Journal of VLSI Signal Processing 41, 9–20, 2005

HiBRID-SoC: A Multi-Core SoC Architecture for Multimedia Signal Processing

HANS-JOACHIM STOLBERG, MLADEN BEREKOVIĆ, SÖREN MOCH, LARS FRIEBE,
MARK B. KULACZEWSKI, SEBASTIAN FLÜGEL, HEIKO KLUßMANN,
ANDREAS DEHNHARDT AND PETER PIRSCH
Institute of Microelectronic Systems, University of Hannover, Appelstr. 4, 30167 Hannover, Germany

Received July 2003; Revised January 2004; Accepted March 2004

Abstract. The HiBRID-SoC multi-core system-on-chip architecture targets a wide range of multimedia applications with particularly high processing demands, including general signal processing applications, video de-/encoding, image processing, or a combination of these tasks. For this purpose, the HiBRID-SoC integrates three fully programmable processors cores and various interfaces onto a single chip, all tied to a 64-Bit AMBA AHB bus. The processor cores are individually optimized to the particular computational characteristics of different application fields, complementing each other to deliver high performance levels with high flexibility at reduced system cost. The HiBRID-SoC is fabricated in a 0.18 μm 6LM standard-cell CMOS technology, occupies about 81 mm^2, and operates at 145 MHz. An MPEG-4 Advanced Simple Profile decoder in full D1 resolution requires about 120 MHz for real-time operation on the HiBRID-SoC, utilizing only two of the three cores. Together with the third core, a custom region-of-interest (ROI) based surveillance application can be built.

Keywords: system-on-chip, multi-core, VLSI, multimedia, MPEG-4, surveillance

1. Introduction

The tremendous progress in VLSI technology allows the integration of an ever-increasing number of transistors on a single chip. Likewise, continuous improvements in algorithm research lead to increasingly sophisticated multimedia signal processing applications, demanding a steadily rising amount of processing power. Due to the high innovation rate in this field, the development and standardisation process of these applications is characterized by rapid changes in the algorithms and tools used. One example is the MPEG-4 video coding standard [1] of the Moving Picture Experts Group (MPEG), which has been introduced in 1999 with the Simple Profile, followed by—among others—the Advanced Simple Profile (ASP) in 2001, and its successor, MPEG-4 part 10 or Advanced Video Coding (AVC), in 2003. In total, more than 10 video profiles have been defined since 1999 within MPEG-4 for various application targets employing numerous algorithm options [2].

While the technological progress generally offers the potential to keep pace with the growing processing demands, the suitability of an implementation for advanced multimedia processing is determined by the architectural concept employed, i.e., how the transistors are actually spent on the chip. In the era of system-on-chip (SoC), multiple processing units can be integrated together with an extensive choice of interface modules on a single chip. In order to meet the demands of multimedia signal processing applications, however, an SoC must provide—in addition to a high level of arithmetic processing power—a sufficient degree of flexibility, integrate a powerful on-chip communication structure, and employ a well-balanced memory system to account for the growing amount of data to be handled when targeting higher-quality applications, e.g., in the area of video.

Existing approaches are either narrowly focused on a specific set of algorithm options, such as dedicated chips for the MPEG-4 Simple Profile [3], or consist of a very general DSP processing core [4] without specialization towards particular properties of the targeted algorithm class, potentially lacking processing performance for schemes with special processing demands. An extension of a programmable core with dedicated modules, as, e.g., in the Trimedia [5], does not help when the functions that have been hard-wired change in a new version of a multimedia standard. Homogeneous multiprocessors offer abundant levels of raw processing power [6, 7], but do not match well real-world compound multimedia processing schemes that typically involve quite diverse processing types.

The HiBRID-SoC multi-core architecture, developed at the University of Hannover, combines high processing power for multimedia schemes with high flexibility due to full software programmability. With three specifically adapted programmable cores on a single chip, various on-chip memory modules, and a 64-Bit AMBA AHB system bus, the HiBRID-SoC provides a versatile solution for stationary or mobile multimedia applications such as MPEG-4 video up to full D1 resolution, surveillance applications including object recognition and coding, or on-board data processing for reconnaissance applications.

In the following section, the architecture of the programmable multi-core SoC is presented in detail, including an overview of the three programmable cores and the hardware implementation results. The software development environment and performance results for application examples are presented in Section 3. Section 4 concludes the paper.

2. HiBRID-SoC Architecture Overview

The main components of the HiBRID-SoC, the three programmable cores, are each adapted towards a specific class of algorithms. The combination of the cores in an SoC design methodology has placed particular demands on the architectural concept of the chip including the on-chip communication structure.

2.1. Multi-Core SoC Architecture

With the rapid development cycles in today's multimedia algorithm research, programmability is a key requirement for a versatile platform designed to follow new generations of applications and standards. With programmable cores, several different algorithms can be executed on the same hardware, and the functionality of a specific system can be easily upgraded by a change in software.

Multi-core SoCs are attractive candidate architectures for multimedia processing: In general, these schemes can be partitioned into stream oriented, block oriented, and DSP oriented functions, which can all be processed in parallel on different cores. Each core can be adapted towards a specific class of algorithms, and individual tasks can be mapped efficiently to the most suitable core.

