A FLEXIBLE PROCESSOR ARCHITECTURE FOR MPEG-4 IMAGE COMPOSITING

M. Berekovic, R. Frase, P. Pirsch

Laboratorium für Informationstechnologie
Schneiderberg 32, 30167 Hannover,
Universität Hannover, Germany

ABSTRACT

This paper proposes a new array architecture for MPEG-4 image compositing. The emerging MPEG4 standard for multimedia applications allows script-based compositing of audiovisual scenes from multiple audio and visual objects. MPEG-4 supports both, natural (video) and synthetic (3D) visual objects or a combination of both. Objects can be manipulated, e.g. positioned, rotated, warped or duplicated by user interaction. A coprocessor architecture is presented, that works in parallel to an MPEG-4 video- and audio-decoder, and performs computation and bandwidth intensive low-level tasks for image compositing. The processor consists of an SIMD array of 16 DSPs to reach the required processing power for real-time image warping, alpha-blending and 3D rendering tasks. A programmable architecture allows to adapt processing resources to the specific needs of different tasks and applications. The processor has an object-oriented cache architecture with 2D virtual address space (e.g. textures), that allows concurrent and conflict-free access to shared data objects for all 16 DSPs. Especially I/O intensive tasks like texture-mapping, alpha-blending, image warping, z-buffer and shading algorithms benefit from shared memory caches and the possibility to preload data before it is accessed.

1. INTRODUCTION

The MPEG-4 standard, currently under development, shall deliver a standardized framework for all kind of multimedia applications: Examples include teleshopping, interactive, web-based games, edutainment, teleconferencing as well as mobile video-communication. This goal is achieved through an object-based approach for description and coding of multimedia contents.

In MPEG-4, audiovisual scenes are composed from multiple audio and visual objects that are coded separately. Information for scene composition is transmitted with the bitstream in a binary format for scene description (BIFS). At the decoder side, this information is used, together with inputs from the user, to composite the scene with audio and visual objects. User interaction like duplication, moving or zooming of objects shall be supported. Figure 1 shows a typical example from a news scene, where two video objects of the speakers have been extracted. In the decoder, they can be composited with arbitrary backgrounds and other visual objects like text overlays or rendered 3D-objects (virtual studio applications).

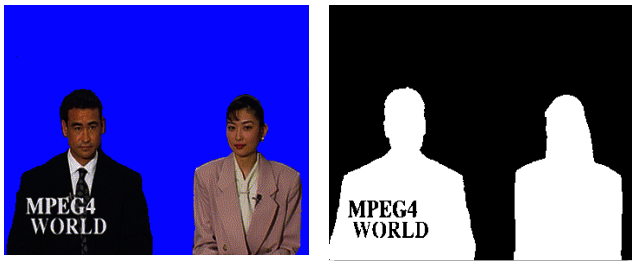

Figure 1. Typical application for MPEG-4 image compositing using multiple visual objects.

Figure 2 shows the block diagram of an MPEG-4 decoder. The incoming bitstream is demultiplexed into several elementary streams. After syntax decode and bitstream parsing, scene-description, control information and variable-length coded data of audiovisual objects are extracted. These objects are then decompressed and collected in an object pool memory. The scene is generated in the compositor from this data basis. In this final step objects may be manipulated, e.g. zoomed, moved, rotated or even cuted from the scene. Compositing includes complete 3D rendering of synthetic objects using a VRML-like scene description in binary format (BIFS).

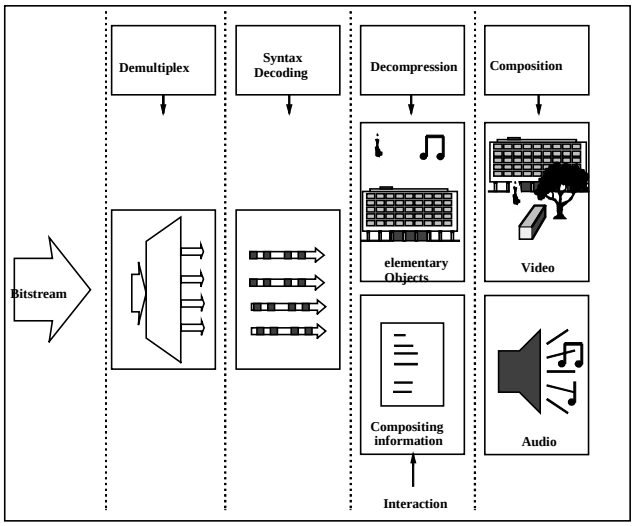

Figure 2. Block diagram of an MPEG-4 decoder.

