FPGA(differentiates from [9, 10]).
- The definition allows implementations with higher data throughput (differentiates from [3, 18]).
- This storage should be utilized by reconfigurable accelerators, supporting important multimedia instructions. We implement the ACQ instruction for the first time (therefore not included in [1, 8]).

We also differentiate from all previous proposals (including [14]) in defining the two-dimensional addressing over a two-dimensional data storage with variable dimensions.

5 The Two-Dimensional Addressing

Visual information has a two-dimensional structure, so the most natural approach for accessing it is to address a two-dimensional memory space. Since the

basic unit being processed in MPEG is the pixel block, we can assume an addressing space with two dimensional logical organization, where the basic addressable units are blocks. Figure 5 depicts an abstract design model of the proposed idea. The interconnect network is responsible for routing the right data block from the memory array to the processing units.

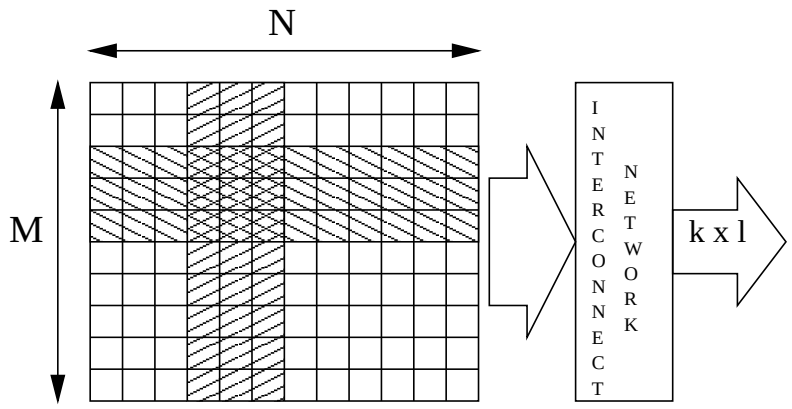

Fig. 5. An Abstract Design Model of a 2D Addressable Memory

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 pixel with coordinates i,j in the addressable area;
$0 \leq i, k < M, 0 \leq j, l < N$;
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 shows that the 2D address of a block is the same as the 2D-coordinates of its upper-left-most pixel in the addressable area. 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 5. The graphical representation of the two dimensional block addressing is depicted in Figure 6.

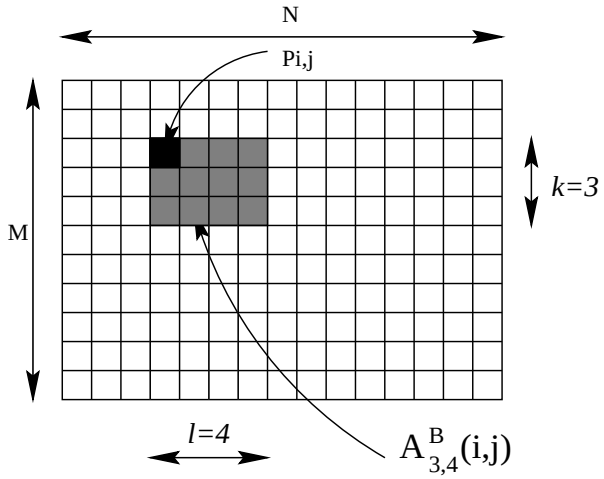

Fig. 6. 2D Block Addressing

The definition does not restrict the dimensions of the addressable area (M x N). These dimensions can take any value, depending on the application being implemented. This is very important for the implementation of MPEG-4, since each VOP has an arbitrary shape and size. Furthermore, each MPEG algorithm requires different memory amount (e.g., search area for motion estimation or a whole frame). 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. The latest FPGAs of Xilinx have up to 3.5 Mb true dual-port RAM on-chip [21], allowing reconfigurable implementations of 2D addressable storages for up to 4 frames or VOPs. 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.

6 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.