In general, different approaches exists to accelerate execution on programmable processors. In most cases, some kind of parallelization technique is employed on instruction level (e.g., very long instruction word, VLIW), data level (e.g., single instruction multiple data, SIMD), or on task level (e.g., simultaneous multithreading). Another very powerful means to accelerate multimedia processing is to adapt programmable processors to specific algorithms by introducing specialized instructions for frequently occurring operations of higher complexity [8].

Considering the wide range of multimedia applications targeted, the need for different cores handling separately typical 16-Bit DSP functions, block-based video tasks on mostly byte data, and bit-sequential stream processing operations has been identified. Consequently, the HiBRID-SoC multi-core architecture, shown in Fig. 1, comprises the following three programmable cores:

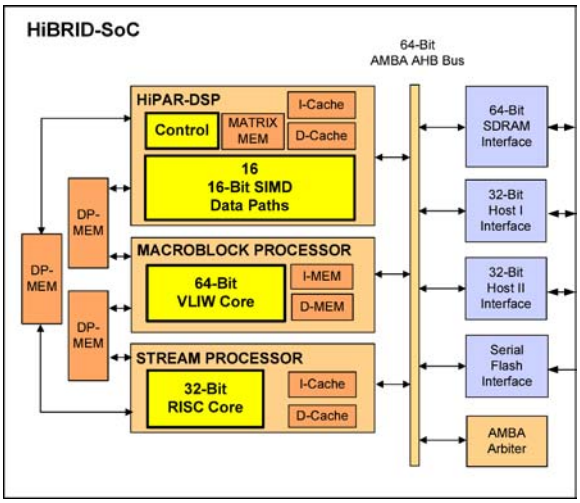

Figure 1. HiBRID-SoC multi-core architecture.

– the HiPAR-DSP core,
– the Macroblock Processor (MP) core,
– the Stream Processor (SP) core.

All three cores have been specifically optimized towards a particular class of algorithms by employing different architectural strategies. The HiPAR-DSP is a 16-data-path SIMD processor core controlled by a four-issue VLIW and is particularly optimized towards high-throughput two-dimensional DSP-style processing, such as FFT-intensive applications or filtering. The MP core has been designed specifically for the efficient processing of data blocks or macroblocks that are typical for many video coding schemes. It has a heterogeneous data path structure consisting of a scalar and a vector unit controlled by a dual-issue VLIW, offers flexible subword parallelism, and contains instruction set extensions for typical video processing computation steps. The SP, finally, consists of a scalar 32-Bit RISC architecture that is more optimized towards control-dominated tasks such as bitstream processing or global system control with a particular focus on high-level language programmability.

A 64-bit AMBA AHB system bus [9] connects all cores to off-chip SDRAM memory via a 64-Bit SDRAM interface, to two versatile 32-Bit host interfaces for access, e.g., to a host PC via PCI, and to serial flash memory for stand-alone applications. While the system bus operates at full internal clock frequency, the SDRAM and host interfaces support a programmable internal-to-external clock ratio (1:1, 1:2, 1:3 for the SDRAM interface, 1:2, 1:4, 1:6, 1:12 for the host interfaces) in order to facilitate adaptation to various system environments. The direct exchange of data and control information between the programmable cores without placing a burden on the system bus is supported by three dual-port shared memories of 2 kbyte size each.

Based on the standardized AMBA system bus, the architectural concept of the HiBRID-SoC follows a platform approach where individual cores may be exchanged for customization. Cores and external interfaces can easily be added for additional functionality, or removed to enhance silicon and pin efficiency.

2.2. *HiPAR-DSP Core*

The HiPAR-DSP is a highly parallel DSP core with a four-issue VLIW-controlled SIMD architecture, which has been previously developed at the University of Hannover [10]. Figure 2 shows a block diagram. The

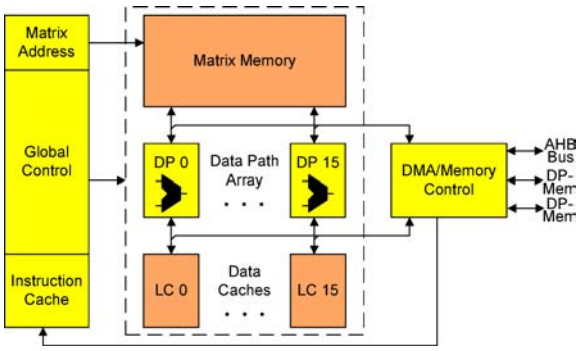

Figure 2. HiPAR-DSP architecture.

core consists of 16 identical 16-Bit data paths (DP0–DP15), each of which contains a local 32-entry register file, an ALU, a shift/round unit and a 16/36-Bit multiply/accumulate unit. Two slots of the VLIW are arithmetic instructions of the data paths for parallel execution, while the other two slots are used for control and memory operations, respectively. Depending on local status flags, a data path can also ignore the current instruction (Autonomous SIMD). Each data path has its own local 512 byte cache memory (LC) for autonomous random access of local data.

A 50 kbyte shared on-chip memory, called Matrix Memory, allows concurrent accesses of all data paths in matrix-shaped access patterns. Write access patterns are 4×4 matrices with distances 2^k, $k \geq 0$ between the elements; read access patterns also include 4-element vector or scalar accesses (distances ≥ 0). Dual-port memory modules enable DMA transfers to occur in parallel to data path accesses. This memory concept provides an easy data exchange between the data paths, which is required for many filter and image processing algorithms.

