A Comparison Between Processor Architectures for Multimedia Applications

Asadollah Shahbahrami Ben Juurlink Stamatis Vassiliadis
Computer Engineering Laboratory
Faculty of Electrical Engineering, Mathematics, and Computer Science
Delft University of Technology
The Netherlands
Phone: +31 15 2787362. Fax: +31 15 2784898.
E-mail: {shahbahrami,benj,stamatis}@ce.et.tudelft.nl.

Abstract— **The efficient processing of MultiMedia Applications (MMAs) is currently one of the main bottlenecks in the media processing field. Many architectures have been proposed for processing MMAs such as VLIW, superscalar (general-purpose processor enhanced with a multimedia extension such as MMX), vector architectures, SIMD architectures, and reconfigurable computing devices. The question then arises: which architecture can exploit the characteristic features of MMAs the most? In this paper, first, we explain the characteristics of MMAs, after that we discuss the different architectures that have been proposed for processing MMAs. Subsequently, they are compared based on their ability to exploit the characteristics of MMAs. Superscalar processors with dynamic out-of-order scheduling provide higher performance than VLIW processors and than superscalar processors with in-order scheduling. Because superscalar architectures include complicated control logic for out-of-order execution, and because VLIW processors have to decode every instruction slot in parallel and need a register file with multiple read and write ports, they are more complex than single-issue vector architectures.**

Keywords: media processing, VLIW, SIMD, multimedia extensions.

I. Introduction

A variety of multimedia processing algorithms are used in media processing environments for capturing, manipulating, storing, and transmitting multimedia objects such as text, handwritten data, 2D/3D graphics, and audio objects. Computer applications are becoming multimedia-rich and the World Wide Web will make future applications contain even more multimedia objects [27], [61], [48]. Multimedia standards such as MPEG-1, MPEG-2, MPEG-4, MPEG-7, JPEG2000, and H.263 put challenges on both hardware architectures and software algorithms for executing different multimedia processing jobs in real-time, because each media in a multimedia environment needs different algorithms, processes, and techniques [47], [43], [13].

The efficient processing of MultiMedia Applications (MMAs) is currently one of the main bottlenecks in the media processing field. To understand the need for new processing for supporting emerging and next generation of multimedia data, it is necessary to first understand the limitations of the current architectural of support. Recently, different architectures have been proposed for MMAs processing. Designs of these architectures ranges from fully custom to fully programmable dedicated architectures, and to General-Purpose Processors (GPPs) with multimedia extensions.

Architectural support for multimedia applications has been classified in three categories by Fritts [23]: application-specific processors, multimedia extensions to a GPP, and media processors. Talla [64] has distinguished those common approaches for handling multimedia application namely, GPPs with Single Instruction Multiple Data (SIMD), Very Long Instruction Word (VLIW) media processing, and Application Specific Integrated Circuits (ASICs). The question then arises: what architecture will be able to best exploit the inherent features of MMAs?

The purpose of this paper is to provide an overview of recent architectural approaches for multimedia processing ranging from dedicated multimedia processor to programmable processor and GPPs with multimedia extensions. We will also present a comparison between architectures that have been used for designing multimedia processors. This paper is organized as follow. Section II presents an overview of multimedia characteristics. Section III describes different classifications of processors that, have been proposed for processing MMAs and describes their advantages and disadvantages. Additionally, different GPPs with MultiMedia eXtensions (MMX) are evaluated and advantages and disadvantages of such systems distinguished for processing of MMAs are given. In Section IV programmable multimedia processor are described and evaluated. Finally, conclusions are stated in Section V.

Operation type	Percentage
ALU	40%
Load/Store	26-27%
Branch	20%
Shift	10%
Integer Mult.	2%
Floating point	3-4%

TABLE I
OPERATION DISTRIBUTION IN MMAS.

Name of Resource	Required Number
Int. ALU	4
Memory Units	2
Branch Unit	1
Shifters	1
Floating P. U	1
Int. Multipliers	1

TABLE II
PROPER RATIO OF FUNCTIONAL RESOURCES FOR PROCESSING MMAS.

Operand size	Usage Frequency
8-bit	40%
16-bit	51%
32-bit	9%

TABLE III
DIFFERENT DATA TYPES IN MMAS.

II. CHARACTERISTICS OF MULTIMEDIA APPLICATIONS

MMAs have many characteristics that make them unique from General-Purpose Applications (GPAs). The most important ones are the following [15], [58], [2], [3]:

- Real-time response: MMAs such as video conferencing and electronic commerce often require real-time response. In addition, they require a certain quality of service.
- Processing of streaming data: MMAs can keep their instruction code on-chip and commonly stream data in from off-chip.
- Significant fine and coarse grained data parallelism: Typically, MMAs perform the same operations on different data item (e.g., pixels). In addition, many functions need to be performed on these data values. Since these operations and functions are largely independent, it is possible to exploit SIMD and Thread-Level Parallelism (TLP).
- Considerable data reorganization: In addition to the SIMD nature of multimedia processing, most applications also need to be able to reorganize the individual data components efficiently to adjust for various data stream layouts. Therefore, MMAs are not well suited for traditional SIMD architectures where data reorganization can be expensive.
- Small loops: MMAs spend nearly 95% of their execution time over the two innermost loops. These loops tend to have a large number of iterations, typically 10 or more, with some loops having hundreds or thousands of iterations [23], [25].
- High memory bandwidth requirement: The applications process large data sets, putting a severe burden on memory system.
- Small data types: MMAs typically use small integer data types of 16 bits or less.

