A Video Signal Processor for MIMD Multiprocessing

Jörg Hilgenstock Klaus Herrmann Jan Otterstedt[*] Dirk Niggemeyer Peter Pirsch

Laboratorium für Informationstechnologie
Universität Hannover
Schneiderberg 32
30167 Hannover, Germany
Tel: +49 511 762 5058
E-mail: hilgenst@mst.uni-hannover.de

ABSTRACT

The video signal processor AxPe1280V has been developed for implementation of different video coding applications according to standards like ITU-T H.261/H.263, and ISO MPEG-1/2. It consists of a RISC processor supplemented by a coprocessor for convolution-like low-level tasks. RISC and coprocessor have been implemented in a standard cell design combined with full-custom modules. The processor was fabricated in a 0.5 $\mu\mathrm{m}$ CMOS technology and has a die size of 82 mm^2. It provides a peak performance of more than 1 giga arithmetic operations per second (GOPS) at 66 MHz. For processing of very computation-intensive algorithms or high data rates, several processors can be bus-connected to form a MIMD multiprocessor system.

1. INTRODUCTION

For real-time coding and transmission of video data, several international standards have been developed. These include ITU-T H.261/H.263 [1, 2] for video telephone, ISO MPEG-1 [3] for multimedia, and ISO MPEG-2 [4] for digital TV. All these standards utilize hybrid coding techniques [5]. They combine computation-intensive low-level tasks, like motion estimation or discrete cosine transform, with data-dependent medium-level tasks, like variable length coding, variable threshold or quantization.

The performance requirements of these standards are determined on one hand by the complexity of the employed algorithms. On the other hand, they are mainly based on image size and frame rate. Altogether, the performance requirements for real-time coding range from a few hundred million operations per second (MOPS) for a video telephone decoder, up to several giga arithmetic operations per second (GOPS) for a digital TV encoder. For these applications, a cost-effective and small system implementation is required. Therefore a VLSI implementation, which can provide the needed processing power, is essential.

[*] now with Siemens Semiconductors, Munich, Germany.

Two implementation strategies can be followed: First, an implementation of the codec's architecture dedicated to the envisaged application field can be used to reduce the required implementation area [6]. The drawback of this approach is, that the implementation of a modified coding algorithm often necessitates a redesign of the involved circuits.

Alternatively, programmable processors can be used. They allow flexible adaption to different coding applications and standards. A disadvantage of programmable solutions is the increased silicon area. Especially the use of general purpose digital signal processors for the above mentioned video coding applications often leads to an unacceptable system size. Therefore, the programmable video signal processor architecture AxPe has been developed, which has been optimized for hybrid video coding algorithms and can be used in a scalable multiprocessor system.

In the following the video signal processor AxPe1280V is presented. Section 2 introduces the architecture of the processor. The implemented test concept is presented in Section 3. Section 4 and 5 describe the design process and implementation details of the chip. In Section 6, a AxPe-based MIMD multiprocessor system is presented.

2. ARCHITECTURE OF THE VIDEO SIGNAL PROCESSOR AXPE1280V

The compression and decompression of digital video signals according to hybrid coding schemes is performed on rectangular blocks of 8*8 or 16*16 pels. In the encoder the entire image is segmented, and low-level and medium-level coding tasks are applied on block level. After processing, a data stream for transmission to a decoder is generated. In the decoder this process is reversed and the images are reconstructed.

The most computation-intensive tasks of the video coding process are low-level tasks. Over 70% of the total computation power are consumed by this group of tasks. Because they have a convolution-like structure and a deterministic program flow, the data accesses and number of execution cycles are predetermined and so the controlling is relatively simple. Less computation-intensive are medium-level tasks. They have a less deterministic program flow with many data dependent branches, which makes parallelization on data level more difficult.

An efficient implementation of programmable processors for video coding systems can be achieved by adaptation of the processor ar-

35th Design Automation Conference ®
Copyright ©1998 ACM
1-58113-049-x-98/0006/$3.50 DAC98 - 06/98 San Francisco, CA USA

chitecture to the coding algorithms. One possible approach is the coprocessor concept. Computation-intensive operations, which are common to all coding schemes, are mapped onto a specialized coprocessor. A programmable processor controls the coprocessor and executes all other parts of the algorithms.