The HiPAR-DSP is controlled by a global control unit, which has its own local 48-entry register file and ALU. It is used for global control operations like address calculation, loops, or branches. Furthermore, the controller fetches the 128-Bit VLIW from the 8 kbyte instruction cache and broadcasts the arithmetic instructions to all data paths.

An autonomously operating DMA unit serves all cache misses and performs data prefetch transfers to the matrix memory. The DMA unit is connected directly to the AMBA AHB bus and the dual-port memories connecting the HiPAR-DSP with the MP and the SP.

At the clock frequency of 145 MHz, the HiPAR-DSP achieves a performance of 2.3 GMAC/s, making it a versatile processor for all kinds of applications with

high numerical processing demands and available data parallelism.

2.3. Macroblock Processor (MP) Core

The MP has a heterogeneous data path structure consisting of a scalar and a vector data path, as shown in 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 splittable 64-entry register file of 64-Bit width. Special function units provide instruction set extensions for common video and multimedia core algorithms. The 64-Bit-wide arithmetic execution units in the vector path, e.g., MUL/MAC or ALU, incorporate SIMD-style 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 per cycle in parallel.

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 was removed by the introduction of field-wise conditional execution in the MP, which allows to sustain the SIMD scheme even for data-dependent processing. With its powerful and flexible support for subword parallelism, the vector data path is particularly suited to process the repetitive operations of typical macroblock algorithms at high throughput.

The MP's parallel data paths are controlled by a dual-issue 64-Bit VLIW. 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 and 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.

The MP accesses its local 16 kbyte instruction memory and 4 kbyte data memory within a single clock cycle. Transfers between local memories and external memory are performed in the background by an autonomous DMA unit as the program execution continues. For the core algorithms, scalar and vector path can operate in concert to exchange blocks of video data between the local data memory and the vector register file without stall cycles by performing stores and loads concurrently.

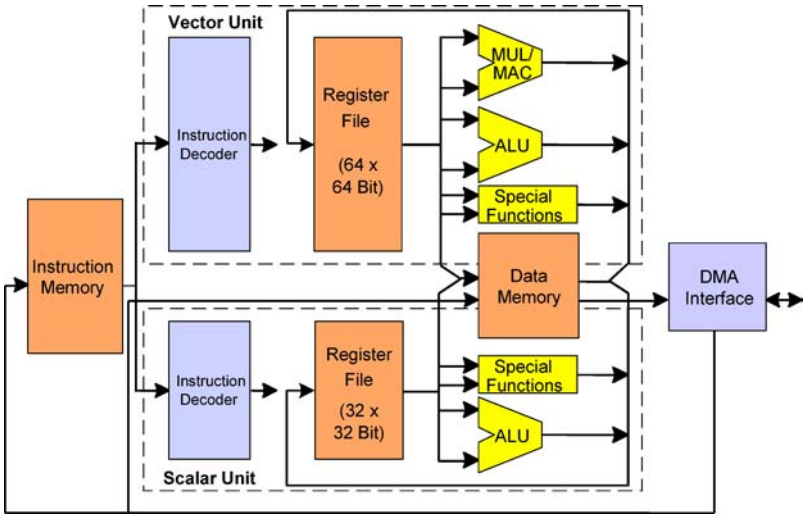

Figure 3. Macroblock processor architecture.

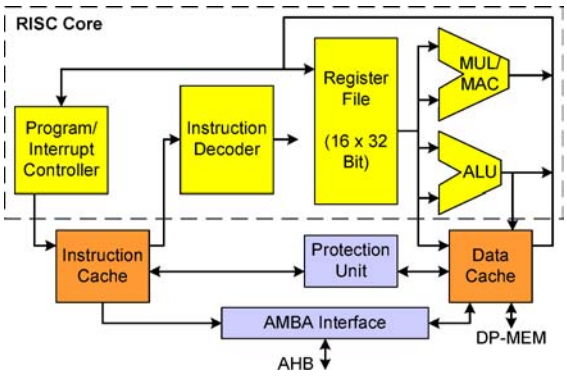

Figure 4. Stream processor architecture.

2.4. *Stream Processor (SP) Core*

The SP (Fig. 4) has been optimized for high-level programmability and efficient processing of control-driven applications. It consists of a scalar single-issue, in-order execution RISC core with a 32-Bit data path consisting of 5 pipeline stages and controlled by 32-Bit RISC instructions. The set of function units comprises a 32-Bit ALU, a MUL/MAC unit, an extended shift capability, and support for fast decoding of prefix code words. Additionally, the SP supports conditional execution, forwarding, interlocks, and provides full interrupt capability.

The SP's memory architecture consists of separate caches for instructions and data, each 4 kbyte in size with 4-way set associativity. They convert the 64-Bit AMBA bus width to the 32-Bit internal width. Both caches share a memory protection unit which can distinguish up to eight regions in the address space. The data cache also performs the memory mapping for the access of the intercore dual-port memories.

2.5. *SoC Communication Concept*

The efficient implementation of real applications requires their subtasks to be mapped onto the different cores according to their computational characteristics. As each core itself is a powerful, self-contained processing device, a coarse-grain partitioning of applications is favored in order to keep the interaction and interdependence of the cores as low as possible for the sake of efficiency. Consequently, the partitioning of applications and mapping onto cores involves the careful definition of data exchange points and of the layout of data structures to be exchanged.