6.1 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 (PB) and this data structure is used by the criterion for an accepted quality. A dedicated function called ACQ is defined in [12]:

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*

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

where

$$acq_i = \begin{cases} 0 \;\; if \;\; SAD\_PB_i > 16 * alpha\_th \\ 1, \; otherwise. \end{cases} \tag{2}$$

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

The ACQ function shows whether the encoding ($BAB^{'}$) of a certain BAB gives an accepted quality result according some specified lossy coding conditions. These conditions are formally determined by the alpha threshold value. Figure 7 shows the influence of the *alpha_th* parameter on the appearance of an encoded VOP. The higher the *alpha_th* value is, the lower the acceptable quality of the encoding is. If *alpha_th=0*, then encoding will be lossless (with the highest visual quality).

We can represent SAD_PB_i as follows:

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

where $P_{i.16+j}$ and $P^{'}_{i.16+j}$ are the *binarized* values of the j^{-th} pixels from PB_i and $PB^{'}_i$ respectively and a value of *0 represents a transparent pixel* while a

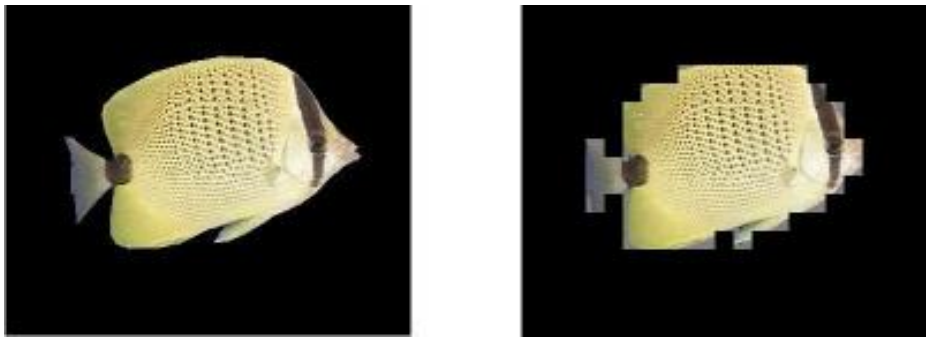

Fig. 7. Alpha threshold influence on the VOP visual quality: left - alpha_th=0; right - alpha_th=256

value of *1 - an opaque one*. According to these assumptions, we can substitute the absolute difference in (3) with a *xor* operation:

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

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

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

According to Definition 2 and Equation (4):

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

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

and

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

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 $\frac{256}{255} \approx 1$:

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

The implementation of Equation (7) is depicted on Figure 8. We can assume the discussed structure as a basic processing element (PE) and (taking into account Equation (6)) we can build the systolic processor shown on Figure 9.

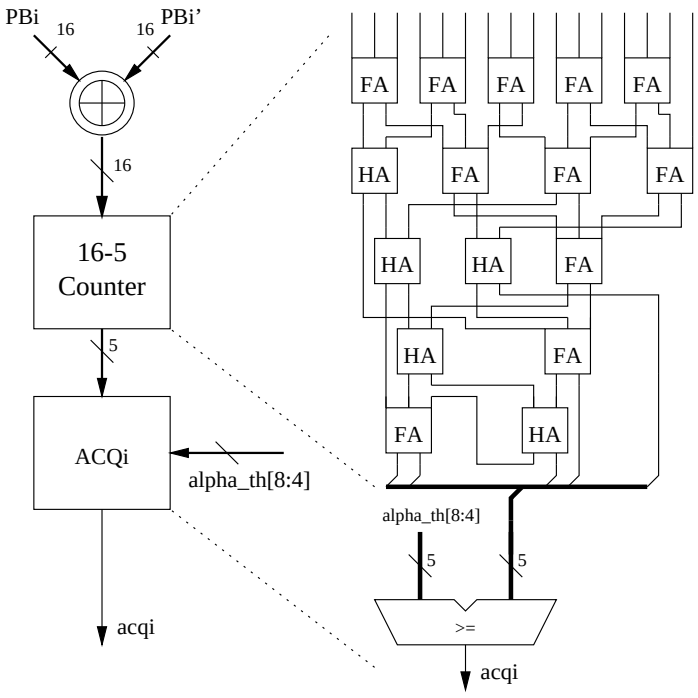

Fig. 8. Accepted quality single pixel-block processing element

6.2 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 throughput 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/cyc for a single BAB. Figures 8 and 9 show the two extreme cases - a pixel 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.