Additionally, MMAs perform significantly more arithmetic operations than GPAs. Table I taken from [26], depicts the operations distribution that are needed to implement MMAs efficiently. As can be seen in this table MMAs more perform integer arithmetic operations than floating-point operations.

Table II desired ratio of functional resources for MMAs is also shown. As it is seen in this table the proper number of integer ALUs and floating-point units for MMAs is 4 and 1, respectively [26].

Furthermore, the most important data types in MMAs are usually small such as 8 and 16 bits. The data characteristics of MMAs are important for two reasons: First, subword parallelism uses small data types to exploit SIMD parallelism. Second, media processors will have narrower datapaths than GPPs because the data types are smaller [62], [39], [50]. Most instructions process integers of only 8 or 16 bits. The only major exception to this is graphics applications, which process mainly single-precision floating-point values.

Table III depicts the distribution of operand size used in MMAs. It shows that most operants are smaller than 16 bits. The memory behavior of MMAs is different from that of GPAs. Most importantly, MMAs have high spatial locality but little temporal locality. Typically, the processor loads a small amount of data, processes it, and it never or rarely re-uses the data again.

Based on these characteristics, MMAs require different architectures than GPAs. So the architectures that have been proposed for processing MMAs are investigated in the following section.

III. PROCESSOR ARCHITECTURES TO SUPPORT MMAS

During the last few years we have been witnessing a change of architectures for MMAs from fully custom to fully programmable dedicated architectures [58], [3], [38],

[37]. A number of Programmable Digital Signal Processors (PDSP) have been used since 1980. They supported specific instructions such as MAC (multiply-and-accumulate), REPEAT, and so on, in the instruction set in order to improve both performance and programmability. Hen [31] has given summary of characteristics of early PDSPs as well as recent PDSPs. Hen classified them based on different implementation methods into four groups, PDSP chip, PDSP core, multimedia PDSPs, and native signal processing (NSP) instruction set processors. Multimedia PDSPs are specifically designed for audio/video applications. One example of this group is the Trimedia TM 1300 [17], [56]. NSP processors extend the instruction set of a GPP to process multimedia data.

Dasu [13] has classified existing media processing strategies according to either the evolution of processing architectures or the functionality of the processors. Based on the evolution of the processing architectures, existing architectures for media processing can be classified as programmable processors, dedicated implementations, and reconfigurable processors. Features such as parallelism, flexibility, and memory intensive data processing allow media processing architectures to be classified based on the approach used to exploit parallelism, exploiting the iterative nature of operations, and reduction of off-chip memory transactions.

Panchanathan [52] and Dasu, and Panchanathan [14] have indicated that there are two approaches in multimedia processor design namely: dedicated and programmable. Another, taxonomy of multimedia processing approaches has been done by Ahmed [18]. This classification is based on three established architecture models: vector processors, superscalar processors, and multimedia processors. Borko [27] has classified processor architectures that have been designed to support MMAs from fully custom to fully programmable dedicated architectures and to GPPs with a multimedia extension. His classification is illustrated in Figure 1. We discuss this classification in more detail in the following sections.

A. Dedicated Multimedia Processors (DMPs)

DMPs are typically custom designed architectures intended to perform specific multimedia functions such as video and audio compression and decompression, and 2D and 3D graphics applications. DMPs use a variety of architectural schemes, ranging from multiple functional units with a RISC core processor to multiprocessors. The most recent dedicated processors use SIMD and VLIW architectures. According to [9] dedicated implementations could become the best selection based on: available technologies, required computational bandwidth, and the target algorithm. Designs of dedicated multimedia processors range from fully custom architectures, with minimal programmability, referred to as function-oriented architectures to fully programmable architectures.

A.1 Function-Oriented Architectures

A function specific implementation is a direct mapping of the multimedia processing tasks to hardware implementation optimized to execute the specific functions. Matching of the individual hardware modules to the processing requirements results in area efficient implementations. Efficiency and speed are typically better than that dedicated by programmable architectures. The general design theme for these types of processors consists of using a RISC core processor for main control, and special hardware accelerators for specific multimedia algorithms such as Discrete Cosine Transform (DCT), quantization, entropy encoding, and motion estimation. Some characteristics of this DSP processors are [21]:

- Multiple data and instruction buses.
- Parallel execution of MAC operation.
- Limited number of instructions.
- Efficient loop control.
- Low-cost alternatives for specific applications.
- DSPs are optimized for performing regular, predictable, computation-intensive tasks.

The advantages of this approach are the following. First, the hardware overhead for control is minimized. Second, the power consumption can be kept low. Third, DSPs achieve high performance for some applications. The disadvantage of this kind of architectures is that they are only suitable for specific functions and later extensions are not possible without a redesign of hardware. Therefore, their flexibility and adaptability to new applications is very limited. An example of a dedicated single-chip implementation of an MPEG-2 video encoder is shown in Figure 2.

A.2 Programmable Architectures

Programmable architectures for processing MMAs can be divided into flexible programmable architectures, which provide moderate to high flexibility, and adapted programmable architectures, which provide higher efficiency but less flexibility. There are some approaches in the design of programmable architectures such as Data-Level Parallelism (DLP), Instruction-Level Parallelism (ILP) [19], and TLP or adaptation to special algorithm characteristics by implementing specialized instructions and dedicated hardware modules that result in higher efficiency for a limited application field.

