MPEG-4 and the New Multimedia Architectural Challenges

Stamatis Vassiliadis, Georgi Kuzmanov, Stephan Wong
Electrical Engineering Department, Delft University of Technology
P.O. Box 5031, 2600 GA Delft, The Netherlands
{S.Vassiliadis,G.Kuzmanov,J.S.S.M.Wong}@ET.TUDelft.NL

Abstract

The recent development of multimedia applications made them one of the most popular and most demanding types of workloads. To meet the new requirements and to map all new multimedia functionality onto systems with restricted resources a dramatical need of visual data compression standards arose. This paper discusses the performance requirements of the MPEG standards and outlines some approaches to meet these requirements. The focus is on the first representative of the latest generation visual data compression standards - MPEG-4. Its computational requirements and new architectural demands are analyzed. An innovative reconfigurable μ-architectural approach is presented as a new concept for a flexible and cost-effective implementation of multimedia processors. At the expense of three new instructions, the proposed mechanisms allow instructions, entire pieces of code, or their combination to execute in a reconfigurable manner. These three instructions are proposed as an ISA extension of a super-scalar processor to illustrate the advantages of this new concept.

Key words: MPEG-4, content-based encoding, data locality and reusability, ISA extension, reconfigurable μ-architecture, microcoded engines

1. Introduction

In the world of telecommunications, where channel bandwidth is the basic restriction for digital video transmission, coding of video data plays a key role for reducing the amount of information to be transferred. The industrial impact of the new digital technology and its growing economical importance urged the development of standards for digital video compression and outlined the basic requirement these standards should meet, namely - best possible visual quality at a given bitrate range. To develop these new video compression standards, the Moving Pictures Experts Group (MPEG) was formed under the patronage of ISO and IEC. 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 [1][5], does not only enable data transmission at very low bit rates (64 kbit/s). The inclusion of entirely new functionalities (e.g., content based coding, interactivity, combination between natural and synthetic scenes and objects) makes most of the specialists refer to MPEG-4 as to a new standard generation rather than as to the next MPEG version.

Digital video compression is mainly based on exploiting specific properties of the human visual system and reducing the redundancy in video data. The basic principles for the latter are spatial and temporal redundancy reductions.

In an MPEG picture, the basic building block is the macroblock (MB). The 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. The core method for visual data encoding in all MPEG standards is the two-dimensional Discrete Cosine Transform (2D DCT). It is performed over each 8x8 block and is used as a basic approach to reduce the spatial redundancy in a picture. Basically, the DCT decomposes data into discrete spatial frequencies, which can be processed in a manner, consistent with the properties of the human eye.

To exploit temporal redundancy, all MPEG standards adopt *motion compensation* techniques. Motion compensation is a process of coding differences (motion) between frames in a video sequence [6]. These differences are estimated as a displacement between pixel areas in the current frame (being encoded) and a previously encoded frame. The measurment of this displacement is the *motion vector*. A process, called *motion estimation*, is performed to determine the motion vectors for each macroblock. This process includes a search algorithm for best matching between the block to be encoded and an area of previously encoded frame. As a criterion for best block matching, the minimal Sum of Absolute Differences (SAD) function is usually used. The SAD sums all absolute differences between the corresponding pixels in two pixel blocks. The best matched pixel area in the reference picture is the one that minimizes its SAD with the current block.

The remainder of the paper is organized as follows. Section 2 briefly presents the basic functionality of MPEG-4 and its computational requirements. In section 3, we propose some approaches to meet the most crucial performance requirements of the MPEG standards. In section 4 we describe the concept of a reconfigurable μ-coded machine organization. Section 5, presents some simulation results in order to illustrate the benefits of the proposed concept in MPEG applications. Finally, section 6 gives some concluding remarks.

2. MPEG-4 - the content-based coding standard

MPEG-4 aims at providing description of tools and algorithms for efficient storage, transmission and manipulation of video data in various multimedia environments. The approach taken by the Experts Group relies on the *content-based coding*, which, combined with various new functionalities, makes MPEG-4 radically different from its predecessors. This approach contributes to more efficient compression and better visual quality at comparable bitrates. Furthermore, content-based representation of visual data gives the end user opportunities for interaction with the content of a visual scene.