Based on this approach, the architecture of the video signal processor AxPe1280V (Figure 1) has been developed. The processor consists of two main modules: A RISC processor and a low-level coprocessor. Both modules are adapted to a subclass of coding tasks. The RISC processor is used for computation of medium level tasks, like quantization or Huffman coding, and performs all control tasks. For fast processing of computation-intensive, convolution-type low-level coding tasks, like discrete cosine transformation or motion estimation, the microprogrammable coprocessor is used. An I/O control unit performs all communication with external devices. It features a bidirectional 32 bit data bus as well as a 16 bit data output bus. This additional 16 bit output bus was implemented to increase the achievable communication bandwidth of the AxPe1280V and to simplify arbitration in a multiprocessor system consisting of several processors.

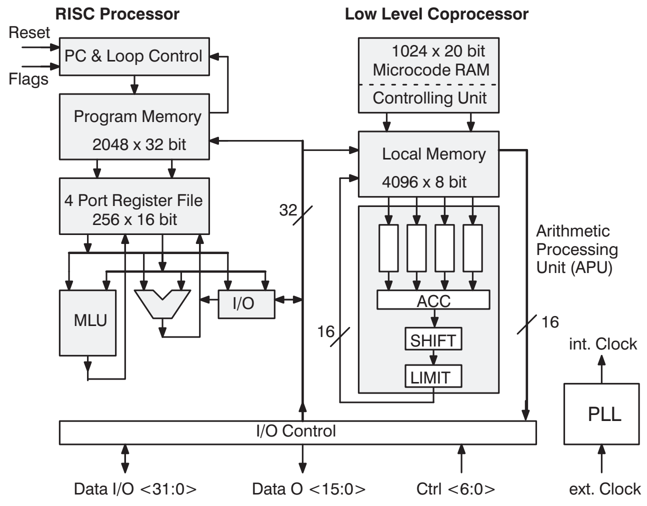

Figure 1: Architecture of the video signal processor AxPe1280V

2.1 Coprocessor

The main modules of the coprocessor are a local memory of 4096 bytes, a fourfold parallel arithmetic processing unit and a microprogrammable control unit. The local memory has been integrated to reduce transfers with external memories and to provide the arithmetic processing unit with enough bandwidth for parallel processing. The microprogrammable control unit masters the operation of the complete coprocessor.

2.1.1 *Arithmetic Processing Unit*

The architecture of the arithmetic processing unit is dedicated to the processing of convolution-type low-level tasks. To achieve a high data throughput, the inherent parallelism of these tasks must be exploited. Low-level tasks, like 2D discrete cosine transform or motion estimation access blocks of 8*8 or 16*16 pixels respectively. These tasks can be separated into 1D operations accessing neighbouring pixels. The presented arithmetic unit makes use of this characteristic by processing four pixels in four identical data paths. Each data path consists of a pipeline of subtracter, complementer, multiplier, and mode shifter. The output of the data paths is fed into a common multioperand accumulator (Figure 2).

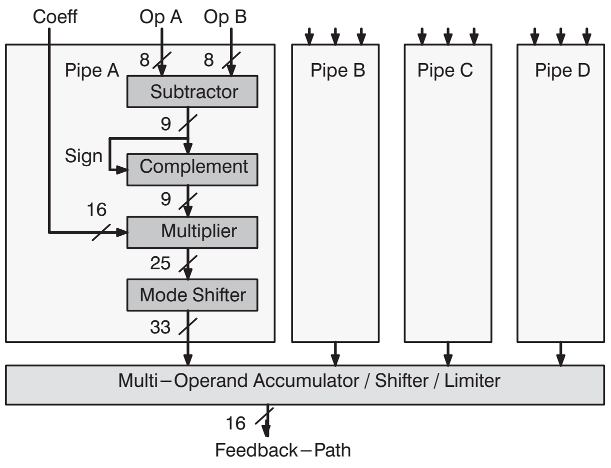

Figure 2: Architecture of the arithmetic processing unit

A generic description of all operations of the arithmetic processing unit is

$$\frac{1}{2^N}\sum \pm(A-B)*Coeff*2^s \qquad s\in(0,8)$$

The operands A and B of all data paths have an 8 bit representation. For most video coding tasks like motion estimation or FIR filtering, 8 bit arithmetic is sufficient, and therefore the data paths have been primarily adapted to these needs. Nevertheless, the mode shifters between multiplier output of each data path and inputs of the multioperand accumulator support 16 bit operations in two consecutive clock cycles. During the first clock cycle, the lower bytes of all operands are processed. The mode shifters are switched transparent. Processing of the upper bytes of the operands is performed during the following clock cycle. To adjust the results of the multipliers before accumulation, the mode shifters perform an 8 bit arithmetic left shift during this cycle.