Programmable architectures have many advantages such as flexibility, high performance, supporting a com-

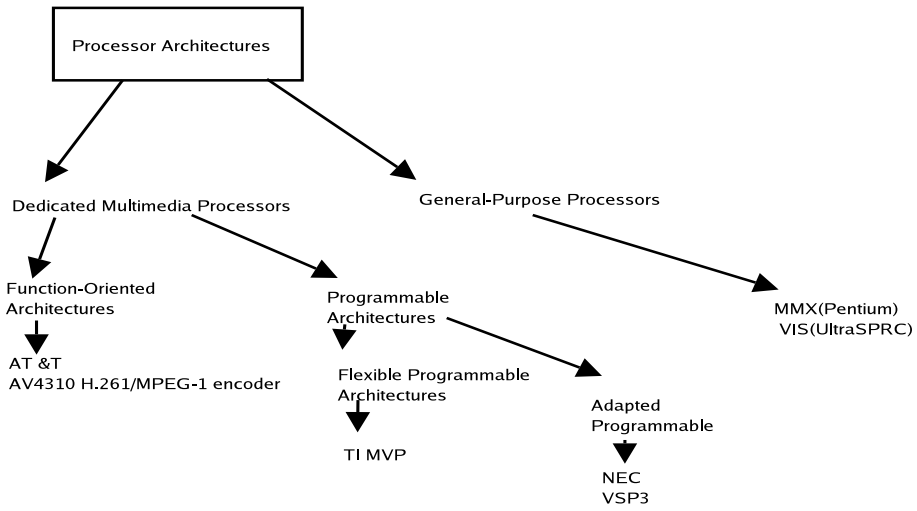

Fig. 1. Classifi cation of processor architectures that support MMAs.

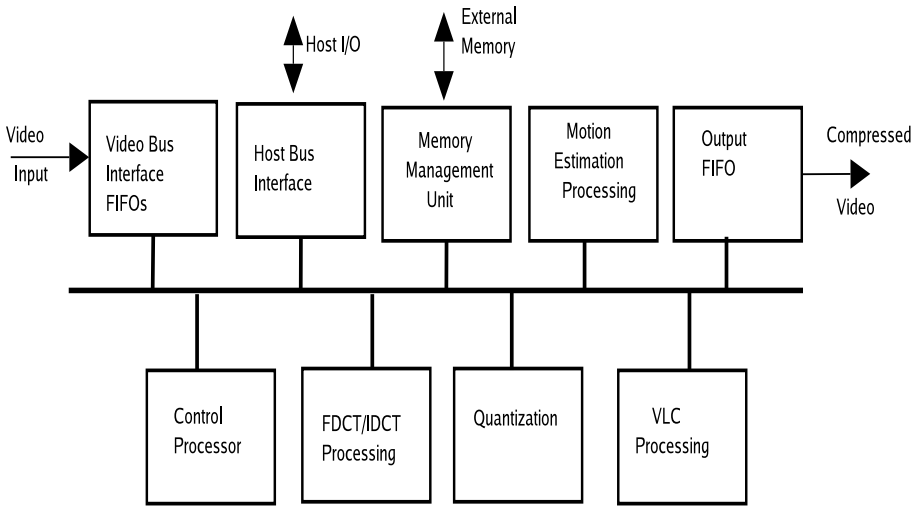

Fig. 2. Block diagram of a typical function specifi c architecture for a video encoder.

plete multimedia environment, and cost-effectiveness. Additionally, programmable architectures can implement different tasks controlled by software. The main advantage of programmable architectures is the increased functionality. However, hardware for control and program storage and software development time are their disadvantages.

An example of flexible programmable architectures is the Multimedia Video Processor (MVP) [29]. Figure 3 shows a block diagram of its architecture. The MVP combines a 32-bit RISC master processor and four 32-bit DSP processor in a crossbar-based SIMD shared-memory architecture. The RISC master processor can be used for control, floating-point operations , audio processing, and 3D graphics.

A.3 Adapted Programmable Architectures (APAs)

APAs provide increased efficiency by adapting the architecture to the specific requirements of video coding applications. These architectures provide dedicated modules for several tasks of the video coding/decoding (codec) algorithm such as DCT module, or Variable Length Coding (VLC). One example of this group is the VideoRISC Processor (VRP) from C-cube Microsystem. Its architecture is depicted in Figure 4. The VRP consists of a 32-bit RISC processor and two special functional units for VLC and motion estimation. Specially designed instructions in the RISC processor provide an efficient implementation of the DCT and other video-related operations. The VRP can perform real-time MPEG-1 encoding and decoding.