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. Another important concept in MPEG-4 is the *sprite* panorama image. The sprite can be viewed as a background image, which is transmitted to the receiver only once. Later, for each consecutive frame only the camera parameters are transmitted and the receiver reconstructs the background image based on the sprite. The whole video frame is reconstructed after the moving foreground objects are also available in the receiver. Each of the video objects is transmitted by a separate bitstream of arbitrary-shaped VOPs. The concept of VOP + sprite frame composition in MPEG-4 is sketched in Figure 1.

Each VOP in MPEG-4 is defined by its *shape* and *texture*, which are coded differently. In MPEG-4, shape is used to distinguish an object from the background and to identify the borders of a VOP.

VOP Shape. The 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

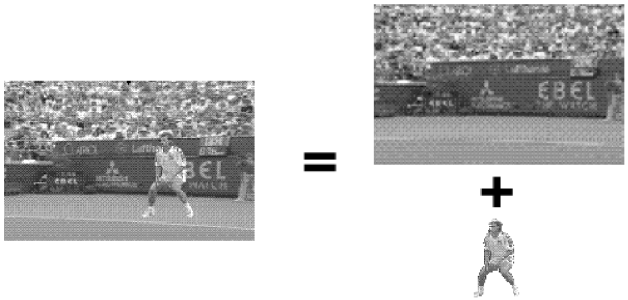

Figure 1: *VOP + Sprite frame composition in MPEG-4*

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).

VOP Texture. Texture encoding of a VOP macroblock is performed with respect to VOP's shape. There are three types of macroblocks in an arbitrary shaped VOP: macroblocks completely located inside the VOP, macroblocks containing VOP's boundary pixels and macroblocks entirely outside the VOP's boundary(transparent MB). Transparent macroblocks are discarded and not encoded and the internal macroblocks are processed by conventional 2D DCT. Boundary macroblocks are proposed with different techniques like shape adaptive DCT (SA-DCT) and low-pass extrapolation padding.

In MPEG-4, motion estimation is similar to MPEG-1/2 with some modifications. The most important new features in motion estimation algorithms for arbitrary shaped VOPs are the special padding technique and the agreement on a coordinate system. The purpose of padding is to ensure more accurate block matching by replacing the pixels outside the boundary of the VOP. In MPEG-4 an object can be anywhere in a video frame, so the *absolute frame coordinate system* is used for referencing the position and motion of all VOPs.

We have discussed scenes and objects of natural video, which are refferet to as *natural scenes and objects*. MPEG-4 also presents the option to combine synthetic scenes and objects with natural ones. The standard treats synthetic objects as a subset of computer graphics and includes *facial*, *body* and *2d mesh* animation which will not be discussed here.

Computational Requirements.

Most of the algorithms involved in MPEG-4 are data-intensive, meaning that a very large amount of data is processed with similar operations [3]. In order to get a better view of MPEG-4, we made profiling experiments and used data reported in papers [3][4][7]. Since MPEG-4 encoder and decoder differ in the requirements of their workload, we will distinguish them in our analyzis. Profiling results are shown in table 1 and 2 for the encoder and the decoder respectively.

kernel	**%**
Motion Estimation	88
Shape Encoding	7
DCT/IDCT	1
Motion Compensation	1

Table 1: *Profiling Results for MPEG-4 Encoder*

kernel	**%**
Shape Decoding	38
IDCT	16
Padding	13
Motion Compensation	11

Table 2: *Profiling Results for MPEG-4 Decoder*

The results show that most of the execution time in the MPEG-4 encoder is spent in *motion estimation*. We can safely claim that the basic efforts should be aimed at accelerating this kernel. The

next most demanding part of the encoding is shape processing. The padding process (less than 1% in the encoder), motion compensation and IDCT take negligible shares of the encoder processor cycles, but they become one of the main kernels in the decoder.

The importance of shape processing is much more evident in the MPEG-4 decoder, where it constitutes the largest part of the workload. In the decoding process we can also see that the IDCT and the padding already take reasonable parts of the computational requirements. Motion compensation is one of the most important processes in the MPEG-4 decoder as well.

Finally, we can conclude that the computational demands of the encoder and the decoder part of MPEG-4 significantly differ. We also note that shape processing is very important since it constitutes considerable parts of both the encoder and the decoder in MPEG-4.

3. Performance requirements of MPEG standards

The specific functionalities described in MPEG standards in most of the cases exceed the performance capabilities of the General Purpose Processors (GPP). Therefore, we obviously need new processors, capable to meet these requirements. To design such processors we can approach the problems from different points of view concerning the *architecture*, *implementation* or *realization* of the MPEG processor. For these three conceptual issues we use the terminology definition from [2].