During processing, data structures are exchanged through the on-chip dual-port memories, each linking two cores. This method provides highest flexibility in terms of data structure layout and is suitable for a wide range of applications. A simple semaphore mechanism can be implemented using designated memory locations in the dual-port memories to synchronize shared access to data structures.

As another means of synchronization, dedicated general-purpose IO registers can be accessed which are arranged as bus slaves on the AMBA bus. The arbitration logic of the AMBA bus inherently guarantees a conflict-free access to the registers from each core.

2.6. *HiBRID-SoC Implementation*

The HiBRID-SoC has been fabricated in a 0.18 μm 6-metal layer standard-cell CMOS technology and integrates about 14 million transistors on the chip. Figure 5 shows the chip micrograph. In total, the HiBRID-SoC occupies about 81 mm^2, with more than half of the area consumed by the HiPAR-DSP and its memories. MP and SP core including memories together account for about 30% of the area, and the rest is occupied by the dual-port memories and interfaces. The chip operates at a frequency of 145 MHz. An overview on the implementation data is given in Table 1.

A flexible PCI evaluation board is available for the development of a wide range of image and video

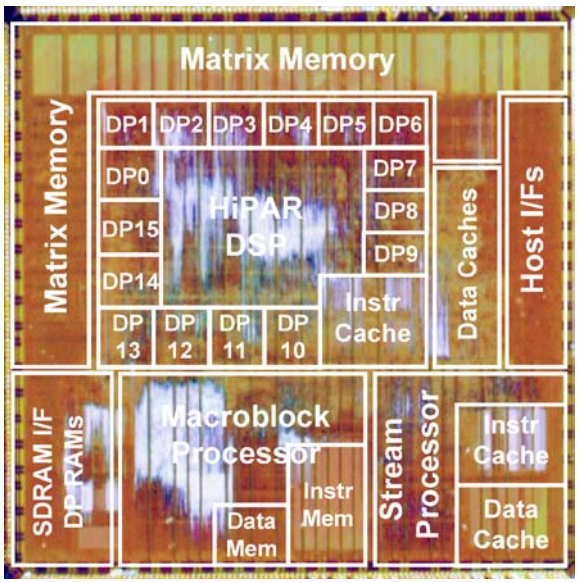

Figure 5. Chip micrograph of the HiBRID-SoC.

Table 1. HiBRID-SoC chip implementation data.

Technology	0.18 μm CMOS, 6-metal layer	
Supply voltage	1.8 V (internal), 3.3 V (I/O)	
Clock frequency	145 MHz	
Transistor count	14 M	
Die size	8.9 mm $\times$ 9.1 mm = 80.99 mm^2	
Signal I/Os	220	
Package	SBGA352	
Power dissipation	Core:	1.76 W @ 145 MHz, 1.8 V
	I/O:	1.6 W (estimated)

encoding, decoding, and processing applications with the HiBRID-SoC.

3. Multimedia Application Development

A powerful software development environment is essential to support the efficient implementation of multimedia applications on the HiBRID-SoC. The development of applications comprises a static assignment of tasks to the most appropriate core and the definition of data exchange structures and synchronization points between the cores.

3.1. Software Development Environment

All three processor cores provide various special architectural features that have to be supported by the software development environment. Specifically the HiPAR-DSP and the MP use data parallelism via SIMD and subword parallelism, instruction parallelism via VLIW, and include special instructions optimized for image and video processing algorithms. For each core, optimizing assemblers are available that support these features and perform code reordering, instruction scheduling, and VLIW parallelization. On top of the assemblers, C and C++ compilers have been developed for each core. The HiPAR-DSP compiler has been extended to support matrix data types for SIMD and Matrix Memory access programming. The MP compiler is mainly used for application controlling, whereas the most computation-intensive core tasks are typically written in assembly language for full exploitation of all data path features. The compiler for the SP supports the full set of features provided by the architecture. For application code development, cycle-accurate instruction-set simulators with graphical user interface are available for each core.

3.2. Task Mapping Strategy

The typical implementation flow for complex multimedia applications consists of partitioning the applications into tasks according to their computational characteristics, and mapping the tasks onto the most appropriate core in a static way to be executed concurrently. Generally, the partitioning of applications will occur at a very coarse-grain level in order to avoid an excessive data exchange and interdependence among cores. This is supported by the full programmability of the cores which each can accommodate to a wide range of different tasks, resulting in a high flexibility in the task mapping in many cases. Tasks which benefit from access to a large 2D memory will be mapped onto the HiPAR-DSP; block-oriented tasks that profit from subword data parallelism are mapped onto the MP; and sequential tasks with complex control flow will run on the SP core. For synchronization and data exchange on task level, a data structure layout can be defined in the intercore dual-port memories. Efficiency of execution increases as the task synchronization occurs on a coarser level.

3.3. Application Example: MPEG-4 Advanced Simple Profile

As one application example, an MPEG-4 Advanced Simple Profile (ASP) decoder has been implemented on the HiBRID-SoC. The implementation utilizes only the MP and SP cores of the HiBRID-SoC, leaving the HiPAR-DSP core completely free for other applications.