Advanced dedicated multimedia processors use SIMD and VLIW architectural schemes and variations to exploit

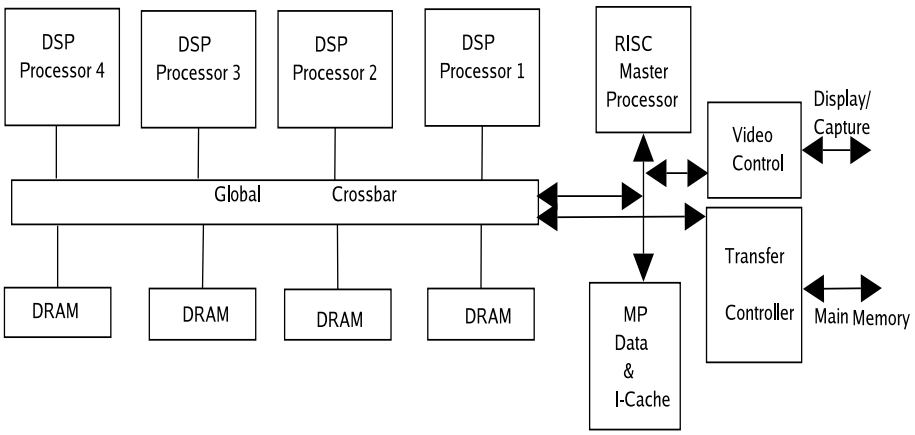

Fig. 3. Block diagram of the multimedia video processor with four DSPs and a Master Processor (MP).

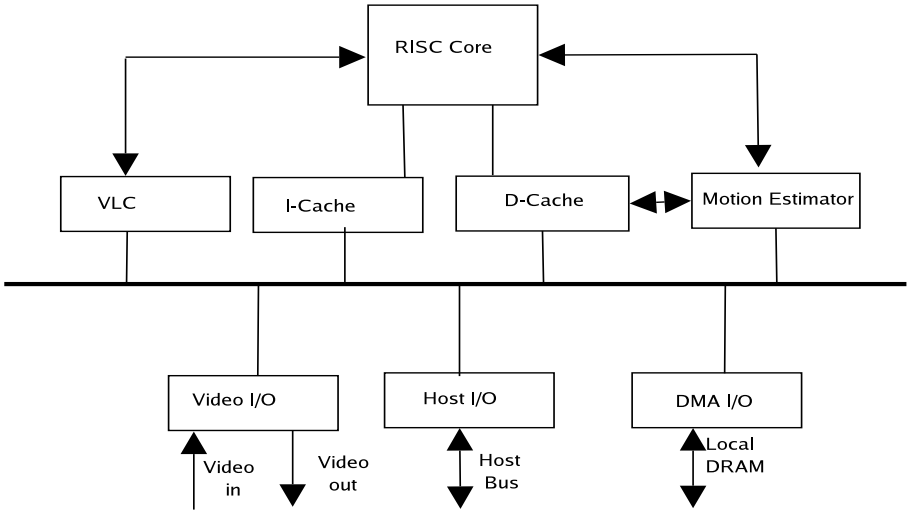

Fig. 4. Block diagram of C-cube's VideoRISC processor which has an adapted programmable architecture.

a high degree of parallelism [35].

A.4 VLIW

A VLIW processor has a multiple independent function units and executes several operations in parallel. These operations are placed in a very long instruction word. It is the compiler's responsibility to find independent instructions that can be grouped in a VLIW. VLIW processors can achieve high performance by utilizing ILP and DLP. A typical VLIW architecture is shown in Figure 5. In this figure, where an instruction contains three opcodes, and three registers for each operation, two sources and one destination. Different fields of the long instruction word contain the operations that activate different functional units. VLIW architectures have many advantages. For example, VLIW processors employ static instruction scheduling performed at compile-time rather than dynamic scheduling performed at run-time as in superscalar processors which requires much more hardware. Furthermore, packing and unpacking overhead is minimal. Since data items maintain their individual identity, it is not required to always treat them as a group [21].

These architectures also have many disadvantages. In particular, the burden of operation scheduling is on by compiler, and the resulting utilization and the obtained parallelism fundamentally depend on the available compiler technology. Furthermore, all operations specified within a VLIW instruction must be independent. VLIW functional hardware is more expensive. While a packed SIMD architecture exploits the same ALU hardware to execute multiple operations on subwords, a VLIW requires multiple functional units with full precision to achieve the same

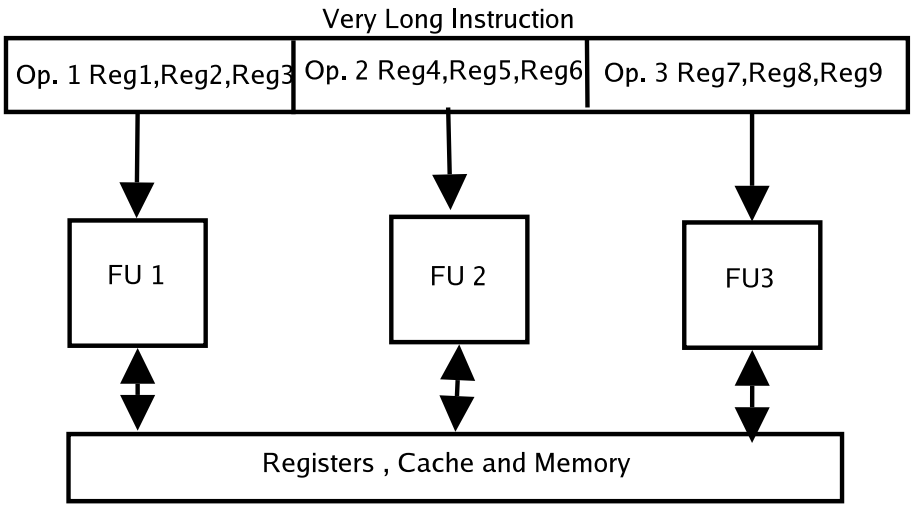

Fig. 5. Block diagram of a typical VLIW processor with 3 Functional Units (FU).

level of parallelism.

