An Implementation of the MPEG-4 ACQ Function

Georgi Kuzmanov, Stamatis Vassiliadis, Jos van Eijndhoven*

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— **An important new feature in MPEG-4 is the *ACcepted Quality Function* (ACQ), which determines whether the encoding of the shape of an object gives an accepted quality according to some specified lossy coding conditions. In this paper, we propose a hardware implementation of the ACQ function as a *systolic structure of processing elements*. The structure is scalable and can be implemented according to different memory bandwidth restrictions. 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 and the results indicate that the proposed structure can easily meet the real-time requirements with its operating latency of 62 ns. The structure can support the ACQ function as an instruction in an ISA extension of a general purpose processor. The ACQ function can be implemented either as a reconfigurable or as a hardwired unit in a dedicated MPEG-4 processor.**

Keywords— **MPEG-4, ACQ, Visual Object, Systolic Structure, ISA extension**

I. Introduction

MPEG-4 aims at providing description of tools and algorithms for efficient storage, transmission and manipulation of video data in various multimedia environments [8], [9]. The approach, taken by the Motion Pictures Experts Group (MPEG) in this standard, relies on the *content-based coding*, which, combined with various new functionalities, makes MPEG-4 radically different from its predecessors. For content-based coding, this new standard 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 is reconstructed. 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* (BAB) 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).

MPEG-4 imposes new, higher computational requirements that already exceed the capabilities of current general-purpose architectures[1]. One possible approach to improve the computational power of a general-purpose architecture is to utilize new, specialized instructions as instruction set extensions [1], [12], [13]. In this paper, we have implemented the new instruction "ACcepted Quality" (ACQ), which supports the identically named function in MPEG-4. The ACQ accelerator 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 for microarchitectural simulations, we reduce the calculation of the ACQ

[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].

function to 62ns, allowing performance gains of the shape encoder by up to 10%.

The remainder of this paper is organized as follows. In Section II, the ACQ function is described and its implementation is proposed. The same section discusses the scalability of this implementation and the data bandwidth it requires. Section III contains an evaluation of the proposed structure and reports the results. Finally, the conclusions of this paper are included in Section IV.

II. The Accepted Quality Function

Previous research in [3][4][6][10] indicate that after motion estimation, the next computationally most demanding algorithm in MPEG-4 is the shape encoding. An essential part of the shape encoding process in MPEG-4 is the necessity to ascertain whether the encoded BAB has an accepted visual quality under some specified lossy coding conditions. To achieve this, 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 *accepted quality function* (ACQ) is defined in [8]:

Definition 1 *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*

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

where

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

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 determines whether the encoding ($BAB^{'}$) of a certain BAB gives an accepted quality result according some specified lossy coding conditions. These conditions are formally included in the alpha threshold parameter. Figure 1 depicts 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).

The ACQ function is intensively used in three basic procedures of the MPEG-4 encoding process, namely:

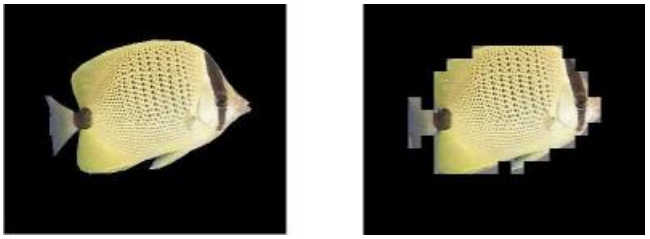

Fig. 1. Alpha threshold influence on the VOP visual quality: left - alpha_th=0; right - alpha_th=256

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 and there exist seven modes to code the shape information of each macroblock [8].

Motion estimation and compensation for shape. This 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 full-color video frames, but in MPEG-4 it is 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 [8].

A. Implementation

To implement the ACQ function we make some mathematical manipulations first. Let us represent SAD_PB_i as follows:

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

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 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 pixel blocks.

According to Definition 1 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 1, $alpha\_th * 16 = alpha\_th_5 * 256$, 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 2. 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 3.

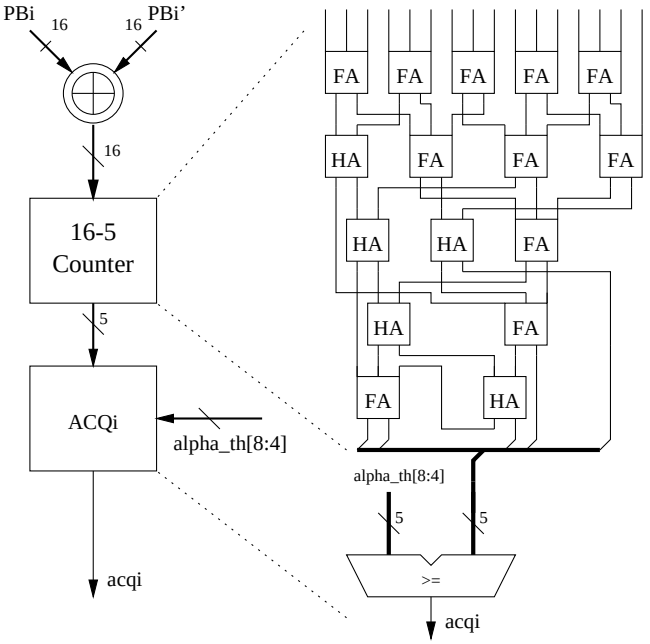

Fig. 2. Accepted quality single pixel-block processing element

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 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 2 and 3 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 an on-chip memory buffer with a suitable organization for the ACQ engine, we will be able to achieve higher data throughput.