Compared to existing video and audio coding standards like H.263 [1] or MPEG-2 [2], arithmetic complexity of MPEG-4 [3] decoders will increase significantly. This is mainly caused by the overhead introduced for the handling, decoding and compositing of multiple objects. Typical compositing tasks like alpha blending, image warping or 3D-rendering involve large amounts of low-level processing and I/O transfers.

Dedicated and programmable single-chip implementations have been proposed for existing video coding standards [4][5][6][7][8]. Instruction set extensions for general purpose processors have proven advantageous for many multimedia signal processing algorithms [9]. However, none of these solutions is adapted to the requirements of scene compositing. Especially real-time image warping and 3D rendering pose major problems to existing multimedia systems.

Several chips that support 3D rendering in hardware are known from literature [10]. The main disadvantage of these solutions is that their are adapted to the specific needs of 3D rendering, but they do not meet the demands of typical multimedia systems like object-oriented processing of data, integration of natural and synthetic visual objects, synchronization of audio and video objects, real-time processing of large textures (e.g. MPEG-2 video streams) or format conversions in real-time. However, the biggest problem they face is their inflexibility to allocate processing resources, both arithmetic units and caches, to *different* algorithms at different points of time. Although general purpose processors fulfill this requirement, they still lack performance for real-time processing of multimedia data.

We present a new array architecture, that has been designed to meet the specific demands of MPEG-4 scene compositing. This includes processing of arbitrarily sized 2D-objects, integration of 3D rendering and video display, real-time image warping and blending at TV quality. The architecture is based on a 2D mesh array of DSPs with autonomous SIMD controlling (A-SIMD). A new cache architecture is presented, that is adapted to the needs of parallel processing of 2D-objects, which is quite typical for image processing, video-coding and 3D rendering tasks.

This paper is organized as follows: The second section gives an overview of typical image compositing tasks and their processing requirements. The third section describes the presented array architecture with a novel object-oriented data cache while the last section will summarize the results.

2. ALGORITHM REQUIREMENTS

Compositing consists of multiple, optional tasks that are performed on 2D or 3D image data. Incoming 2D video objects can be repositioned, zoomed or even warped (full perspective transform). 3D objects can be rotated and moved in three dimensions. They are represented by meshes and corresponding textures.

Figure 4 shows the 3D rendering pipeline. It is subdivided into front-end calculations for the geometry setup, that typically involve floating-point operations and the back-end processing for rasterization, that is performed on large amounts of (pixel) integer data. Rasterization consists of texture mapping, bi- or trilinear interpolation, shading and z-buffer algorithm. It should be noted, that texture mapping with bilinear interpolation is quite similar to the image warping algorithm, foreseen by MPEG-4. Other high performance applications are thinkable. Data from a video stream could be used as textures in 3D scenes instead of rather static images.

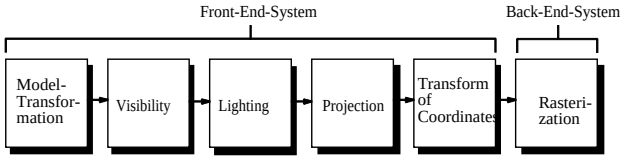

Figure 3. 3D rendering pipeline consisting of front-end and back-end computations

Figure 4 shows the system architecture of a MPEG-4 decoder system. Memory for the video-objects is shared between decoder processor and compositor processor. The compositor itself consists of two parts: the front-end system and the back-end (rasterizer). The proposed DSP array architecture is optimized for the processing of data-intensive low-level tasks. Due to the similarities in processing requirements, it is used for the back-end processing and for compositing of 2D video data (image filtering, warping and alpha blending). However, another processing module that supports floating-point operations is needed for the front-end system. It can run fully parallel to the video-decoding and compositing processors.