Only two different modes - transparent mode and 8 bit arithmetic left shift - have to be provided by the mode shifters. Therefore, an implementation of the mode shifters increases the gate count of the arithmetic processing unit only marginally. More significant with respect to silicon area is, that 9x16 bit multipliers are required for the realization of 16 bit arithmetic.

Due to the different operation modes, the presented arithmetic processing unit combines fast fourfold parallel 8 bit arithmetic with high accuracy 16 bit arithmetic. Operating in 8 bit mode with a clock rate of 66 MHz, a peak performance of 1056 MOPS is achieved by the arithmetic processing unit. Because all 16 bit operations require two clock cycles, the effective parallelism in the 16 bit mode is reduced to two. Therefore, a single arithmetic processing unit provides a peak performance of 528 MOPS in 16 bit mode.

2.1.2 *Local Memory*

The local memory supports on-chip storage of input data and intermediate results of the arithmetic processing unit. Furthermore, the local memory serves as a 32 bit memory mapped interface between the coprocessor and the RISC processor core.

Three different access requests have to be served by the local memory:

- Simultaneous data in- and output via the video interface.
- Access of the RISC to the data in the local memory.
- Simultaneous access of all four data paths and storage of the results.

While the transfers of the video interface and the RISC with the local memory are 32 bit wide, the bandwidth requirements of the arithmetic unit are significantly higher. Each of the four data paths needs two operands (2*8 bit) and a coefficient of 16 bit. This leads to a read access of 128 bit. Additionally, the result of the accumulator/shifter/limiter (16 bit) has to be stored in the local memory.

The local memory was realized as dual-port RAM. To serve all memory requests, it can operate in the above mentioned three access modes. With integration of this 4096 byte local memory, parts of the processed video image can be kept locally. This is especially advantageous for motion estimation in hybrid coding applications, as it allows to carry out block matching within a search range of up to +/- 22 pixels horizontally and vertically without reloading, which leads to a significant decrease in overall processing time.

2.1.3 *Microprogrammable Control Unit*

The microprogrammable control unit features a 1024x20 bit microcode RAM and masters the operation of the local memory and the arithmetic processing unit. Each clock cycle it generates two addresses for the ports of the local memory and a control word for the arithmetic processing unit. Loop-based processing is supported by this unit, so that operations like 2-D discrete cosine transform or half-pel motion estimation can be executed on the coprocessor autonomously.

2.2 RISC Processor Core

The RISC processor core of the video signal processor is used for medium-level and control tasks. These tasks are characterized by data-dependent operations, i.e., the control flow of these tasks shows a high degree of data-dependent branches.

Several RISC cores are available for embedded applications and can be implemented as hard or soft macros. Because they are usually designed for control applications, they do not fulfill the requirements of video processing applications very well. However, controlling is a main task the RISC in the AxPe has to perform.

For the AxPe investigations were carried out to evaluate the necessary performance for medium-level processing [7]. Because the coprocessor can execute low-level tasks autonomously in parallel to the computation of medium-level tasks in the RISC, a good balance between the computation times of both units can reduce the number of stall cycles. The investigations have shown that on one hand commercially available cores offer more features, than are required. For example data caching speeds up access to large external memories, but this is not essential for hybrid video coding applications as data access patterns are known in advance.

On the other hand the performance for processing of medium-level tasks can be improved significantly by speeding up kernel routines. Especially the performance of quantization and Huffman coding can be improved by adapted execution units. Therefore, an adapted RISC core (Figure 1, left part) for the AxPe has been designed, which complements the low-level coprocessor.

The developed 32 bit RISC core is based on a harvard architecture with a 16 bit data path. The RISC processes one scalar instruction per clock cycle. Instructions are fetched from a 2048 * 32-bit on-chip program memory and executed in a four stage instruction pipeline, which consists of the stages `fetch, decode&read, execute, write`. In the `decode&read` stage operands are read from a 256 * 16 bit register file for being executed either by the ALU or by the medium-level unit (MLU). The register file can be addressed either directly or indirectly. For indirect addressing three indirect address registers are used, two for sources and one for destination. Additionally, a irregular indirect addressing scheme is supported by an implemented RAM-based address table. Because video coding schemes require a meander scan for quantization of video blocks, this feature speeds up the run level coding process.