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

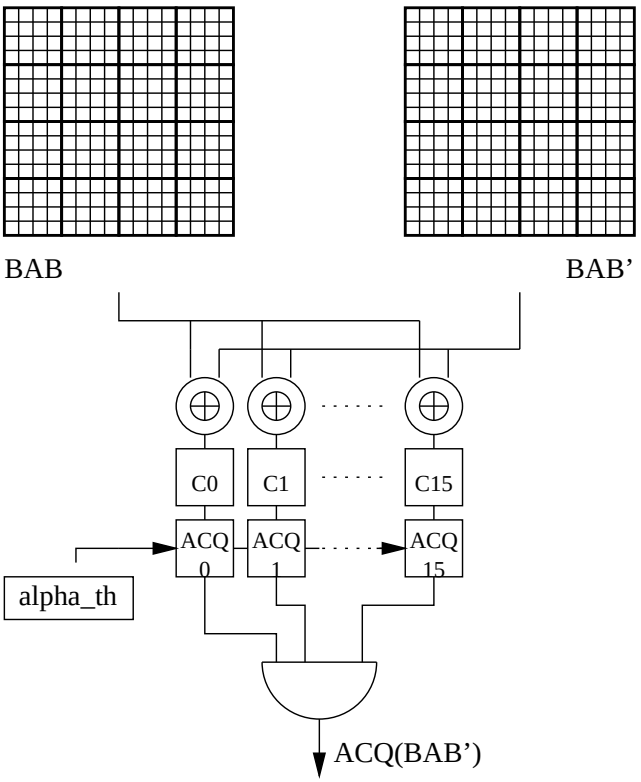

Fig. 3. The ACcepted Quality processing structure

III. 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 synthesized for Altera FPGA. The reference software for the evaluation of the structure is 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 I suggests the processing latency and memory bandwidth, re-

TABLE I

PROCESSING TIME AND REQUIRED DATA BANDWIDTH ACCORDING TO THE NUMBER OF PROCESSING ELEMENTS (FOR ALTERA FPGA)

Number of PE	Processing time in 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

quired for different number of processing elements in an Altera FPGA. Besides the operating latency, we use another measurement for the speed of the engine in terms of processed data units per time unit. In the proposed engine, 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 [7], 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 to process 16 and 32 video objects 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 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. As such a simulator, we have used the *sim-outorder* from the SimpleScalar Toolset (version 2.0) [5] to simulate the ACQ instruction as an instruction set extension of a superscalar MIPS architecture. The simulated machine organization has been adopted from [11] and the simulation results indicate up-to 10% faster performance, while running the MPEG-4 shape-encoding algorithm.

IV. CONCLUSIONS

In this paper we proposed an implementation supporting the MPEG-4 ACQ function as an instruction set extension of a general-purpose processor. The designed scalable structure has been modeled in VHDL and synthesized for an Altera FPGA. Simulation results indicate capabilities of processing up to 16 129 032 BAB/s and performance gains of up to 10%. An implementation of the ACQ unit either as a hardwired, or as a reconfigurable accelerator of a general purpose architecture would benefit the real-time implementation of an MPEG-4 shape encoder.

V. 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] 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.

[4] 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.

[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] 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.

[7] ISO/IEC JTC11/SC29/WG11 W2502. ISO/IEC 14496-2. Final Draft International Standard. Part2: Visual, Oct. 1998.

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

[9] ISO/IEC JTC1/SC29/WG11 N4030. MPEG-4 overview, March. 2001.

[10] S. Vassiliadis, G. Kuzmanov, and S. Wong. MPEG-4 and the New Multimedia Architectural Challenges. In *15th International Conference SAER'2001*, St.Konstantin, Bulgaria, 21-23 September 2001.

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

[12] S. Wong, S. Cotofana, and S. Vassiliadis. Multimedia Enhanced General-Purpose Processors. In *International Conference on Multimedia and Expo*, New York City, NY, USA, 2000.

[13] S. Wong, S. Cotofana, and S. Vassiliadis. Coarse Reconfigurable Multimedia Unit Extension. In *9th Euromicro Workshop on Parallel and Distributed Processing PDP 2001*, pages 235–242, Mondova, Italy, 2001.