7 Evaluation

To evaluate the proposed structure of the ACQ function accelerator, 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

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

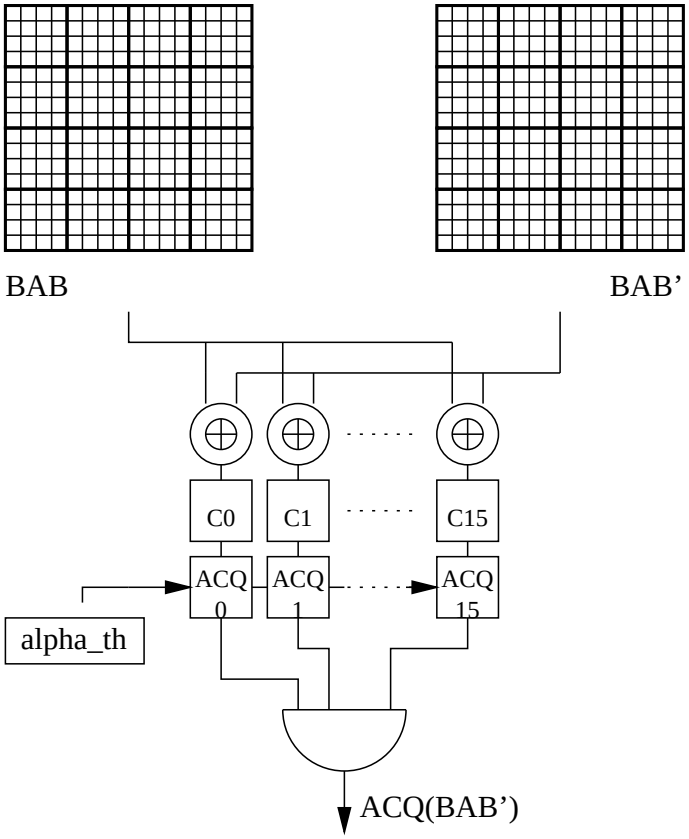

Fig. 9. The ACcepted Quality processing structure

synthesized for Altera FPGA. The reference software for the evaluation of the structure was Altera Max+Plus II. 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 1 suggests the processing latency and memory bandwidth, required for different number of processing elements in an Altera FPGA. Besides the operating latency, we use another measurement for the speed of the unit in terms of processed data units per time unit. In the proposed structure the basic data units are BABs and we achieve a speed of up to 16 129 032 BAB/s. Since there is a macroblock corresponding to any BAB and the macroblock processing speed is defined in the MPEG-4 profiles [11], we can use our results to estimate the real-time operating capabilities of the circuit. For the core and main MPEG-4 profiles, the required real-time rates are 23 860 MB/s and 97 200 MB/s (macroblocks per second) respectively. These numbers are well below our simulation results and, assuming that a macroblock manipulation involves a BAB processing as well, it is evident that the proposed ACQ

Table 1. Processing speed and required data bandwidth according to the number of processing elements (for Altera FPGA)

Number of PE	Processing latency, ns	BAB/s	Data 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

engine can easily 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. The evaluation assumptions are described bellow:

- As a simulator, we have used the *sim-outorder* from the SimpleScalar Toolset (version 2.0) [5]. The base machine has a super-scalar MIPS architecture and comprises of the following units: 4 integer ALUs, 1 integer MULT/DIV unit, 4 FP adders, 1 FP MULT/DIV-unit, and two memory ports.
- We simulated the MOLEN machine organization, adopted from [20].
- The benchmark software we used was the MPEG-4 VM of the European ACTS Project MoMuSys [7].