To support data transfer between the register file and the local memory of the coprocessor, a dedicated I/O unit is used. Typically data blocks of 8x8 pixel have to be transmitted between RISC and coprocessor. Thus, this unit is optimized for block transfers. It contains registers for indirect addressing of the local memory and for controlling the coprocessor. Status information from the coprocessor is read by this module and evaluated by the RISC.

In addition to the execution of standard arithmetic and logic instructions, the ALU of the RISC processor provides special instructions for the processing of data dependent operations. These instructions, as well as the standard instructions, are executed within one clock cycle. Table 1 gives an overview of the additional ALU instructions and possible applications.

Instruction		Required for Task
$A - \|B\|$		Variable Length Coding
$A + \|B\|$		
$A - B$	$if\ (A<0)$	Quantization
$A + B$	$if\ (A>0)$	
A	$if\ (A=0)$	
$A - B$	$if\ (A<0)\ \&\ (A\ even)$	
$A + B$	$if\ (A>0)\ \&\ (A\ even)$	
A	$else$	

Table 1: Additional ALU instructions

The second execution unit in the RISC processor, the medium-level unit (MLU) has been implemented to speed up multiplications, which frequently occur in quantization and inverse quantization tasks. These tasks represent the most computation-intensive part of medium-level processing. A 17*17 multiplier for signed fixed-point multiplications and a shifter were implemented in this unit. A postponed limiter supports range control and rounding of the result and detects an overflow or underflow (Figure 3). The MLU is highly configurable and can not only be used for this special purpose.

The instruction memory of the RISC is addressed by the PC&Loop-

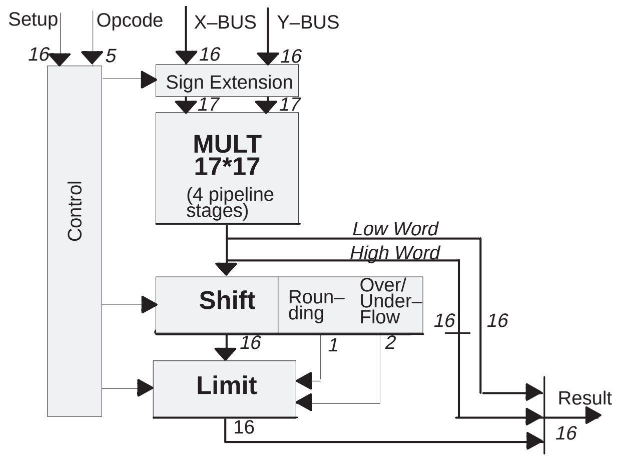

Figure 3: Architecture of the medium-level unit

Control unit, which supports hardware controlled instruction loops. To execute loops, a single register has to be initialized with the loop count by a single instruction. After invoking a hardware controlled loop, the unit performs a cyclic execution of a specific code segment without any overhead for branches. This feature reduces the amount of control instructions in medium-level tasks, which are typically based on loops of a fixed loop count.

3. INTEGRATION OF TEST STRUCTURES

The AxPe1280V contains test structures, that allow to carry out several tests on the integrated logic, memory and communication network. It features a scan path for bus tests (ScBT), which surrounds the core similar to a boundary scan path and is utilized for bus tests.

Furthermore, the AxPe1280V is equipped with a built-in self-test (BIST). The BIST is performed and controlled locally, because access to the modules from the outside is very limited and is also subject to defects. Therefore, a programmable fault-tolerant BIST controller has been integrated within the video signal processor. A scan path plus four signals secured by a parity bit are sufficient for programming and controlling the BIST controller. If the BIST controller detects an error in itself or in its control signals, or if the test of the processor fails, it moves into an isolation state. In this state the processor is prevented from accessing the output bus, the scan paths of this module are bypassed and the local clock is frozen, preventing dynamic power dissipation within the disabled processor. The isolation state can only be left by applying a reset instruction to the BIST controller. Together these measures ensure that only a sufficiently small number of potential defects remains within the chip, which could affect a multiprocessor system based on the AxPe1280V.