In addition, VLIW register connectivity is more expensive than that of an SIMD. A packed SIMD architecture uses a single register file port to read or write multiple values, but a VLIW requires multiple ports, one per functional unit. This is due to the fact that a VLIW architecture with N 2-input/1-output functional units requires reading $2N$ values and writing N values per cycle. For a single register file, this means that $2N$ read ports and N write ports are needed.

The Philip's TM1000 [17], [56] contains a VLIW processor, as well as a video and audio I/O subsystem. The processor has an instruction set that is optimized for processing audio, video, and graphics. A block diagram is depicted in Figure 6. The TriMedia TM-1000 is a general-purpose microprocessor for real-time processing of audio, video, graphics, and communications data streams. The key features of the Trimedia TM1000 are the following: 16 KB data cache, 32 KB instruction cache, 27 functional units, 2 DSP ALUs that can each perform either 32-bit or 4 x 8-bit or 2 x 16-bit partitioned arithmetic operations. 2 DSP multipliers that can execute two 16-bit or four 8-bit multiplications per cycle and, it can issue 5 instructions to 5 out of the 27 FUs per cycle (i.e., there are 5 slots in each VLIW).

B. General-Purpose Processor (GPPs)

In order to enhance the performance of MMAs, GPP vendors have extended their ISAs. These ISA extension operate in a SIMD fashion to exploit the DLP present in MMAs. GPPs apply the MicroSIMD [46] approach by sharing their existing integer or floating-point data paths. The goal in designing SIMD media ISA extensions for GPP has usually been to utilize Subword Level Parallelism (SLP) with existing hardware and without sacrificing the general-purpose nature of the processor. Subword parallelism provides a very low-cost form of small-scale SIMD parallelism in a word-oriented processor. A word-wide integer functional unit can be partitioned into parallel subword units, with small hardware overhead. As illustrated in Figure 7, a 64-bit adder may be partitioned into four 16-bit adders. Such a partitionable adder allows four 16-bit additions, or a single 64-bit addition, to be performed in a single cycle. The overhead cost is very small, since the same datapaths are used in either case: two 64-bit registers read and one register write. A processor with two 64-bit partitionable ALUs could support eight parallel 16-bit operations with just a 6-ported (4 read and 2 write ports) register file, while a processor with eight independent 16-bit functional units requires a 24-ported register file.

Subword parallelism is a form of vector processing. A register is viewed as a small vector with elements that are smaller than the register size. This requires small data types and large register sizes. Multimedia kernels process small data types and the registers of GPPs satisfying these requirements. In particular, the double-precision FP registers can hold several of such elements. The same operation is applied to the subwords at the same time. The SLP is a cost-effective solution to exploit the DLP present in MMAs. There is no need to replicate the functional units and the memory port can supply multiple elements at no cost. Initial implementations of GPPs with multimedia extensions are Intel's MMX [55], [54], Sun's Visual Instruction Set (VIS) [68], Compaq's Motion Video Instructions (MVI) [4], MIPS Digital Media eXtension (MDMX) [49], and HP's Multimedia Acceleration eXten-

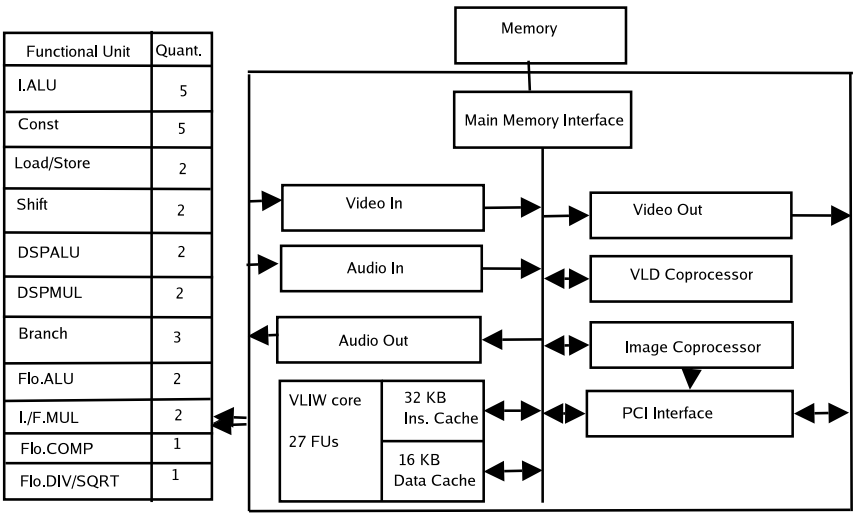

Fig. 6. Block diagram of the Philip's Trimedia TM1000.

