A PROGRAMMABLE CO-POCESSOR FOR MPEG-4 VIDEO

M. Berekovic, H.-J. Stolberg, P. Pirsch,

Institut für Theoretische Nachrichtentechnik
und Informationsverarbeitung
Universität Hannover, Germany

H. Runge

Robert Bosch GmbH
Hildesheim, Germany

ABSTRACT

A programmable processor architecture for MPEG-4 video is proposed, that can serve as a coprocessor module in MPEG-4 decoder systems. It consists of a 64-bit dual-issue VLIW macroblock engine, a separate RISC core for bitstream parsing and system processing, and an autonomous I/O processor. A separate DSP is used for MPEG audio support. The architecture is fully programmable and supports parallelism on data-, instruction- and thread-level to cope with the high flexibility and processing demands of the MPEG-4 standard. The first implementation will support real-time decoding of MPEG-4 advanced simple profile or of MPEG-4 ACE-profile (CCIR601, single-object). Future designs will add support for object-based MPEG-4 functionalities. The paper focuses on the architecture, instruction set, and performance of the macroblock engine, which operates as an autonomous co-processor and carries most of the workload in MPEG-4 video processing. It has a RISC-based architecture with support for parallel processing of instructions and data. Special instructions are implemented with specific support for video processing.

1. INTRODUCTION

In contrast to its predecessors, MPEG-1 [1] and MPEG-2 [2], which focus on specific applications such as playback from CD-ROM or digital TV, the upcoming MPEG-4 [3][4][5] standard offers a standardized framework for a whole range of multimedia applications. Examples include teleshopping, interactive TV, internet games, or mobile video communication. MPEG-4 integrates different types of multimedia data and services by the introduction of an object-based approach for the description and coding of multimedia contents. Key aspects of MPEG-4 include, among others, independent coding of objects in a picture; the ability to interactively composite these objects into a scene at the display; transmission of 3D scene descriptions; temporal and spatial scalability; and improved error resilience.

As MPEG-4 targets a much broader range of different applications and bitrates than previously defined hybrid video coding standards like H.263 or MPEG-2, it employs a higher number of different algorithms and coding modes. Therefore, MPEG-4 implementations require a more software-oriented approach to be efficient. However, the total computational load for an optimized implementation of an MPEG-4 codec exceeds the performance levels of today's DSPs, making further hardware acceleration a necessity. For that purpose, we develop a new architecture which employs mainly three (optional four) independent processor cores. Each of them is optimized for the processing of specific data types, such as video, audio or stream processing. Video coding, the main computational load, is carried out by the macroblock engine, a 64-bit dual-issue VLIW core.

Section 2 gives an overview of the MPEG-4 standard. The proposed architecture is detailed in section 3 while in section 4 the architecture of the macroblock engine is presented. Section 5 concludes the paper.

2. THE MPEG-4 STANDARD

Current MPEG and ITU audiovisual codecs work frame and block based. At the sender site, video frames and audio are rendered, composed, coded, multiplexed and transmitted to the receiver. At the receiver site, the transport stream is demultiplexed, video and audio data are decoded, synchronised and presented as defined by the sender site. In contrast to that, an MPEG-4 scene consists of one or more audio-visual objects (AVOs) from multiple sources that are coded separately, using different coding tools for video, 3D graphics, speech or music. Thus, the composition of the final scene to be shown at the display is shifted from the studio (encoder) to the receiver (decoder) side [Fig. 1].

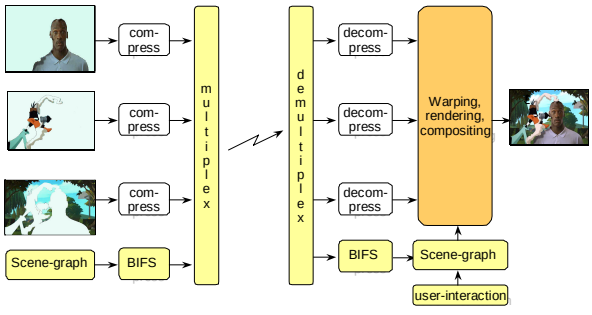

Fig. 1: MPEG-4 Coding Scheme

In MPEG-4, the algorithms used for coding of natural video are based on the block-oriented hybrid coding scheme, as they are known from MPEG-1 and MPEG-2, but were

extended to allow the coding of arbitrarily shaped video objects and 3D graphics objects. Further extensions were added for better coding efficiency (GMC, quarter-pel MC). For the use in error prone environments, error resilience features are addressed by several parts of the MPEG-4 standards. This makes MPEG-4 especially suitable for the use in wireless portable or mobile applications.