MPEG-4 ASP belongs to the group of streaming video profiles as defined in MPEG-4 Version 2/Amd. 2 [11]. It provides the capability to distribute single layered, frame-based video at a wide range of bitrates, targeting video on Internet. The Advanced Simple Profile, in particular, combines high coding efficiency and reduced implementation complexity: With bi-directionally predicted video object planes (B-VOPs) and advanced motion compensation tools such as quarter-pel motion compensation (QMC) and global motion compensation (GMC), the coding performance is clearly superior to the Simple profile. Omitting arbitrarily-shaped objects, on the other hand, keeps the implementation complexity significantly lower than, e.g., for the Advanced Coding Efficiency (ACE) profile or even the Core profile. A detailed complexity analysis of MPEG-4 ASP based on bitstream characteristics

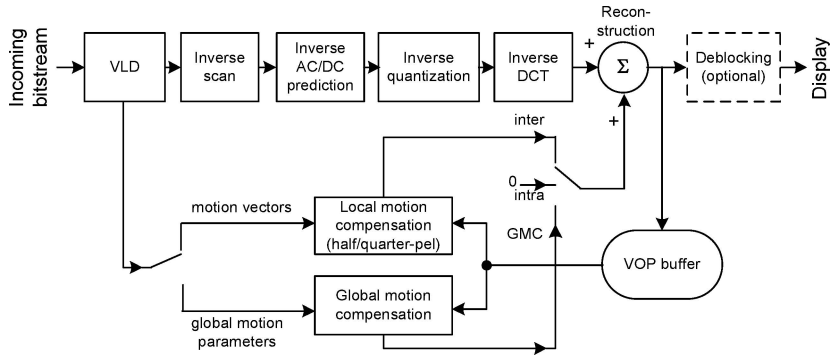

Figure 6. MPEG-4 ASP decoder block diagram.

is provided in [12]. Figure 6 shows the block diagram of an MPEG-4 ASP decoder.

Six levels (L0–L5) have been defined for ASP in total, with maximum bitrates from 128 kBit/s (L0, L1) to 8 MBit/s (L5) and typical session sizes from QCIF (176×144 for L0, L1) up to D1 resolution (720×576 for L5). ASP supports up to four rectangular objects (Advanced Simple or Simple type), and L4 and L5 additionally include support for interlaced video coding.

The tasks of MPEG-4 ASP are partitioned onto the cores according to their computational characteristics. The SP performs all bitstream parsing-related tasks as well as motion vector decoding and variable-length decoding (VLD). In addition, the SP also performs inverse DC/AC prediction and inverse quantization (IQ)—for low-bitrate streams, the extensive parallel processing capabilities of the MP would be wasted here due to sparse coefficient matrices. After IQ, the stream processor prepares the discrete cosine transform (DCT) coefficients macroblock-wise in the dual-ported on-chip memory and signals their availability to the MP for further processing. Likewise, decoded motion vectors and coding mode information are also exchanged between SP and MP through the shared dual-port memory.

All subsequent steps are performed on the MP core, in dependence of the VOP type, macroblock type, and the signaled coding mode. The IDCT is efficiently implemented on the MP with its four 16-Bit MACs operating in parallel. Reduced-complexity versions of the IDCT can save up to 90% of cycles for the DC-only case compared to the full IDCT. Pixel interpolation for half-pel motion compensation is supported by the MP's average instruction that also takes into account the rounding control parameter defined in MPEG-4. FIR filtering in quarter-pel compensation also relies on the MAC instruction, and VOP reconstruction benefits from the saturation mode available in the MP's arithmetic instructions.

The organization of memory transfers is of particular importance when implementing MPEG-4 ASP decoding, particularly when targeting higher resolutions such as D1 involving larger data volumes. By utilizating the several data transfer paths inside the MP, memory cycles can be hidden almost completely behind the processing cycles. The data transfer paths include: SP/MP dual-port memory to MP data memory, dual-port memory to MP vector register file, data memory to vector register file, or data memory to external frame buffer via autonomous DMA via AMBA bus. Both vector unit and scalar unit can initiate memory transfers simultaneously.

A system view of the MPEG-4 ASP decoding flow on the HiBRID-SoC is depicted in Fig. 7. The incoming bitstream is read by the SP through host interface 0. The SP signals the availability of a new data structure in the dual-port memory to the MP core on macroblock level. The MP core completes the decoding process by using the external SDRAM as buffer for reference frames and writing out the decoded video sequence through

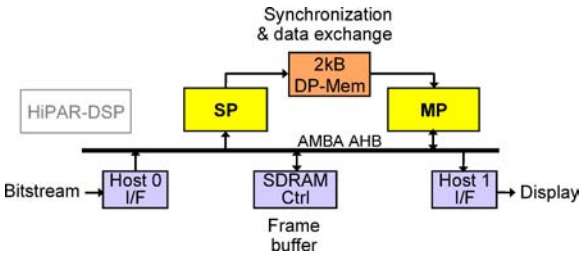

Figure 7. MPEG-4 ASP decoding flow using MP and SP core.

Table 2. MPEG-4 ASP decoder performance on MP, 720 × 576@25Hz, 1.5–3 MBit/s.

Decoder subtask	*Cycles/block*	*MHz*	*%*
IDCT			
Skipped	8	1.5	1.2
DC only	21	0.6	0.5
Full IDCT	190	11.6	9.4
Motion compensation			
Basic mode (full pel)	71	14.3	11.6
Linear interpolation	78	6.4	5.2
Bilinear interpolation	139	6.0	4.9
8-tap FIR (QMC)	104	16.9	13.7
Warping (GMC)	1,670	56.8	46.0
Reconstruction			
Copy	10	0.6	0.5
Add	18	0.6	0.5
Average	26	2.1	1.7
Average and add	36	2.0	1.6
Others		4.0	3.2
Total		123.4	100.0

host interface 1. Both the SP and MP core operate fully parallel in a macroblock level pipeline.