Architecture. The architecture of any computer system is defined to be *the conceptual structure and functional behavior as seen by its immediate user*. The conceptual structure, e.g. data structures, and functional behavior, e.g. DCT, of a multimedia system is determined by the functional requirements of the corresponding multimedia application. In our case, considered hereafter, this application will be any of the MPEG standards.

Implementation. The implementation is defined to be the *logical organization of the dataflows and controls of a computer system*. This organization must perform the functionality as required from architecture level. The designer must design new units, come up with new organizations, incorporate a certain level of parallelism, utilize microcode to control the units.

Realization. The realization is defined to be the *physical structure embodying the implementation*. Examples of physical structures are CMOS transistors and Field-Programmable Gate Arrays (the discussion of the ralization issue is out of the scope of this paper).

The most crucial performance requirements, which have to be met by an MPEG system are *high computational power* and *enormous data memory bandwidth*. From the architectural point of view we can define some architectural issues that would help the implementation to meet these requirements. An architecture, entirely dedicated for the application field (MPEG standards), would obviously enable the implementation of a high performance specialized processor, but this is the most expensive solution. A less expensive and still effective approach appears to be the redefinition of an existing general purpose architecture.

To increase the computational power of the system we can define new instructions as an extension of the general purpose Instruction Set Architecture(ISA extension) [11][12]. We have to be careful, however, with the number of new instructions we define, since we may not be able to encode and implement them all within the fixed instruction format. Thus, we have to analyze the application domain and to extract functions or kernels that would effectively improve system performance when implemented as instructions. In MPEG 1,2 and 4, examples for such kernels are the chrominance and luminance encoding/decoding with the DCT/IDCT functions, and the motion estimation algorithm with the SAD function. In MPEG-4, the new functionality can be accelerated by defining new instructions in the shape encoding process or in padding and wavelet-based encoding.

Another important element of an architecture is its *basic data structure* (data type). If we carefully choose these structures, we can also achieve performance benefits. While in most GPP the data types

are bits, bytes and words, in MPEG standards the 8x8 *pixel block* can be defined as a basic data structure. In MPEG-4 the *binary alpha block* can also be referred to as a separate data type.

To solve the problem with the required enormous data throughput we can exploit two important properties of the data in MPEG, namely its *locality* and *reusability*. Let us define an architecture with frame/VOP data storages (for MPEG1,2/4), where the data immediately to be processed (frame/VOP) will be stored. As architectural issues, these storages will be displayed to the programmer. Consequently, a piece of the data is localized in the data storage and for multiple processing (reusability) of these data we need just one data transfer from the main memory. In MPEG, however, data is not allays processed in the order it is stored in the memory. Figure 2 shows the block data overlap in MPEG motion estimation algorithms, where we have to access data blocks that are different from the ones in the original block division of the frame or the VOP. In such a case, the implementation of a separate (block) addressing mode will help to speed-up the data transfers from data storages.

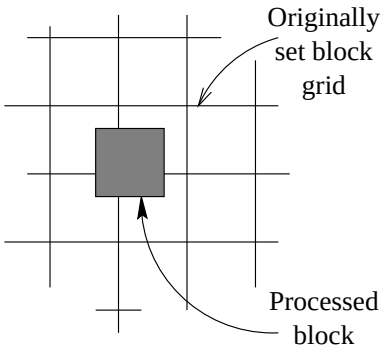

Figure 2: *Block Data Overlap in MPEG Motion Estimation*

Typically the MPEG standards do not state precisely all the algorithms that should be used to implement the described functionality. Furthermore, the implementations of some standard issues are optional (e.g., sprite encoding in MPEG-4) and sometimes they can not even coexist with other optional issues. Thus, different profiles (with different functionality) can be implemented within the same MPEG standard. We also showed in the previous section that MPEG-4 encoder and decoder have different computational requirements. Therefore, it is expensive and not effective, to implement into the architecture the whole standard functionality as hardwired circuits, since many of the available resources will not be utilized by a real application. To keep the implementation of an architecture at a reasonable cost-performance ratio, we can use the *reconfigurable* approach. We propose, by displaying means to maintain the reconfiguration at architectural level, to achieve a high flexibility in tuning the system for the specific application. The controllability of the reconfiguration process by only three new instructions is discussed in detail in section 4. Such an architecture can be implemented as a *custom computing machine*.