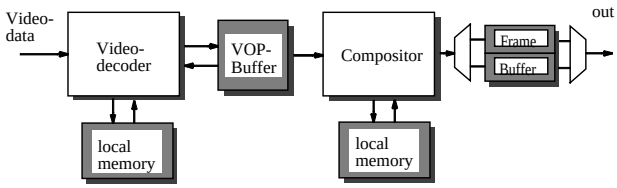

Figure 4. System architecture of a MPEG-4 decoder.

Table 1 summarizes front-end and back-end arithmetical requirements of 3D rendering. Many (costly) division operations are needed for back-end processing, especially for the calculation of the texture addresses. In a typical multimedia scenario, depending on number and size of the video-objects, a similar amount of operations is needed for image warping and alpha blending of 2D video objects. Since these parameters are application-dependent and can be changed by user interaction, it is not possible to give deterministic performance requirements for an MPEG-4 decoder system (as is the case for MPEG-2).

Triangles/ sec	Front-End [MFLOPs]	Front-End [MDIVs]	Back-End [MFLOPs]	**Back-End [MDIVs]**
10^5x40 pix	39	1	162	10
10^6x40 pix	390	9	1612	100
1^6x100 pix	39	1	1930	23
10^6x100 pix	390	9	3860	233

Table 1. Performance requirements for 3D rendering

Figure 5 shows the typical programming model for 2D-image filtering, which is similar for most image processing tasks. An input object with 2D data structures is processed with a smaller 2D object (the filter mask). As a result, a third object is generated. All three objects have 2D address spaces, that can differ in size.

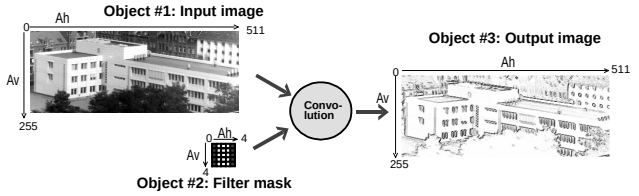

Figure 5. Typical 2D image processing scheme

Parallelization strategy is crucial for the performance of image processing systems. Figure 6 shows two possible MIMD-style parallelization schemes. The processing elements (PEs, here: DSPs), can either process different segments of the target image or different objects (here: triangles) in parallel. Parallelization on different segments will reduce utilization, since 2D objects are not equally distributed between image segments. The problem for parallelization on 2D objects (right side of Fig. 6) is, that since objects can be located at the same space in the image access conflicts for depth-information (z-, frame-buffer) occur, leading to a performance degradation.

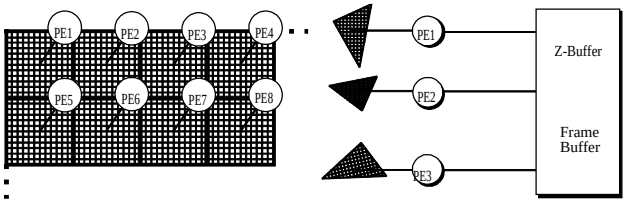

Figure 6. MIMD-style parallelization on segments (left) or objects (right)

We selected a parallelization scheme within objects, that allows simpler SIMD-style processing. Fig. 7 shows the parallelization of scanline algorithms for the processing of triangle data, that are very common in 3D rendering. Different PEs can operate either with different image lines or columns. Furthermore, matrix-shaped access patterns are possible.

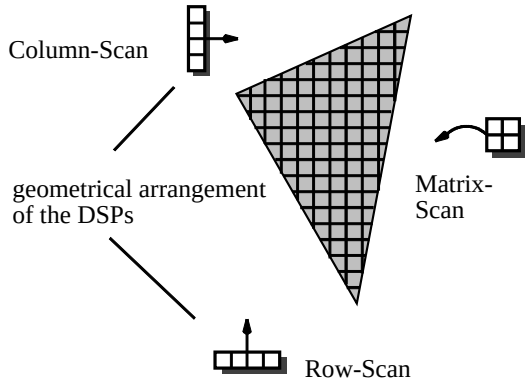

Figure 7. Parallelization of scan-line algorithms

3. ARRAY PROCESSOR ARCHITECTURE

An array architecture consisting of 16 programmable DSPs, each of them capable to executes up to 3 instructions per cycle, meets the demanding processing requirements of typical image compositing tasks. A parallel shared memory cache architecture delivers the bandwidth that is needed for real-time processing of video-streams. Figure 8 shows the basic concept of the DSP array processor architecture.

