In *IEEE Signal Processing Magazine*, vol. 14, no. 4, pp. 48–51, July 1997

On Implementation of Media Processors

Peter Pirsch, Hans-Joachim Stolberg, Yen-Kuang Chen, and S. Y. Kung
University of Hanover and Princeton University

Multimedia signal processing (MSP) involves the joint processing of digital information in various representations. It covers a very broad spectrum of applications:

- Audio & speech processing: audio compression (G.711, G.722, G.728), Dolby surrounding;
- Image & video processing: resolution conversion, image enhancement, image restoration, image compression (JBIG, JPEG), video compression (MPEG);
- Content-based indexing & retrieval: feature extraction (fillet coordination, moment, histogram), pattern recognition, face detection/recognition, fusion of multi-modality;
- 2-D, 3-D, & 4-D graphics: volume rendering, modeling transformation, texture mapping, shading, shadowing, ray-tracing, computer-assisted animation, virtual reality, etc.

In general, MSP applications demand real-time processing capabilities. From algorithmic perspectives, important characteristics of the MSP algorithms can be summarized as following:

- Intensive computation for highly regular operations;
- Intensive I/O or memory accesses, data reusability, and data locality;
- High control complexity in less computational intensive tasks;
- Frequent encounters of small integer operands.

Conventional standard processors do not correspond well to these characteristics of MSP algorithms. Therefore, special architectural approaches are necessary for multimedia processors to deliver the required high processing power with efficient use of hardware resources.

Function-specific ASICs dedicated to particular MSP subtasks can typically achieve highest performance levels at minimum cost for silicon area and optimized power consumption. However, they lack the flexibility to accommodate to algorithm modifications, and the large design effort justifies a dedicated implementation only for high-volume consumer market products of fixed functionality.

Alternatively, programmable approaches offer a higher degree of flexibility. In order to attain MSP performance, architectural strategies for programmable processors are based on parallelization and adaptation principles. Several alternatives exist to exploit the parallelization potential of MSP algorithms for programmable architectures:

- Aiming at data parallelism, SIMD (Single Instruction Stream, Multiple Data Streams) architectures are characterized by several data paths executing the same operation on different data entities in parallel. While thus a high degree of parallelism can be achieved with little control overhead, data path utilization rapidly decreases for scalar program parts. In general, pure SIMD architectures are not an efficient solution for complex multimedia applications; they are best suited for algorithms with highly regular computation patterns.
- Architectures featuring a split-ALU are based on a principle similar to SIMD: a number of lower-precision data items are processed in parallel on the same ALU. Figure 1 shows a possible implementation of the split-ALU concept. The advantage of this approach is its small incremental hardware cost provided a wide ALU is already available. However, the exploitable data parallelism decreases where processing with higher precision/word-length is required. Recent multimedia extensions of general-purpose processors are typically based on this principle.
- Instruction level parallelism is targeted by VLIW (Very Long Instruction Word) architectures, which specify several operations within a single long instruction word to be executed concurrently on multiple function units (Figure 2). In contrast to superscalar architectures, VLIW processors have to rely on static instruction scheduling performed at compile-time. The advantage is a simplified design as no hardware support for dynamic code reordering is required. In essence, VLIW architec-

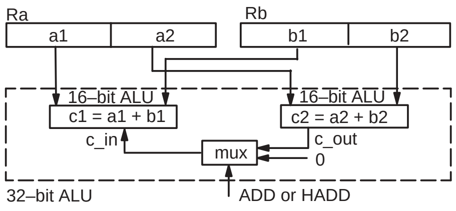

Figure 1: Split ALU implementation example. Two 16-bit ALUs can operate jointly as a combined 32-bit ALU on two 32-bit operands or separately on lower and upper 16-bit halfwords of the 32-bit operands, introducing data parallelism.

tures shift complexity from hardware to the compiler. However, run-time dependencies may be difficult to anticipate at compile-time, and worst-case assumptions have to be made for code scheduling.

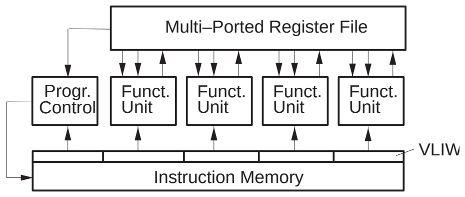

Figure 2: VLIW architecture overview. A very-long-instruction-word comprising several operation slots targets multiple function units, with a fixed assignment of operation slots to function units.