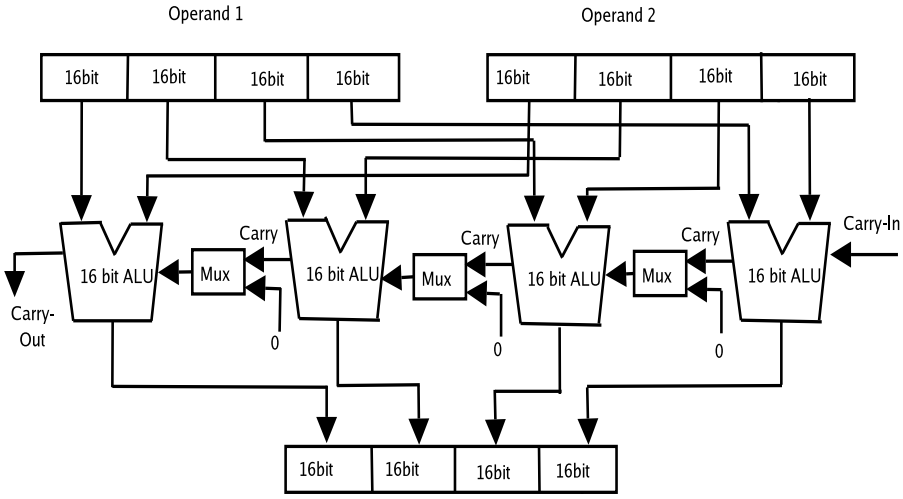

Fig. 7. MicroSIMD Parallelism uses packed data types and a partitionable ALU.

sion (MAX) [44], [45]. These extensions supported only integer data types and were introduced in the mid-1990's. 3DNow [1] was the first to support floating-point media instructions. It was followed by Streaming SIMD Extension (SSE) and SSE2 from Intel [59], [67]. Motorola's AltiVec [16] supports integer as well as floating-point media instructions.

The main differences between these processors are in the way that they reconfigure the internal register file structure to accommodate SIMD operations, and the multimedia instructions they choose to add. Multimedia instruction sets can be broadly categorized according to the location and geometry of the register file upon which SIMD instructions operate. The alternatives are reusing the existing integer or floating point register files, or implementing an entirely separate one. The type of register file affects the width and therefore the number of packed elements that can be operated on simultaneously (vector length). Despite the similarities, each approach to subword extensions is unique [32]. Key differences include the amount of additional hardware required, ranging from MAX-2, which reuses the integer registers and execution units and requires virtually no additional execution hardware, to AltiVec, which requires an entirely new execution unit.

In Table IV common and distinguishing features of available GPPs with multimedia instruction set extensions are summarized [3], [31], [63], [22].

The most important features of some GPPs with specialized media instruction set extensions are:

- The processors issue and execute two or more multimedia instructions per cycle.
- These processors issue and execute instructions out-of-

GPP with Multimedia exten.								
ISA Name	AltiVec	MAX-1/2	MDMX	MMX/ 3DNow	VIS	MMX/ SIMD	SSE	SSE2
Company	Motorola	HP	MIPS	AMD	Sun	Intel	Intel	Intel
Instruction Set	Power PC	PARISC2	MIPS-V	IA32	P. V.9	IA32	IA64	IA64
Processor	MPC7400	PA RISC	R1000 PA8000	K6-2	Ultra Sparc	Pentium2	P.3	P.4
Date	1999	1995	1997	1999	1995	1997	1999	2000
Datapath	128-bit	64-bit	64-bit	64-bit	64-bit	64-bit	128-bit	128-bit
Size of Reg. File	32x128b	(31) /32x64b	32x64b	8x64b	32x64b	8x64b	8x128b	8x128b
Shared with	Dedicated	Int. Reg.	FP Reg.	Dedicated	FP Reg.	FP Reg.	Dedicated	Dedicated
Int. data types								
8-bit	16		8	8	8	8	16	16
16-bit	8	4	4	4	4	4	8	8
32-bit	4			2	2	2	4	4
64-bit							2	2
Int. Arith.								
Shift Right/Left	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Mul.-add	Yes	No	No	Yes	Yes	Yes	Yes	Yes
Shift-Add	No	Yes	No	No	No	No	No	No
Floating Point	Yes	No	Yes	Yes	No	No	Yes	Yes
Single Precision	4x32		2x32	4x16 2x32			4x32	
Double Precision				1x64				2x64
Accumulator	No	No	1x192b	No	No	No	No	No
Num. of Ins.	162	(9) 8	74	24	121	57	70	144
Num. of operands	3	3	3-4	2	3	2	2	2
Sum of Abs. Diff.	No	No	No	Yes	Yes	No	Yes	Yes
Modulo Add/Sub	8, 16 32	16	8, 16	8, 16 32	16, 32	8, 16 32, 64	8, 16 32, 64	8, 16 32,64
Satura. Add/Sub	U8, U16, U32 S8, S16, S32	U16, S16	S16	U8, U16 S8, S16	No	U8, U16 S8, S16	U8, U16 S8, S16	U8, U16 S8, S16

TABLE IV

SUMMARY OF AVAILABLE MULTIMEDIA EXTENSION WITH GPP (SN AND UN INDICATE N-BIT SIGNED AND UNSIGNED INTEGER PACKED ELEMENTS, RESPECTIVELY. WHILE VALUES WITHOUT A PREFIX U OR S IN LAST ROW, N, INDICATE OPERATIONS WORK FOR EITHER SIGNED OR UNSIGNED VALUES).

order. To do so, they require a substantial amount of hardware.

- These hardware components occupy a large portion of the silicon area and contribute significantly to the power dissipation.
- These processors employ a dynamic branch prediction technique.
- The cache mechanism is designed to exploit 1D locality of consecutive addresses, but media applications require multidimensional locality of accesses.
- The word lengths of these processors is 32 or 64 bits. But the word lengths needed for multimedia applications are typically 8 or 16 bits.
- They implement in excess of 15 million transistors on a chip.