To utilize the ACQ instruction in the MPEG-4 encoder, we have modified its source code by including some assembly calls to the *configure* and *execute* instructions as defined in MOLEN [20]. However, we haven't modified the compiler of the SimpleScalar Toolset to model the inclusion of the new instructions. Instead, we have used the instruction annotation feature of the sim-outorder simulator. When annotating the new *configure* and *execute* instructions, we have taken into consideration the timing of the original architecture. The timings of the FPGA reconfiguration and the ACQ instruction execution have also been included into the simulation model. We simulated the MOLEN machine organization with a reconfigurable ACQ extension and the simulation results indicated up to 10% faster performance, while running the shape-encoding part of the core MPEG-4 profile. With the reported data, however, we can indirectly (by means of an instruction) estimate the performance gains of the two-dimensional block addressing. In this particular evaluation, the performance acceleration is a result both of the two-dimensional addressing and the high computational power of the ACQ instruction.

8 Conclusions and future work

In this paper we discussed the problem of allocating visual data in MPEG standards with respect to the efficient data access and processing. To deal with this

problem, we made three new definitions for a potential MPEG architecture: a two dimensional data storage, a block data type and a new addressing function. We showed that the introduced addressing mode, referred to as two-dimensional addressing, is feasible in MPEG-4. A vector instruction ACQ utilizing the new memory access was proposed and its scalable implementation was investigated. We achieved a considerable processing speed-up, because:

- the two-dimensional addressing is more suitable and faster than conventional addressing schemes when applied over block organized visual data;
- we defined and implemented a new function that plays a key role in the investigated class of algorithms.

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

Memory addressing implementation. A fast and cost-effective implementation of the two-dimensional addressing will make the benefits of this mode stronger.

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

9 Acknowledgements

This research is supported by PROGRESS, the embedded systems research program of the Dutch organization for Scientific Research NWO, the Dutch Ministry of Economic Affairs, the Technology Foundation STW (project AES.5021) and PHILIPS Research Laboratories, Eindhoven, The Netherlands.

References

1. M. Berekovic, H.-J. Stolberg, M. B.Kulaczewski, P. Pirsh, H. Moler, H. Runge, J. Kneip, and B. Stabernack. Instruction set extensions for MPEG-4 video. *Journal of VLSI Signal Processing*, 23(1):27–49, October 1999.
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, December 1971.
5. D.C.Burger and T.M.Austin. *The simpleScalar Tool Set, Version 2.0. Technical Report CS-TR-1997-1342.* University of Wisconsin-Madison, 1997.
6. 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, February 1996.

7. G.Heising and M.Wollborn. *MPEG-4 video reference software package.* ACTS AC098 mobile multimedia systems (MOMUSYS), December, 1999.
8. 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.
9. K. Herrmann, S. Moch, J. Hilgenstock, and P.Pirsch. Implementation of a Multiprocessor System with Distributed Embedded DRAM on a Large Area Integrated Circuit. In *IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT)*, pages 105–113, October 2000.
10. J. Hilgenstock, K. Herrmann, and P.Pirsch. Memory Organization of a Single-Chip Video Signal Processing System with Embedded DRAM. In *9-th Great Lakes Symposium on VLSI*, pages 42–45, March 1999.
11. ISO/IEC JTC11/SC29/WG11 W2502. ISO/IEC 14496-2. Final Draft International Standard. Part2: Visual, Oct. 1998.
12. ISO/IEC JTC1/SC29/WG11 N3312. MPEG-4 video verification model version 16.0.
13. ISO/IEC JTC1/SC29/WG11 N4030. MPEG-4 Overview - (V.18 - Singapore Version), March 2001.
14. P. M. Kogge. *The Architecture of Pipelined Computers.* McGraw-Hill, 1981.
15. D. H. Lawrie. Access and alignment of data in an array processor. *IEEE Transactions on Computers*, C-24(12):1145–1155, December 1975.
16. J. W. Park. An efficient buffer memory system for subarray access. *IEEE Transactions on Parallel and Distributed Systems*, 12(3):316–335, March 2001.
17. 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, August 1996.
18. 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 January 2000.
19. D. C. van Voorhis and T. H. Morrin. Memory systems for image processing. *IEEE Transactions on Computers*, C-27(2):113–125, February 1978.
20. S. Vassiliadis, S. Wong, and S. Cotofana. The MOLEN rm-coded processor. In *11th International Conference on Field Programmable Logic and Applications (FPL)*, 2001.
21. XILINX. Virtex-II Platform FPGA Handbook, December 2000.