All logic parts of the video signal processor have been analyzed with respect to testability, which can be achieved by pseudorandom test patterns generated by BILBOs (Built-In Logic Block Observers) [8]. Selected system registers have been replaced by modified BILBOs, which — beside their normal function as registers — can perform tasks as a TPG (Test Pattern Generator), TAE (Test Answer Evaluator), or as a stage of a scan chain. BILBOs running in the TPG mode generate pseudorandom test patterns. BILBOs acting as TAEs are employed to collect and compress the test answers of the tested logic.

The control signals of all BILBOs in the RISC processor are generated by the BIST controller. All control signals of the BILBOs in the coprocessor (arithmetic processing unit, control unit, and local memory) as well as some additional coprocessor test control signals are set by the RISC processor.

The video signal processor including BIST properties has an area overhead of about 3.5% on top of a processor containing a scan path as its only test feature.

4. DESIGN PROCESS

In the beginning of this project several hybrid coding algorithms were analyzed according to the employed operations and their specific performance requirements. This led to the presented coprocessor concept. Parts of the architecture were modeled in the programming language C and extensive simulations with several filtering and coding algorithms were carried out. Bottlenecks of the initial concept were identified and eliminated.

After the detailed architecture specification was set up, the design of the video signal processor AxPe1280V was carried out by a core team of four engineers supported by several graduate students doing their project work or thesis in this project. This approach required to segment the chip design into many relatively small functional units which could be designed and simulated independently. For this reason a detailed functional and interface specification of the modules before implementation was crucial.

Most modules of the RISC processor and parts of the control logic of the coprocessor were specified in Verilog on register transfer level (RTL). For implementation of the arithmetic units like ALU, multiplier, multioperand accumulator, and barrel shifter two alternatives were considered: A standard-cell design in Verilog and synthesis or the full-custom implementation of these modules. On one hand a standard-cell implementation of these units would have reduced design time, but on the other hand would have led to a significant increase in the required silicon area for these modules. Experiments showed that a standard-cell implementation of the arithmetic would also decrease the overall performance of the video signal processor drastically compared to a full-custom implementation. For these reasons the arithmetic modules were designed especially for the AxPe1280V in full custom logic with Cadence tools. Other full-custom modules like memories and an analogue PLL were from a library developed by Philips semiconductors.

Each implemented module of the processor was simulated on functional level. After this initial verification, several modules were connected to larger units and simulated again. To integrate the instantiated full-custom modules in this simulation, behavioral Verilog models were created for these modules. More and more Verilog modules were connected together and a simulation of the complete RISC processor and coprocessor was carried out. Several test programs were developed in assembler and executed on the description of the processor. Their evaluation unveiled the remaining errors in the Verilog implementation and also helped to optimize parts of the architecture.

The verified Verilog modules on register transfer level were synthesized with Synopsys' design compiler. To achieve good synthesis results and to reduce the wiring overhead in the standard-cell layout, small modules containing not more than 2000 gates were synthesized independently. As the processor was already divided in relatively small functional blocks, no additional segmentation of these modules was necessary. The synthesized netlist of each module was again simulated with the Verilog simulator and timing was optimized.

For the final implementation, the synthesized and full-custom modules were integrated with the Cadence Design Framework II schematic tool and the test logic (see Section 3) was added. The layout of the AxPe1280V was also carried out with Cadence Design Framework II. Standard-cell logic was placed in the middle of the chip and the full-custom modules at the border. This approach was chosen, because it was not recommended to route signals over full-custom memories in this 3 metal layer process. Furthermore, a distribution of standard-cell logic over the whole chip would have caused wiring delay and additional silicon area for routing.

5. PROCESSOR IMPLEMENTATION

The processor has been fabricated in a 0.5 $\mu\mathrm{m}$ 3 metal layer CMOS technology by Philips semiconductors. The circuit consists of about 1.3 M transistors and has a die size of 82 mm^2. It is available in a QFP160 package. The power supply is 3.3 V for the core and the pad drivers. The power consumption is typically less than 0.7 W at 66 MHz. Because the complete processor is realized with static registers and memory cells it can be switched in stand-by mode. In stand-by mode the core's power consumption falls below 17 mW. This feature allows to integrate the video signal processor into mobile communication systems, e.g., notebook-based video telephones. A PLL which multiplies the external clock by a programmable factor of 1, 1.5, 2, 3, and 6 has been integrated in the AxPe1280V to allow a flexible system integration of the processor.

Extensive tests have been carried out with the AxPe1280V and have unveiled that the achievable clock rate is mainly limited by the on-chip memory in the RISC processor. With enhancements in these memories, their controlling, and by fabricating the chip in a new technology, the maximum clock rate can increase significantly.