- Task level as well as data level parallelism can be exploited by MIMD (Multiple Instruction Streams, Multiple Data Streams) architectures, which are characterized by a number of parallel data paths featuring individual control units. Thus, MIMD processors offer the highest flexibility for algorithms to be executed in parallel. However, they incur high hardware cost for multiple control units as well as for a memory system delivering sufficient bandwidth to supply all required instruction streams. Furthermore, synchronization difficulties and poor programmability prevented MIMD processors from widespread use in multimedia applications so far.

Adaptation of programmable processors to special MSP algorithm characteristics include the following strategies:

- A widely employed way of adaptation is to introduce specialized instructions for frequently recurring operations of higher complexity, e.g., a multiply-accumulate operation with saturation. By replacing longer sequences of standard instructions, the use of specialized instructions may significantly reduce the instruction count, resulting in faster program execution (Figure 3). The design complexity required for implementing specialized instructions can usually be kept at modest levels; the decision about which instructions to implement depends on the probability of their use.

Operation: *r2 = sat(r2 + sat(#imm × r1))*

```
mov #imm, r3             maci #imm, r1, r2
mul r1, r3
bv _sat1
add r3, r2
bv _sat2
```

Figure 3: Specialized instructions replace sequences of standard instructions. By introducing a single new instruction comprising a frequently executed sequence of standard instructions, the instruction count of multimedia code can be reduced significantly.

- An approach combining both parallelization and adaptation principles are coprocessor architectures. By incorporating one or more separate modules adapted to specific tasks, coprocessors allow to execute highly regular program parts with high processing requirements on dedicated hardware, while more irregular but less computationally intensive control tasks are performed on a programmable processor core. Alternatively, a coprocessor may also be considered for more irregular tasks with special characteristics that cannot be executed efficiently on a general-purpose processor. Due to the adaptation, coprocessor architectures imply a tradeoff between high efficiency and flexibility.

For complex MSP systems, the inherent interaction of various design parameters comprising hardware and software issues must be taken into account. These issues are highly dependent on each other. Therefore, the system design may be best accomplished via a **codesign of algorithm/architecture**: algorithms should be partitioned to facilitate dedicated processing modules; and architectures tailored to achieve higher efficiency for the special application domain. Such a design approach naturally leads to a **coprocessor** architecture.

To help illustrate the above-mentioned **codesign methodology**, we adopt a generic architecture style as depicted in Figure 4. It comprises of **array processors** as hardware accelerators and **RISC cores** as the basis of programmable processor. In order to have an effective execution, given an application domain, the algorithms must

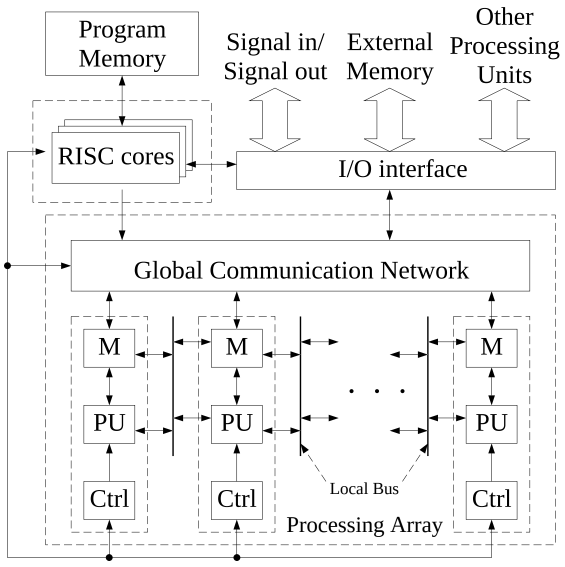

Figure 4: A generic architectural style for high performance multimedia signal processing. There are two main components: (1) **internal design space:** *RISC cores* (e.g. VLIW architecture) as the basis of programmable processor for complex but less computationally demanding operations, and (2) **external design space:** *hardware accelerators* (e.g. processor arrays) for computationally intensive but highly regular operations.

be (manually or semi-automatically) partitioned into two distinctive design spaces, denoted I and II:

1. Design space I comprises of **high-parallelism** algorithm components, i.e., *computational intensive and regular components*. For it the hardware solution is preferred.

   For the design space I, a systematic MSP mapping method can facilitate design of processor arrays for computational intensive and regular components. Massive parallel and pipelined computing engines can provide very high computational power for regular, intensive operations. Various formal systematic procedures for systolic designs of many classes of algorithms have been proposed. They transfer the computational intensive, regular operations into simple processing elements, each with a fixed, data-independent function, along with a one- or two-dimensional nearest-neighbor communication pattern. These are the basic components of our design methodology for the MSP system.