Custom Computing Machine. A *General Purpose Processor* (GPP) augmented with a Field-Programmable Gate Array(FPGA) is referred to as a *Custom Computing Machine*(CCM). The GPP (core processor) controls the execution and the reconfiguration, tuning the latter for specific algorithms or for the general-purpose paradigm. Such an organization can achieve orders of magnitude improvements in performance over a GPP alone, while preserving the flexibility of the programmable circuits over Application-Specific Integrated Circuits (ASIC).

4. Reconfigurable μ-architecture

This section discusses an innovative Custom Computing Machine organization, introduced in [9] and [10]. The reconfiguration of the hardware and the execution of the code on the reconfigurable

hardware is done in firmware via the ρ-microcode (an extension of the classical microcode to include reconfiguration and execution for resident and non-resident microcode). The microcode engine is extended with mechanisms that allow permanent and pageable reconfiguration and execution code to coexist.

The proposed machine organization is depicted in Figure 3. Instructions are fetched from the memory and stored in the instruction buffer (I_BUFFER). The ARBITER performs a partial decoding on the instructions in order to determine where they should be issued. Instructions that have been implemented in fixed hardware are issued to the core processor (CP). Instructions, dedicated for the reconfigurable part of the processor, are issued to the *reconfigurable unit*.

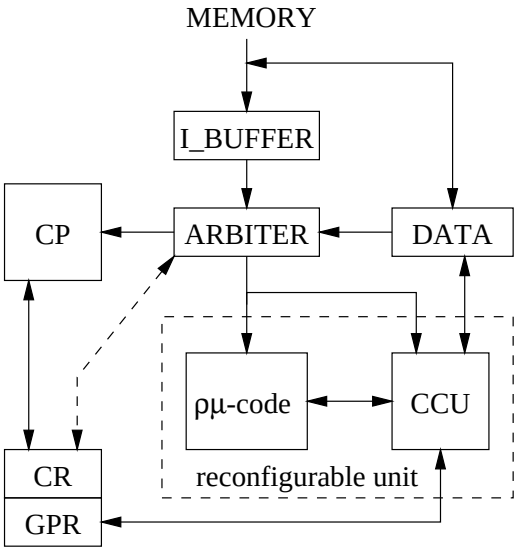

Figure 3: *The proposed machine organization*

The reconfigurable unit consists of a custom configured unit (CCU) and the $\rho\mu$-code unit. An operation, executed by the reconfigurable unit, is divided into two distinct phases: **set** and **execute**. The **set** phase is responsible for reconfiguring the CCU hardware enabling the execution of the operation. This phase may be divided into two subphases - partial set (**p-set**) and complete set (**c-set**). In the **p-set** phase the CCU is partially configured to perform common functions of an application (or group of applications). Later, the **c-set** sub-phase only reconfigures that blocks in the CCU, which are not covered in the **p-set** sub-phase in order to *complete* the functionality of the CCU. For the reconfiguration of the CCU, reconfiguration microcode is loaded into the $\rho\mu$-code unit and executed to perform the actual reconfiguration. The **execute** phase is responsible for the actual operation execution on the CCU, performed by executing a (resident) *execution microcode*. It is important to emphasize that both the **set** and **execute** phases do not specify a certain operation that has to be performed. Instead, the **p-set**, **c-set** and **execute** instructions directly point to the (memory) location where the reconfiguration or execution microcode is stored. This mechanism allows us to simply point to memory addresses, instead of specifying new instructions for the operations.

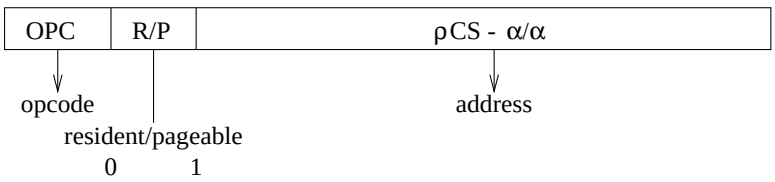

Figure 4: *The **p-set**, **c-set** and **execute** instruction format*

The instruction format of **p-set**, **c-set** and **execute** instructions is given in Figure 4. The opcode (OPC) specifies which instruction to perform. The R/P-bit specifies where the microcode is located and how to interpret the address field, i.e., as a memory address α (R/P=1) or as an address of the on-chip storage in the $\rho\mu$-code unit (R/P=0), indicated by ρCS-α. The address always points to the location of the first microcode instruction. The microcode is terminated by an *end_op* microinstruction.