The MPEG-4 ASP decoder has been implemented for real-time operation in full D1 resolution (720 × 576, 25 Hz) at bitrates of 1.5–3 MBit/s. The performance requirements of the relevant decoder subtasks on the MP are given in Table 2. In total, around 120 MHz are required on the MP to perform real-time MPEG-4 ASP decoding. An MPEG-4 ASP encoder is estimated to achieve a frame rate of 25 Hz in CIF resolution (352 × 288).

In comparison, Equator's MAP-CA processor [15] has been reported to only decode MPEG-4 ASP Level 3 in CIF format at 300 MHz [16]. Dedicated implementations, e.g., Sigma Design's EM847× family [17] frequently lack support for features specific to the ASP, such as quarter-pel motion compensation or global motion compensation.

As the results in Table 2 show, GMC warping consumes almost half of the processing cycles in this implementation. As the decoder is implemented completely in software, detailed tuning of particularly this task is possible in order to further decrease the computational load. This can be advantageous if additional tasks are desired to be mapped to the MP core, e.g., optional deblocking.

It has to be noted that the MPEG-4 ASP decoder implementation presented here requires only two of the three cores of the HiBRID-SoC. A large amount of processing resources is still available to add further value to this application or to perform other applications simultaneously, as demonstrated in the following application example.

3.4. Application Example: Region-of-Interest (ROI) Sensitive Video Encoding

The complementing capabilities of the cores in the video compression and image processing domain makes the HiBRID-SoC also an ideal implementation platform for advanced surveillance application schemes. For a region-of-interest (ROI) sensitive video encoding scheme, the HiPAR-DSP operates on the input video stream of a static or moving camera and performs object tracking consisting of object pixel detection, connected component labeling, and ROI detection. For these tasks, the data path array and matrix memory of the HiPAR-DSP can efficiently be utilized by employing a divide-and-merge strategy of operation. The ROI coordinates as the result of the object tracking and ROI detection are transferred through the dual-port memory to the MP core, providing the coding control for an MPEG-4 Simple Profile encoding scheme being performed on the MP and SP core.

Figure 8 shows the flow of the ROI sensitive encoding scheme and demonstrates how the ROI information is incorporated into the encoded video. The full programmability of the HiBRID-SoC allows to customize the compression scheme, e.g., by choosing different quality parameters for the ROI and background regions. On the receiver side, another HiBRID-SoC can serve as a multi-channel decoder for several incoming data streams. The ROI sensitive encoding schemes

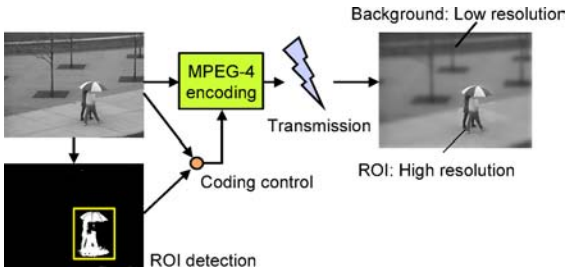

Figure 8. ROI sensitive encoding scheme.

Table 3. Performance of core tasks in the ROI sensitive encoding scheme, 720 × 576@25 Hz.

Subtask	*Cycles/block*	*MHz*
Object tracking (HiPAR-DSP)		
Object pixel detection	1,500	4.1
Connected component labelling	24,100	65.1
MPEG-4 encoding (MP)		
Motion estimation (SAD), full-/half-pel	91/183	64.3

facilitates a remote surveillance application using a very low bitrate channel and has the additional benefit of assisting the human interpretation of the recorded scenes.

The performance data for the core tasks of the ROI sensitive encoding scheme on both the HiPAR-DSP and the MP core are given in Table 3. The resulting computational load is about 50% on the HiPAR-DSP for the ROI detection in full D1 resolution and about 90% for the MPEG-4 Simple Profile encoding on the MP/SP core combination.

3.5. Further Classes of Applications

Several applications from the image and general signal processing domain have already been implemented on the HiPAR-DSP, e.g., image synthesis from Synthetic Aperture Radar (SAR) data [14]. The algorithms are based on 2-dimensional filtering in the frequency domain, requiring a high FFT performance. At a clock frequency of 145 MHz, the HiPAR-DSP can perform a complex 16 Bit 1k FFT in 22 μs and a 4k FFT in 106 μs. This performance compares favorably with a number of other DSPs running at much higher clock frequency, as Table 4 shows. Performance benchmarks for other applications implemented on the HiPAR-DSP core are given in Table 5.

Table 4. Performance comparison on 1k complex 16-Bit FFT (from [19]).

DSP	Clock speed (MHz)	Complex 1k FFT (μs)
HiPAR-DSP core	145	22
MPC7410	500	21
TMS320C6203	300	44
TMS320C6201	200	66
TMS320C6701	167	108
ADSP-21060	40	460
TMS320C40	50	1890
TMS320C30/31	40	2534

Table 5. HiPAR-DSP core image processing performance benchmarks.