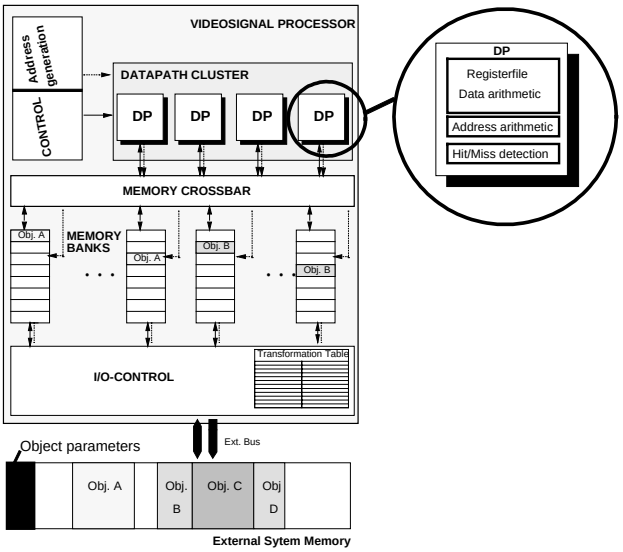

Figure 8. Processor architecture (with 4 DSPs)

3.1 Parallelization Strategy

It has been shown, that SIMD parallel processing fits the needs of rendering and compositing algorithms. However, an extension of this concept for autonomous addressing of local data and conditional execution of instructions within the DSPs proves very useful for parallelization of many algorithms (ASIMD) [11].

3.2 Parallel Cache Architecture

Compositor performance greatly benefits from concurrent and conflict-free access to 2D image, texture and Z-buffer data that is hold in on-chip cache memories. A shared memory cache has been implemented, that allows conflict-free access to multiple objects with arbitrary size and 1D or 2D address space [12][13]. Figure 9 shows the basic principle of the cache architecture. The DSPs have parallel access to shared image data, that is stored in a 2D array of memory blocks that are connected to the DSPs via configurable crossbars. In the shown example, 9 memory blocks are needed to enable concurrent, conflict-free access for four DSPs. The processors can access 2D objects either in matrix or vector form. Once defined, an access-style for an object cannot be changed without cache-flush. The position of matrix or vector access can be arbitrarily changed every clock-cycle. Furthermore, as shown in figure 9, undersampling is possible. In the shown example, the processors can leave out one or three samples. However, if two samples are skipped, all 4 datapaths need access to the same memory block, resulting in 3 stall cycles.

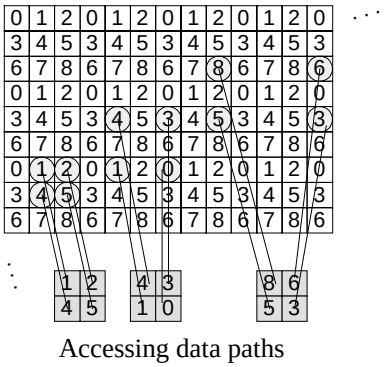

Figure 9. Cache organization with 9 memory-blocks.

Figure 10 shows, how concurrent cache access with autonomous, independant address calculation within the DSPs is utilized for 3D texture mapping and 2D image warping. Following this principle, 2D arrays of pixels can be processed in parallel, leading to a linear speedup, as long as the cache miss rate and access conflicts do not increase significantly.

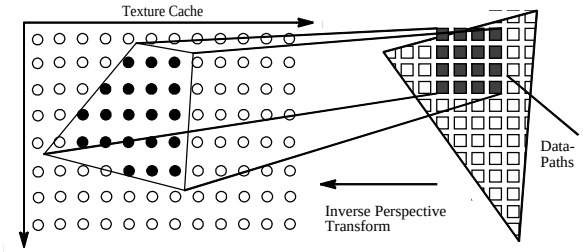

Figure 10. Parallel access to texture cache.

3.3 Data Path Architecture

Figure 11 shows the architecture of the DSP datapath. It consists of a register file of 16 32-bit registers and four different arithmetic modules, a 32 bit splitable ALU, a 32 bit splitable MAC unit, a shift and round unit as well as a division module. Up to three instructions can be executed every clock-cycle.