5. The $\rho\mu$-coded processor and MPEG

We have evaluated the proposed scheme by cycle accurate simulations with MPEG-2 and MPEG-4 software encoders, used as benchmark multimedia applications. As simulator we have used the *sim-outorder* from the SimpleScalar Toolset. To achieve realistic results we have assumed the ALTERA chip family and the FPGA design software MAX+PLUS II and FPGA Express. We have described all implementations in VHDL to estimate their *total area* (in terms of LUTs) and the *number of cycles* an operation takes.

Cycle normalization. Since the cycles in a reconfigurable machine are slower than in a hard-wired machine, we use a *normalization* to determine the exact number of cycles, required for the reconfigurable part. For example, if we assume a super-scalar machine, clocked at 1 GHz and an FPGA, clocked at 200MHz, the number of cycles a reconfigurable instruction requires, should be multiplied by 5 in order to normalize this number to the super-scalar cycles. Further, we have considered for reconfigurable code the following operations: Sum of Absolute Differences (SAD) [8] and Two-Dimensional Discrete Cosine Transform (2D DCT). These operations have been chosen after analyzing the benchmark software.

Area and speed estimates: Table 3 presents the area and clock requirements for the implementations of the selected operations on an Altera APEX20K FPGA Chip. These implementations do not pretend to be optimal and were only used to provide us with a realistic estimation of the number of clock cycles for further simulations.

operation	area	clock cycles (frequency)	normalized cycles
16x16 SAD	1699 LUTs	39 (197 MHz)	234
16x16 multiply	1482 LUTs	12 (175 MHz)	69
32-bit adder	382 LUTs	5 (193 MHz)	21
2D DCT	using multiplier & adder		282

Table 3: *Area and speed estimates*

Simulation results: As it was mentioned before, the most computationally expensive algorithm, both in MPEG-2 and MPEG-4, is the motion estimation. A basic operation in ME is SAD, therefore its reconfigurable implementation is the biggest contributor to the overall performance gain. Simulation results show that we can gain up to 45% decrease in number of clock cycles, assuming instruction execution delay of 20 cycles. Even if the reconfigurable implementation takes 234 cycles(see Table 3) to perform the instruction, we still gain around 20% in performance. The DCT operation contributes less in the overall performance improvement keeping the overall speedup stable within 12%-13% for

the execution latency range 20 - 282 cycles. Finally, allowing the CCU to be reconfigured to either the DCT or the SAD implementation, we can achieve an overall execution time decrease of 32%.

6. Conclusions and future research directions

In this paper, we outlined some of the basic features of visual data coding standards, in particular MPEG-4. An analyzis of the performance requirements of recent multimedia applications and some directions to meet them were presented. We discussed the reconfigurable paradigm from a new, $\mu-$architectural point of view. A machine organization was proposed as an immediate implementation of the new reconfigurable concept. Simulation results show that such a $\rho\mu-$architecture can be successfully utilized by different algorithms in MPEG standards.

The demonstrated potentials of this concept open new research directions, related to its future utilization by different multimedia applications and standards. New specialized memory architectures and organizations, incorporated in reconfigurable systems will help for further increase of data throughput and more effective data processing. Defining application specific data types will cause the implementation of new instructions and a cost-effective speed-up of the systems. The identification of a sufficient number of complex functions and their implementation as reconfigurable microcoded engines will develop the potentials of the reconfigurable computing to make it a cost-effective alternative for future multimedia processing.

References

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

[2] G. Blaauw and F. 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] ISO/IEC JTC1/SC29/WG11 N4030. Mpeg-4 overview - (v.18 - singapore version), March. 2001.

[6] Y. Q. Shi and H. Sun. *Image and Video Compression for Multimedia Engineering.* Boca Raton CRC Press, 2000.

[7] 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.

[8] S. Vassiliadis, E. Hakkennes, J. Wong, and G. Pechaneck. The Sum Absolute Difference Motion Estimation Accelerator. In *24th Euromicro Conference*, Vasteras, Sweden.

[9] S. Vassiliadis, S. Wong, and S. Cotofana. The MOLEN rm-coded processor. Technical Report 1-68340-44(2001)-01, Computer Engineering Laboratory,TU Delft, the Netherlands, 2001.

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

[11] 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.

[12] 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.