3. MPEG-4 DECODER

An MPEG-4 decoder chip is currently being developed by Bosch and University of Hannover. The application focus of the multimedia chip is on mobile and stationary real-time communication and interactive broadcast systems for mobile receivers. First silicon is expected for early next year.

3.1 Computational requirements

Due to its concept, MPEG-4 differs significantly from existing audiovisual coding standards in terms of its requirements on processing power, flexibility and memory bandwidth [6], [7].

First complexity assessments of the standard show that at least a decoder software implementation will be possible on advanced DSP or RISC processors for the simpler profiles and levels [8]. However, the computation requirements for video broadcast with high frame rates, full CCIR601 resolution, or for two-way real-time communication for mobile and portable applications, exceed the capabilities of current programmable processors. Furthermore, power-consumption, cost and size of the processing hardware are important parameters for mobile and portable applications. An optimised MPEG-4 processor platform therefore must combine the flexibility (i.e. programmability) required for the variety of different tools with a minimum of cost and power consumption of the implementation.

Table 1: Algorithmic function classes and their properties

	Algorithm example	Type of parallelism	Complexity, processing requirements	Data types
Stream Processing	Parsing Composition RLD, VLD	mostly sequential	high complexity non-word aligned processing	short (<16 bit int)
Macroblock processing	DCT/IDCT Filters ME, MC	data, instruction	low complexity high data bandwidth block oriented	short (8, 16 [32 bit] int)
Presentation processing	Compositing Rendering Graphics	data	medium complexity high data bandwidth irregular access	medium (16, 32 bit, int floating point)
Sequential word aligned processing	Audio Codecs CELP HVXC	data, instruction	medium complexity irregular data access	Varying (8 -24bit int, floating point)

One possible key to achieve this goal is the partitioning of the processor design into programmable units, which are optimised for a certain class of algorithmic functions within MPEG-4 [8]. An analysis of a word-width optimised MPEG-4 software reference model as a basis for the processor partitioning showed four classes of algorithms with similar properties in terms of complexity, processing requirements and parallelisation potential (Table 1).

Stream processing algorithms are control flow oriented with a high share of inherently sequential code, many interdependencies and non-word aligned processing. A generic RISC architecture with instruction set extensions for bit operations and code word transforms is a natural choice for this type of processing. Macroblock-based processing of video is still one of the most demanding parts of MPEG-4, especially due to the high throughput of image data and the addition of new algorithms. Programmable processor architectures with splitable ALUs have proven useful for coping with the bandwidth and processing requirements of these mainly regular, block oriented algorithms. Presentation processing, i.e., compositing of VOPs, includes overlay calculation, geometrical transforms, and a number of typical graphics algorithms, like texture mapping and bi- or trilinear interpolation. In the audio coding sector, processing consists of classical DSP algorithms, word aligned processing with a high share of multiply-accumulate operations and a high dynamic range. Due to the several alternative audio algorithms, the mapping on a conventional DSP structure is the most obvious solution.

3.2 Overall architecture

The algorithmic partitioning of MPEG-4 type processing is directly mirrored in the architecture of the MPEG-4 decoder chip (Fig. 2). The processor consists of three (optional four) independent, programmable processors. Each of them is optimised for one of the algorithmic classes.

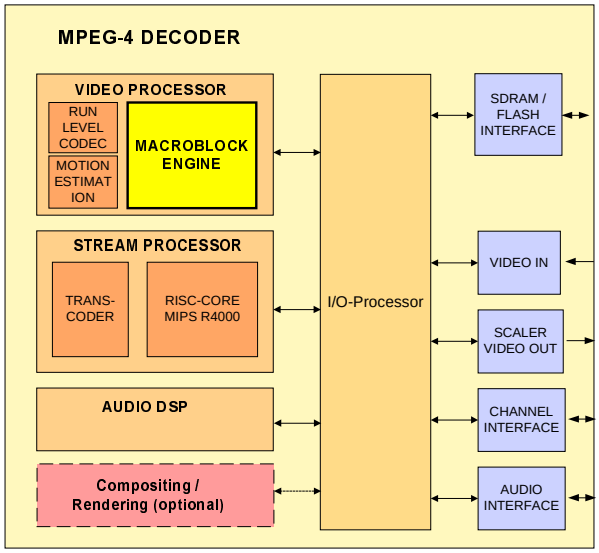

Fig. 2: Architecture of the proposed MPEG-4 decoder.