Figure 4 shows a die microphotograph of the developed processor. The main characteristics of the chip are given in Table 2.

Technology	0.5 $\mu\mathrm{m}$ CMOS, 3 metal layer
Die Size	82 mm^2
Full-Custom Modules	RAM, Arithmetic
Standard-Cell Logic	Controlling
Transistors	1.3 M
On-Chip Memory	123 kbit
Maximum Clock Rate	66 MHz
Power Consumption	0.7 W @ 66 MHz
Computational Power	1056 MOPS
Max. Input Data Rate	132 Mbyte/s
Max. Output Data Rate	66 Mbyte/s

Table 2: Main characteristics of the AxPe1280V

6. MIMD MULTIPROCESSOR SYSTEM

For several video coding applications the performance of one AxPe1280V is sufficient. When connecting several processors to a MIMD (multiple instruction, multiple data) multiprocessor system (Figure 5), real-time processing of more complex applications

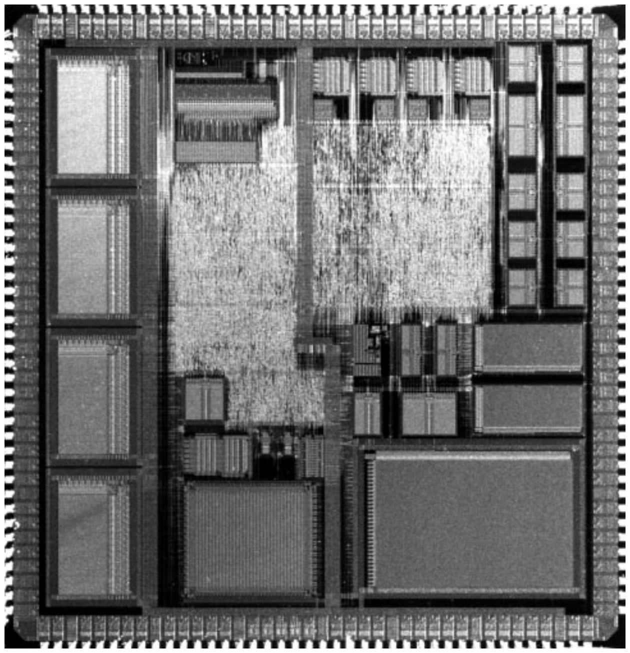
Figure 4: Die microphotograph of the video signal processor

is supported. The image data is distributed sequentially to the processors in the multiprocessor system and all processors perform the hybrid coding scheme in parallel on macro block level. Because each processor features on-chip program and data memories, the data is processed asynchronously and the overall structure of the multiprocessor system is MIMD based.

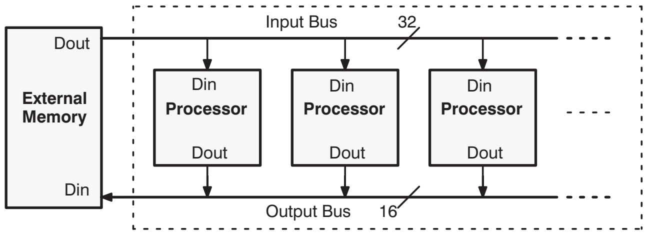

Figure 5: MIMD-based multiprocessor system

When using a SPMD scheme (single program, multiple data) a simple technique for bus arbitration can be applied (Figure 6). One processor loads and stores data from or to the external data memory at a specific period of time, while all other processors compute on locally stored data. This allows a high utilization of the buses and stall cycles can be eliminated. For block-based video coding applications like ISO MPEG-2, where the communication requirements between the processors can be neglected, the performance of the multiprocessor system is almost direct proportional to the number of processors. This speedup is only limited by the bandwidth of the input and output bus. Table 3 gives an overview on the number of processors required for different hybrid coding applications.

A multiprocessor system based on up to 6 AxPe1280V has been implemented as a PCI board for personal computers. Sample applications, that have been programmed on this system, are video coder/decoder according to ITU-T H.261 and H.263 and several proprietary video processing applications.