Algorithm	Description	Performance
Histogramming	256 k samples, 256 grey levels	380 μs
Hough transform	R-quant = 2048, Φ-quant = 128, image size 512 × 512, 30% black pels	63 ms

All three cores together provide a powerful implementation platform for a wide range of applications in the multimedia signal processing domain with the specific advantage to be able to follow future evolutions of algorithms and processing schemes.

4. Conclusions

The HiBRID-SoC provides a powerful and versatile system-on-chip solution for various kinds of multimedia signal processing applications. With its three programmable cores adapted to different classes of algorithms, different applications can efficiently be mapped onto this architecture. The full software programmability of all three cores facilitates to keep pace with the rapid algorithm developments in this field. The extensible platform approach of the HiBRID-SoC allows to exchange or add additional IP cores for further customization towards specific system environments.

Acknowledgments

This work was funded by the Fraunhofer Gesellschaft under contract T/F31D/1A232/P1307. The authors would like to thank X. Mao for design contributions and Micronas GmbH for tape-out support.

References

1. ISO/IEC JTC1/SC29/WG11 N4668, "Overview of the MPEG-4 Standard," Jeju, March 2002.

2. P. Pirsch, M. Bereković, H.-J. Stolberg, and J. Jachalsky, "VLSI Architectures for MPEG-4," in *Proc. 2003 International Symposium on VLSI Technology, Systems, and Applications*, 2003, pp. 208–212.
3. M. Takahashi, T. Nishikawa, M. Hamada, T. Takayanagi, et al., "A 60 MHz 240-mW MPEG-4 videophone LSI with 16-Mb embedded DRAM," *IEEE Journal Solid-State Circuits*, vol. 35, no. 11, 2000, pp. 1713–1721.
4. N. Seshan, "High VelociTI Processing," in *IEEE Signal Processing Mag.*, 1998, pp. 86–101.
5. Philips, *TriMedia TM-1300 Media Processor Data Book*, Sep. 2000.
6. M. Rudack, M. Redeker, J. Hilgenstock, S. Moch, and J. Castagne, "A Scalable Large Area Integrated Multiprocessor System for Video Applications with Self-Configuration Facilities," *IEEE Design & Test of Computers*, vol. 19, no.1, 2002, pp. 6–17.
7. B. Ackland, A. Anesko, D. Brinthaupt, et al., "A Single-Chip, 1.6-Billion, 16-b MAC/s Multiprocessor DSP," *IEEE Journal of Solid-State Circuits*, vol. 35, no. 3, 2000.
8. M. Bereković, H.-J. Stolberg, M.B. Kulaczewski, P. Pirsch, H. Möller, H. Runge, J. Kneip, and B. Stabernack, "Instruction Set Extensions for MPEG-4 Video," *Journal VLSI Signal Processing Systems*, vol. 23, 1999, pp. 27–50.
9. ARM Ltd., *AMBA Specification Rev. 2.0*, www.arm.com, May 1999.
10. J.P. Wittenburg, W. Hinrichs, H. Lieske, H. Kloos, L. Friebe, and P. Pirsch, "HiPAR-DSP—A Scalable Family of High Performance DSP-Cores," in *Proc. of the 13th Annual IEEE International ASIC/SOC Conference*, 2000, pp. 92–96.
11. ISO/IEC 14496-2:2001/Amd 2:2002, "Streaming video profile," Feb. 2002.
12. H.-J. Stolberg, M. Berekovic, P. Pirsch, and H. Runge, "The MPEG-4 Advanced Simple Profile—A Complexity Study," in *Proc. 2nd Workshop and Exhibition on MPEG-4*, 2001, pp. 33–36.
13. J. Curlander and R. McDonough, *Synthetic aperture radar: Systems and Signal Processing*, Wiley-Interscience, 1991.
14. C. Simon-Klar, L. Friebe, H. Kloos, H. Lieske, W. Hinrichs, and P. Pirsch, "A Multi DSP Board for Real Time SAR Processing using the HiPAR-DSP 16," in *Proceedings of the International Geoscience and Remote Sensing Symposium 2002*, 2002, pp. 2750–2752.
15. C. Basoglu, W. Lee, and J. O'Donnell, "The Equator MAP-CA DSP: An End-to End Broadband Signal Processor VLIW," *IEEE Trans. Circuits Systems Video Technol.*, vol. 12, no. 8, 2002, pp. 646–659.
16. DynaPel Systems, Inc., "DynaPel MPEG-4 Video Codec SDK," Product Brochure, Nov. 2002.
17. Sigma Designs, "EM8470 Series: MPEG-4 Decoder for Set-top, DVD and Streaming Applications," Product Brief, 2002.
18. ISO/IEC JTC1/SC29/WG11 and ITU-T SG16 Q.6, "Study of Final Committee Draft of Joint Video Specification (ITU-T Rec. H.264—ISO/IEC 14496-10 AVC)," Awaji, Dec. 2002.
19. Model 4294 Quad MPC7410 PowerPC VME Processor, *Pentek Processor Brochure*, 2002, http://www.pentek.com/support/GetOTD.cfm?Filename=procbroc.pdf.

Hans-Joachim Stolberg received the Dipl.-Ing. degree in electrical engineering from the University of Hannover, Germany, in 1995.