The stream processor consists of a standard RISC core with instruction and data cache. To achieve high performance for bit stream parsing and composition, variable length coding and decoding, a transcoder coprocessor has been added. The transcoder allows the transformation of bit segments of variable length into another code and also provides additional instructions for efficient bit manipulation. A standard DSP may be added for MPEG-audio processing. They use a shared-distributed memory system for storage and exchange of local data. The video processor is optimised for macroblock oriented algorithms. It consists of two hard-wired

units for motion estimation and run length coding and a vector-based, programmable macro block engine for all other macroblock-oriented algorithms. For the higher MPEG-4 profiles, a separate presentation processor for video compositing and rendering can be added. It will be part of a future implementation, where object-based funcionalities are targeted. All transfers to and from external memories (FLASH ROM and SDRAM) are scheduled by an autonomously operating I/O processor.

4. MACROBLOCK ENGINE

4.1 Data Path Architecture

The structure of the macroblock engine is characterized by the existence of two parallel data paths, the scalar and the vector data path [Fig. 3]. The scalar data path operates on 32- bit data words in a 32-entry register file and provides control instructions such as jump, branch, and loop. The vector data path is equipped with a 64-entry register file of 64-bit width. The 64-bits-wide arithmetic execution units in the vector path, e.g., MUL/MAC or ALU, incorporate subword parallelism by processing either two 32-bit, four 16-bit, or eight 8-bit data entities in parallel within a 64-bit register operand. The MUL/MAC unit even delivers a 128-bit result by writing back two 64-bit registers, thus preserving the full precision of the computed result. This way four 16-bit multiplications with 32-bit accumulation can be performed in parallel. With its support for subword parallelism, the vector data path is particularly suited to process the repetitive operations of typical macroblock algorithms at high throughput.

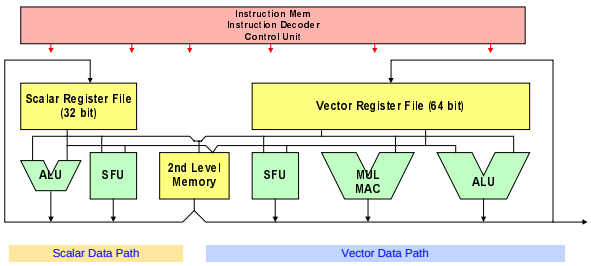

Fig. 3: The two data paths of the macroblock engine. Macroblock engine. Special function units (SFU) provide instructions characteristic for MPEG-4.

The macroblock engine's parallel data paths are controlled by a dual-issue 64-bit VLIW(very long instruction word). By default, the first slot's instruction is issued to the vector path, and the second one is issued to the scalar path, enforcing parallel execution. However, also two vector or two scalar instructions can be paired within a VLIW. With four read/two write ports on the vector register file, even two vector instructions can execute in parallel provided they do not belong to the same instruction group (i.e., are not executed on the same hardware unit). If parallelization is not possible, the instruction decoder autonomously serializes the execution of instructions. The flexible utilization of the VLIW minimizes the number of void instruction slots and promotes code density.

Instructions and data are supplied to the macroblock engine via local memories, which are accessible within a single clock cycle. Transfers between external RAM and local memories are performed in the background by programmed DMA as the program execution continues. For the core video algorithms, scalar and vector path can operate in concert to exchange blocks of video data between the second-level memory and the vector register file without stall cycles by performing parallel stores and loads.

4.2 Instruction Set

The instruction set of the macroblock engines falls into two parts: Scalar instructions and vector instructions. Only vector instructions support subword parallelism. Common features of both instruction types include signed/unsigned interpretation of operands and saturation/modulo arithmetic variants where useful.

The instruction set can be divided into the following groups: arithmetic, logic, shift&round, min/max/clip, data formatting, program control, and data transfer. The arithmetic group includes, among others, the MUL/MAC operation with multiple precision. The RND instruction combines an arbitrary shift with different rounding modes as defined by MPEG-4. MIN, MAX, and CLIP instructions eliminate comparisons and associated branches in frequent video operations. The data formatting group comprises instructions for arbitrary permutations of data fields.

The vector data path supports parallel processing on subwords. Originally, SIMD-style processing of data types packed into a single word assumes the same operation to be applied uniformly to all data items. Whenever subword data items are to be treated differently, program execution has to be serialized, resulting in a loss of performance. This limitation can be removed by the introduction of field-wise conditional execution, that allows to sustain the SIMD scheme even for data-dependent processing. A set of condition flags is available for each data field in the vector path, whose state is determined by the execution of a previous instruction (with the explicit permission to modify the flags). A subsequent conditionally executed instruction then calculates the result of an operation in dependence on the state of the condition flags.

Figure 4 shows the example of a conditional move instruction, which moves the contents of a sub-field depending on the status of a certain conditional bit that has been set individually for each data field by previous operations. Further instructions have been included that are useful only in conjunction with the conditional execution mode; the ADDSUB instruction, e.g., performs an addition for the 'true' case and a subtraction for the ' false' case.