The SIMD efficiency of these processors is often low, because the overhead/supporting instructions dominate the dynamic instruction stream [66]. Additionally, the execution time is also increased because of conventional architectural limitations such as cache misses, resource stalls, and branch misspeculations. Multimedia extensions have proven to provide significant performance benefits by exploiting the DLP present in multimedia codes. However, these GPPs equipped with multimedia extensions have the following limitations:

- Memory misalignment problems: The nature of subword data introduces memory misalignment problems. Accessing data that is not aligned requires extra instructions.
- Mismatch between storage and computational formats: The computational format is usually larger than the storage format.
- Limitation on the amount of parallelism: The fixed size of the multimedia registers limits the amount of paral-

lelism that can be exploited by a single instruction to at most 8 (VIS, MMX) or 16 (SSE, AltiVec) parallel operations, while more parallelism is present in MMAs.

- Overhead instructions: Implementations of multimedia kernels with short-vector SIMD extensions require a significant amount of overhead for converting between different packed data types and for data alignment, increasing the instruction count. For VIS up to 41% of the total instruction count constitutes overhead [60].
- No strided memory accesses: Most GPPs can only perform stride-1 memory accesses. It is therefore, inefficient to access, for example, a column of a matrix.
- Scalability: The scalability of subword parallel processors cannot be achieved by simply increasing the machine word size. It can be argued that subword processing is a low cost addition to GPPs to exploit the wide datapaths. However, reversing the argument may not be valid. Scaling the performance by doubling the machine word size would require increasing the data path size beyond that required for other GPAs.

Additionally, the main drawback of the multimedia instruction sets is that they suffer from a lack of compiler support [5]. This limits developers to using in-line assembly macros and low-level library calls. Table V (taken from [31]) compares different solutions for multimedia processing.

B.1 Efficiency of microSIMD architectures

MicroSIMD architectures support the subword parallelism defined in the multimedia extensions for GPPs. They are more efficient than other parallel architectures for the design of media processors. A Multiple Instruction, Multiple Data (MIMD) architecture consists of multiple processors, and each processor can execute a different instruction in each cycle. Each processor has its own register file. For four processors, four instructions must be issued. Some sort of interconnection network between the register files is needed to move data between the processors. A SIMD has the same datapaths as a MIMD architecture, except that a single instruction is issued to all the processors in a cycle. In superscalar processor, the register file is shared between M parallel functional units. In each cycle, N different instructions are issued, where $N \leq M$. VLIW architectures look like a superscalar architecture except that only a single instruction is issued each cycle. However, this single instruction consists of up to M different operations, one for each of the parallel functional units [46].

The parallel functional unit approach of superscalar and VLIW architectures is more efficient than the parallel processor approach of MIMD and SIMD. Sharing the register file reduces the overhead of having to move data between the register files via an interconnection network or crossbar switch. The microSIMD approach is more efficient than the superscalar and VLIW architectures, because the register file has been simplified considerably without losing any parallelism.

Both microSIMD and SIMD architectures require only one instruction, whereas MIMD multiprocessors and superscalar processors require n instructions for n-way parallelism. While VLIW architectures require only one instruction, this contains n different operation fields. The reduced number of instructions of microSIMD architectures, or reduced code size with respect to VLIW architectures, is a cost reduction since it reduces the instruction memory requirements. It is also a performance benefit, since potential cache misses during instruction fetches are also reduced. Figure 8 shows the number of instructions that need to be issued in order to achieve the same degree of parallelism in the different parallel architectures.

Let us assume that each architecture supports 4-way parallelism and that require designators are 5 bits wide, allowing 32 registers to be addressed. Table VI depicts the area requirements of the register files of the different architectures. The MIMD and SIMD architectures both have four register files, with 128 registers in total, and each register capable of holding a 16-bit operand. Their area requirements are proportional to the total number of bits in all four register files, with an overhead of d per register file, and an addressing overhead of e per register. The microSIMD architecture can hold the same number of 16-bit operands in one quarter the number of registers, since these are packed as four 16-bit subwords per 64-bit register. Hence, it has slightly less area requirements due to lower area overhead for the registers and register file than the MIMD or SIMD architectures.

A microSIMD architecture is less flexible than a MIMD, superscalar or VLIW architecture, however, because the same operation needs to be performed on every data element. Nevertheless, because pixel-oriented computations exhibit high degrees of data parallelism, filling four or eight subword parallel slots for microSIMD execution is easy, and linear [46].

B.2 Comparison Between Static and Dynamic Scheduling

A comparison between static and dynamic scheduling on the three basic architectures, out-of-order superscalar, in-order superscalar, and VLIW processor has been given in [24], [65]. The same conclusions that can be drawn from these studies are:

- Static scheduling performs nearly as well as dynamic in-order scheduling for media processing, with average In-

Solution	Performance	Flexibility	Power	Cost	Density
ASIC	High	Low	Low	Low	Medium
Programmable architecture	Medium	High	Medium	High	Medium
GPPs with multimedia extension	Low	High	Medium	High	High

TABLE V
COMPARISON OF DIFFERENT SOLUTIONS FOR MULTIMEDIA PROCESSING.