From 1995 to 1996, he was with the NEC Information Technology Research Laboratories, Kawasaki, Japan, working on efficient implementations of video compression algorithms. Since 1996, he has been with the Institute of Microelectronic Systems at the University of Hannover as a Research Assistant. During summer 2001, he was a Monbukagakusho Research Fellow at the Tokyo Institute of Technology, Japan. His current research interests include VLSI architectures for video signal processing, performance estimation of multimedia schemes, and profile-guided memory organization for signal processing and multimedia applications.
stolberg@ims.uni-hannover.de

Mladen Bereković received the Dipl.-Ing. degree in electrical engineering from the University of Hannover, Germany, in 1995.

Since then he has been a Research Assistant with the Institute of Microelectronic Systems of the University of Hannover. His current research interests include VLSI architectures for video signal processing, MPEG-4, System-on-Chip (SOC) designs, and simultaneously multi-threaded (SMT) processor architectures.
berekov@ims.uni-hannover.de

Sören Moch received the Dipl.-Ing. degree in electrical engineering from the University of Hannover, Germany, in 1997.

Since then he has been Research Assistant with the Laboratory for Information Technology, University of Hannover. His current research interests are in the area of processor architectures for image, video and multimedia signal processing applications.
moch@ims.uni-hannover.de

Lars Friebe studied electrical engineering at the Universities Ulm and Hannover, Germany. In 1999, he worked at the NEC System ULSI Research Laboratory in Kanagawa, Japan. He received the Dipl.-Ing. degree in electrical engineering from the University of Hannover, Germany, in 1999.

Since then he has been a Research Assistant with the Laboratory for Information Technology, University of Hannover. His current research interests are in the area of parallel programmable VLSI architectures for real-time image processing.
friebe@ims.uni-hannover.de

Mark B. Kulaczewski started his studies in electrical engineering at the University of Hannover, Germany. In 1994, he transferred to Purdue University, West Lafayette, USA, and received the M.S. degree in electrical engineering in 1996.

Since 1997 he has been a Research Assistant at the Laboratory for Information Technology and the Institute of Microelectronic Systems, University of Hannover. His current research interests include programmable real-time architectures for video coding and image segmentation, and instruction-set extensions for cryptographic applications.
mbk@ims.uni-hannover.de

Sebastian Flügel was born in Crivitz, Germany, in 1975. He received his Dipl.-Ing. degree from the Department of Electrical Engineering of the University of Rostock in 2001.

Since then he has been a Ph.D. candidate at the Institute of Microelectronic Systems at the University of Hannover. He works in the field of architectures and systems for video processing systems. His focus is on algorithms for video encoding and the development of optimized hardware architectures.
fluegel@ims.uni-hannover.de

Heiko Klußmann received the Dipl.-Ing. degree in computer engineering from the University of Hannover, Germany, in 2002.

Since then he has been a Research Assistant with the Institute of Microelectronic Systems of the University of Hannover. His current research interests are in the area of programmable architectures for real-time video signal processing.
klussman@ims.uni-hannover.de

Andreas Dehnhardt was born in Frankfurt am Main, Germany, in 1976. He received his Dipl.-Ing. degree in electrical engineering from the University of Hannover, Germany, in 2002.

Since then, he has been a Research Assistant with the Institute of Microelectronic Systems, University of Hannover. His current research interests include programmable architectures for multimedia applications and implementation of real-time MPEG-4 encoding schemes.
dehnhard@ims.uni-hannover.de

Peter Pirsch received the Ing. grad. degree from the engineering college in Hannover, Hannover, Germany, in 1966, and the Dipl.-Ing. and Dr.-Ing. degrees from the University of Hannover, in 1973 and 1979, respectively, all in electrical engineering.

From 1966 to 1973 he was employed by Telefunken, Hannover, working in the Television Department. He became a Research Assistant at the Department of Electrical Engineering, University of Hannover, in 1973, a Senior Engineer in 1978. During 1979 to 1980 and in Summer 1981 he was on leave, working in the Visual Communications Research Department, Bell Laboratories, Holmdel, NJ. During 1983 to 1986 he was Department Head for Digital Signal Processing at the SEL research center, Stuttgart. Since 1987 he is Professor in the Department of Electrical Engineering, since 2002 in the Department of Computer Science at the University of Hannover. He served as Vice President Research of the University of Hannover from 1998 to 2002. His present research includes architectures and VLSI implementations for image processing applications, rapid prototyping and design automation for DSP applications. He is the author or coauthor of more than 200 technical papers. He has edited a book on VLSI Implementations for Image Communications (Elsevier 1993) and is author of the book Architectures for Digital Signal Processing (John Wiley 1998).

Pirsch is a member of the IEEE, the German Institute of Information Technology Engineers (ITG) and the German Association of Engineers (VDI). He was recipient of several awards: the NTG paper price award (1982), IEEE Fellow (1997), IEEE Circuits and Systems Golden Jubilee Medal (1999). He was member or chair of several technical program committees of international conferences and organizer of special sessions and preconference courses. He has held several administrative and technical positions with the IEEE Circuits and Systems Society and other professional organizations. Dr. Pirsch currently serves as Vice President Publications of the IEEE Circuits and Systems Society. Since 2000 he is chairman of the Accreditation Commission for Engineering and Informatics of the Accreditation Agency for Study Programs in Engineering, Informatics, Natural Science and Mathematics (ASIIN). Dr. Pirsch is chair of the VDI committee on Engineering Education.
pirsch@ims.uni-hannover.de