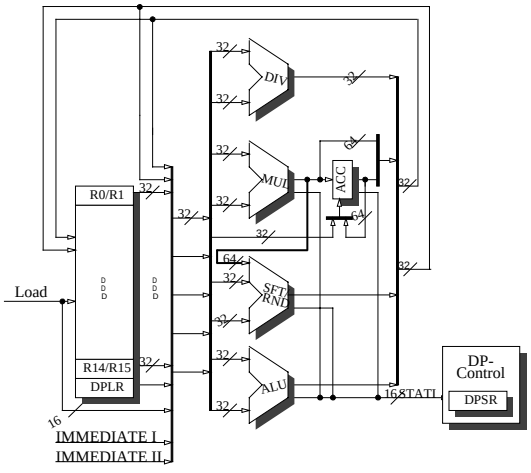

Figure 11. Architecture of the DSP datapath.

4. SUMMARY

We have presented the concept of a new array processor architecture for MPEG-4 compositing. The architecture is optimized to process typical compositing tasks such as 3D rendering, filtering, format conversion or alpha blending in parallel using 16 DSPs that are autonomously SIMD controlled. An object-oriented cache is provided, that allows parallel, conflict-free access to multiple image data objects of different size and dimension. Address calculation is performed in parallel to arithmetic processing within the cache unit. Programming is significantly simplified by a 1D/2D virtual address memory model with support for handling of multiple-objects. Translation from virtual internal to physical external addresses is done fully in hardware in Cache and DMA units. Using sophisticated RAM technology like RAMBUS, high performance multimedia applications based on MPEG-4 will become feasible.

5. REFERENCES

[1] ITU-T Draft Recommendation H.263: “Video Coding for Low Bitrate Communication”. July 1995.

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

[3] ISO/IEC JTC/SC29/WG11 “MPEG-4 video verification model V8.0”. MPEG96/N1796 July 1997.

[4] Turley, J. “Toshiba, TI Roll Out Set-Top Box Chips”. *Microprocessor Report*, Vol. 10, No. 7, p. 16, 1996.

[5] Pirsch, P., Demassieux, N., Gehrke, W. “VLSI Architectures for Video Compression – A Survey”. *Proceedings of the IEEE*, Vol. 83, No.2, pp. 220-246, February 1995.

[6] Glaskowski, P. N. “First Media Processors Reach the Market”. *Microprocessor Report*, Vol. 11, No. 1, pp. 10-16, Jan. 1997.

[7] Pirsch, P., Freimann, A., Berekovic, M. “Multimedia Signal Processors”. *Multimedia Hardware Architectures* in Proceedings SPIE Vol. 3021, pp. 2-13, 1997.

[8] Slavenburg, G. A., Rathnam,, S., Diskstra, H. “The Trimedia TM-1 PCI VLIW Media Processor”. *Proceedings Notebook for Hot Chips VIII*, pp. 171-177, *Stanford, 1996*.

[9] Lee, R. B. “Subword Parallelism with MAX-2”. *IEEE micro*, Vol. 16, No. 4, pp. 51-59, *August, 1996*.

[10] Glaskowski, P. N. ”3D Chips Break Megatriangle Barrier”. *Microprocessor Report*, Vol. 11, No. 7, pp. 16-21, 1997.

[11] Kneip, J., Berekovic, M., Wittenburg, J. P., Hinrichs, W., Pirsch, P. ”An Algorithm Adapted Autonomous Controlling Concept for a Parallel Single-Chip Digital Signal Processor”. *Journal of VLSI Signal Processing 16*, Vol. 11, No. 1, pp. 31-40, May 1997.

[12] Volkers, H. “Ein Beitrag zu Speicherarchitekturen programmierbarer Multiprozessoren der Bildverarbeitung”. PhD thesis, VDI Fortschrittsberichte, Reihe 9, Nr. 211, VDI Verlag 1992.

[13] Kneip, J., “Objektorientierte Cache-Speicher für programmierbare monolithische Mutiprozessoren in der digitalen Bildverarbeitung”. PhD thesis, University of Hannover, 1997.