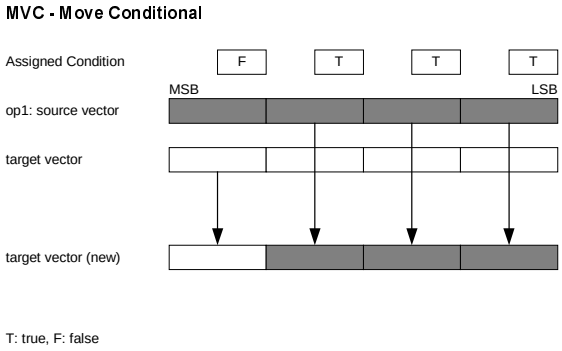

Figure 4. Conditional move instruction (MVC).

4.3 Performance Estimation

The macroblock engine has been implemented in VHDL and synthesis results suggest that it will consume an area of appr. 5mm^2 and achieve a 133 MHz clock on a 0.18µ CMOS technology. Implementation of MPEG-4 ACE Profile (single VOP, CCIR601, 2Mbit/s) is in progress. Core algorithms to be implemented include DCT, motion compensation and postprocessing. First performance estimates are shown in Table 2. The left side indicates the number of clock cycles needed for the algorithm, while the right side denotes the number of cycles needed to decode one second of the bitstream. The 2D-IDCT for example, consumes 190 clock cycles for a complete 8x8 transform. Motion compensation requires about 91 cycles per block (full-pel). Postprocessing needs 377 cycles. The overall cycle-count shows that real-time decoding of ACE profile CCIR601 video is feasible.

Table 2: Performance estimates for an MPEG-4 ACE profile decoder (single object, CCIR601@2Mbit/s)

algorithm	#cycles/block	#Mcycles/s
IDCT (DC only)	21	
IDCT (full)	190	
IDCT		11
MC (full-pel)	91	
MC (overall)		53
Post-Processing	377	65
other		4
Overall		133

5. CONCLUSION

With it's object oriented, generic approach and a wide range of profiles and levels, the emerging MPEG-4 video coding standard covers a much broader range of real time communication and broadcast applications than previous hybrid coding standards. This flexibility results in a much higher algorithmic complexity and significantly increased demands on processing power. The proposed MPEG-4 multimedia processor responds to these demands by integrating three programmable cores and a VLIW macroblock engine on a single chip, where each unit is adapted to the specific processing requirements of one algorithmic function class of MPEG-4. The 64-bit dual issue VLIW macroblock engine is optimized for the processing of high-throughput video data and carries most of the workload of the chip. It has been extended with special instructions that speed-up execution of MPEG-4 video algorithms. The processor runs at 133 MHz and is capable to decode MPEG-4 ACE profile video in real time (single object, CCIR601, 2Mbit/s).

Acknowledgements

The described work was partly funded by the German Bundesministerium für Wirtschaft und Technologie (BMWi) and was partly carried out within the framework of the MEDEA A116 project (M4M-MPEG fo(u)r Mobiles).

6. REFERENCES

[1] ISO/IEC 11172-1/-2/-3, Information Technology, "Coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbit/s: Systems/Video/Audio," 1994.

[2] ISO/IEC 13818-2, "Generic coding of moving pictures and associated audio, (MPEG-2), Part2: Video," Nov. 1993.

[3] ISO/IEC JTC11/SC29/WG11 N2323, "Overview of the MPEG-4 standard," July 1998.

[4] ISO/IEC JTC11/SC29/WG11 W2502, "ISO/IEC 14496-2. Final Draft International Standard. Part 2: Visual," Atlantic-City, October 1998.

[5] L. Chiariglione, "Impact of MPEG Standards on Multimedia Industry," *Multimedia Signal Processing, Proceedings of the IEEE*, Vol. 86, No. 6, pp. 1222-1227, June 1998.

[6] M. Berekovic, H.-J. Stolberg, M. B. Kulaczewski, P. Pirsch,. H. Runge, H. Möller, J.Kneip, "Instruction Set Extensions for MPEG-4 Video," *Journal of VLSI Signal Processing Systems*, Vol. 23, No. 1, October 1999, pp. 27-50.

[7] M. Berekovic, P. Pirsch, T. Selinger, K.-I. Wels, C. Miro, A. Lafage, C. Heer, G. Ghigo, "The TANGRAM Co-Processor for MPEG-4 Visual Compositing," *IEEE Intl. Workshop on Signal Processing Systems*, SIPS99, Oct. 99, pp. 311-320.

[8] J. Kneip, S. Bauer, J. Volmer, B. Schmale, P. Kuhn, M. Reißmann, "The MPEG-4 Video Coding Standard - a VLSI point of view," IEEE Intl. Workshop on Signal Processing Systems SIPS98, Boston, Oct. 1998.