2. Design space II comprises of **high-versatility** algorithm components, i.e., *complex but less computational intensive components*. For it the software solution is preferred.

   For the design space II, the complex but less computational intensive components (e.g. controlling, data-dependent tasks) are supported by the software solution on RISC cores. Minor adaptations to improved multimedia processing algorithms can be achieved by software updates.

The overall architectural design can be divided into *internal* and *external* design spaces, as illustrated in Figure 4. The *internal* design focuses on core processor upgrade, while the *external* design focuses on *accelerators* that off-load task from the main core. While the external design lies exclusively in the design space I (e.g. parallel SIMD or systolic arrays), the design of internal core processor involves both design space I (e.g. fine-grain VLIW parallelism) and space II (e.g. expanded core datapath for supporting multimedia extension).

The rapid progress in VLSI technology will soon reach more than 100 million transistors in a chip, implying tremendous computation power for many multimedia applications. The silicon area required for implementing a specific function will decrease considerably, and a higher functionality can be realized on a single chip. For example, Chromatic's `Mpact` media processor. This trend leads to the monolithic integration of programmable processor cores, function-specific modules, and various system interfaces in order to enable high multimedia functionality at decreased system design costs.

In summary, the future MSP implementation hinges upon an optimal tradeoff between the two design spaces, which can be effectively addressed by a codesign approach. In the codesign process, the degree of flexibility demanded by the less predictable algorithm components competes with the high efficiency of systematically derived dedicated approaches for regular, fine-granularity components. The optimum partition has to be determined individually for the targeted application domain. The best architectural mix depends on the characteristics of the algorithms to be implemented and on the targeted application spectrum. In some instance, the best algorithms can be designed only by fully taking into account the architectural characteristics. For example, Philips Electronics presents a single chip MPEG2 video encoder, `I.McIC`. In order to enable digital recording at the latest consumer storage media, such as D-VHS and DVD, this cost-effective single-chip solution operates in MPEG2 ML@SP mode only. In the future, VLSI technology progress allows the economic implementation of architectures with higher flexibility, increasingly shifting the weight from more regular to high-complexity applications.

References

[1] *IEEE Micro: Special Issue on Media Processing*, vol. 16, no. 4, Aug. 1996.

[2] *Proceedings of International Solid-State Circuits Conference (Conference Theme: Multimedia)*, Feb. 1997.

[3] *Multimedia Hardware Architectures 1997*, Proc. SPIE 3021, Feb. 1997.

[4] V. Bhaskaran, K. Konstantinides, R. B. Lee, and J. P. Beck, "Algorithmical and Architectural Enhancements for Real-Time MPEG-1 Decoding on a General Purpose RISC Workstation," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 5, no. 5, pp. 380-386, Oct. 1995.

[5] S. Dutta, A. Wolfe, W. Wolf, and K. J. O'Connor, "Design Issues for Very-Long-Instruction-Word VLSI Video Signal Processors," in *Proceedings of IEEE Workshop on VLSI Signal Processing*, pp. 95-104, Oct. 1996.

[6] R. P. Kleihorst, A. van der Werf, W. H. A. Brüls, W. F. J. Verhaegh, and E. Waterlander, "MPEG2 Video Encoding in Consumer Electronics," *Journal of VLSI Signal Processing Systems*, vol. 17, Dec. 1997.

[7] S. Y. Kung and Y.-K. Chen, "On Architectural Styles for Multimedia Signal Processors," in *Proceedings of IEEE Workshop on Multimedia Signal Processing*, (Princeton, NJ), June 1997.

[8] K. Nadehara, I. Kuroda, M. Daito, and T. Nakayama, "Low-Power Multimedia RISC," *IEEE Micro*, vol. 15, no. 6, pp. 20–29, Dec. 1995.

[9] T. Nishitani, P. H. Ang, and F. Catthoor, eds., *VLSI Video/Image Signal Processing*, (Norwell, MA), Kluwer Academic Publishers, 1993.

[10] P. Pirsch, ed., *VLSI Implementations for Image Communication*, (Amsterdam, Netherlands), Elsevier, 1993.

[11] P. Pirsch, N. Demassieux, and W. Gehrke, "VLSI Architectures for Video Compression–A Survey," *Proceedings of the IEEE*, vol. 83, no. 2, pp. 220–246, Feb. 1995.

[12] J. L. van Meerbergen, P. E. R. Lippens, W. F. J. Verhaegh, and A. van der Werf, "PHIDEO: High-Level Synthesis for High Throughput Applications," *Journal of VLSI Signal Processing*, vol. 9, no. 1-2, pp. 89–104, Jan. 1995.