Application	Standard	Source rate	Number of AxPe		
			codec	encoder	decoder
Video telephone	H.261 / H.263	QCIF, 10 Hz 0.38 Mbyte/s	1	—	—
		CIF, 10 Hz 1.52 Mbyte/s	2	—	—
		CIF, 30 Hz 4.56 Mbyte/s	4	—	—
Video CD	MPEG 1	SIF, 25 Hz 3.80 Mbyte/s	—	3	2
Digital TV	MPEG 2	CCIR601, 25 Hz 20,74 Mbyte/s	—	16	6
Image coding	JPEG	CIF, 10 Hz 1.52 Mbyte/s	—	1	1

Table 3: Number of AxPe processors required for hybrid coding applications

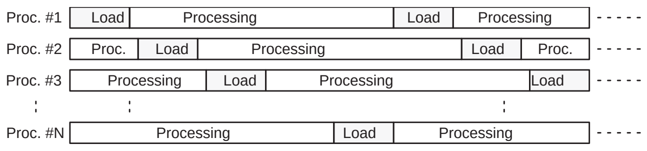

Figure 6: Loading scheme for MIMD-based multiprocessor system

The AxPe1280V has been designed to be used as building block for a monolithic multiprocessor system. Such a system with 9 AxPe640V, a previous version of the presented AxPe1280V, has already been implemented [9]. The monolithic integration is supported by the separate I/O buses of the processor and the sophisticated self-test capabilities of each processor, as discussed in Section 3.

7. CONCLUSION

The presented video signal processor AxPe1280V is optimized for block-based video coding standards like ITU-T H.261/H.263, and ISO MPEG-1/2. It consists of a RISC processor core for medium-level video processing and control tasks, and a coprocessor for computation-intensive convolution-like tasks.

The processor was designed in standard-cell logic combined with full-custom modules for arithmetic and memories, and was realized in a 0.5 μm 3 metal layer CMOS technology. It provides an arithmetic performance of more than 1 GOPS at a clock rate of 66 MHz.

For applications requiring higher computational power, the processor can be integrated into a scalable MIMD-based multiprocessor system, consisting of several AxPe1280V. For block-based video coding algorithms, almost a linear speedup can be achieved by parallelization on video data level. Such a multiprocessor system has been implemented on a PCI board for personal computers.

Current activities include the optimization of software tools for AxPe program development and the implementation of new applications on the AxPe multiprocessor system.

8. ACKNOWLEDGMENTS

The authors would like to thank Klaus Gaedke and Hartwig Jeschke for their contributions to this project. Furthermore we thank Axel Werner and Jens Castagne from Philips Semiconductors ASIC Support Centre, Hamburg, for providing full custom modules, the layout, and the fabrication of the processor. This work is supported by FhG under contract no. T/F41B/T0183/P1307.

9. REFERENCES

[1] ITU-T Recommendation H.261, ”Video Codec for audiovisual Services at p x 64 kbits”, International Telecommunication Union, March 1993

[2] Draft ITU-T Recommendation H.263, ”Video Coding for Low Bit Rate Communication”, International Telecommunication Union, December 1995

[3] ISO/IEC JTC1/SC2/WG11 MPEG-90/176 Rev. 2, 1990.

[4] ISO/IEC JTC1/SC29/WG11 CD 13818-2, 1994.

[5] H.G. Musmann, P. Pirsch, H.-J. Grallert, “Advances in Picture Coding”, Proc. IEEE, Vol. 73, No. 4, pp. 523-548, 1985.

[6] Y. Okada et. al. ”An 80 mm^2 MPEG2 Audio/Video Decode LSI”, IEEE International Solid-State Circuits Conference, Vol. 40, pp.264-265, Feb. 1997.

[7] K. Herrmann, M. Seifert, K. Gaedke, H. Jeschke, P. Pirsch,”A RISC Core for a Monolithic Video Signal Processor”, in Rabaey, J. et al., Eds., Workshop for VLSI Signal Processing, Oct. 1994, pp. 368-377, IEEE, New York, NY, 1994.

[8] B. Könemann, J. Mucha, G. Zwiehoff, “Built-In Logic Block Observation Techniques”, Proc. 1979 Intern. Test Conf., Cherry Hill, NJ, USA, October 1979, pp. 37-41.

[9] J. Otterstedt, K. Gaedke, K. Herrmann, M. Kuboschek, H.U. Schroeder, A. Werner, ” A 16 cm^2 Monolithic Multiprocessor System Integrating 9 Video Signal-Processing Elements”, IEEE International Solid-State Circuits Conference, Vol. 39, pp. 306-307, Feb. 1996.