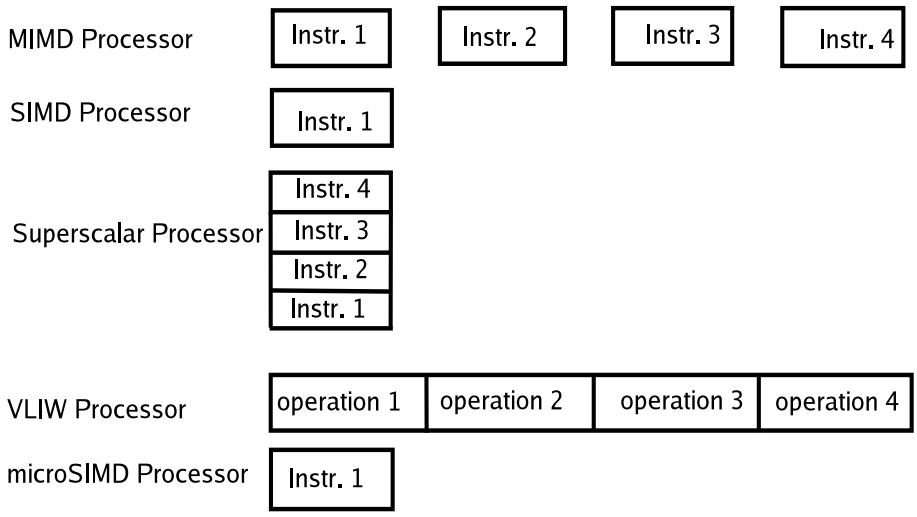

Fig. 8. Instructions needed per cycle for parallelism of degree 4.

Parallel Architecture	# of Register Files	Total Register	Width of Register	Max. Number of 16-bit Operands	Approximate Area for all of Registers
MIMD	4	128	16-bit	128	F(4*32*16)+4(d+32e)
SIMD	4	128	16-bit	128	F(4*32*16)+4(d+32e)
Superscalar	1	32	16-bit	32	F(32*16)+d+32e
VLIW	1	32	16-bit	32	F(32*16)+d+32e
MicroSIMD	1	32	64-bit	128	F(4*32*16)+d+32e

TABLE VI
STORAGE CAPACITY AND AREA REQUIREMENTS WITH FIXED NUMBER OF BITS PER REGISTER ADDRESS. (NUMBER OF REGISTERS PER REGISTER FILE IS 32 REGISTERS. "D": AN OVERHEAD PER REGISTER FILE, AND "E": AN ADDRESSING OVERHEAD PER REGISTER).

structions Per Cycle (IPCs) of 1.32 and 1.38 for VLIW and in-order superscalar, respectively.

- As can be expected, dynamic out-of-order scheduling with an average IPC of 2.17, provides much better performance than static scheduling.
- Out-of-order superscalar processors achieve 64% better performance on average than VLIW processors over all MediaBench applications [40].
- Out-of-order execution and branch prediction are important for MMAs.

Additionally, static branch prediction is typically much less accurate than dynamic branch prediction. The authors in [65] have found that SIMD versions of some benchmarks (like filtering, autocorrelation, and dot product) exhibit a speedup ranging from 1.0 to 5.5 over non-SIMD, 3-way superscalar processor that performs dynamic scheduling, while the speedup of the VLIW versions ranges from 0.63 to 9. Additionally, out-of-order execution and branch prediction techniques are extremely important to exploit the data parallelism in media applications.

B.3 Vector Processors

A vector processor is a processor that can processes entire vectors with one instruction. Vector architectures are a good candidate for multimedia processing [20], [42], because they are an effective way to exploit DLP. There are

two fundamental approaches to implement a vector processor. The first way is to replicate the functional units and achieve parallelism by processing all elements of the vector at the same time. This necessitates an interconnection network that introduces extra cost in the design and many paths will be needed from the memory to the processor. Such architectures are called a parallel vector processors. The second method approach is based on having one or relatively few pipelined functional units, that process vector elements in a pipelined fashion. These processors are called pipelined vector processors. This approach has been more widely adopted than parallel vector processors.

In [41] it has been shown that a vector architecture is a cost-effective solution for applications with high DLP. Based on the high DLP of multimedia applications, a vector architecture is a more cost-effective architecture than a superscalar architecture. The Intelligent RAM (IRAM) project [53] investigates having the vector processors in the memory. This architecture exploits the high memory bandwidth of RAM and incorporates a vector processor that can exploit the high bandwidth.

The Vector IRAM (VIRAM) [36] is a vector architecture that supports narrow data types. The vector elements can be 16, 32, or 64 bits wide. A control register specifies the element size. Based on [36], a cache-less VIRAM is 2 times faster than a 4-way superscalar processors running at a 5 times higher clock frequency and consumes 10 times more power. The VIRAM architecture, despite issuing only a single instruction per cycle, is also 10 times faster than 5- to 8-way VLIW architecture.

Vector processors provide performance benefits for both partially vectorizable programs with short vectors and highly vectorizable benchmarks. Furthermore, the VIRAM microarchitecture has significantly lower complexity than superscalar processors. Because both the vector coprocessor and the main memory system are modular, the control logic is simple, and there is no need for caches or circuit design for high clock frequency. On the other hand, superscalar processors include complicated control logic for out-of-order execution, which is difficult to design at high clock rates. VLIW processors are also more complicated than single-issue vector processors. Additionally, VLIW architectures shift significant complexity to the compiler. The scalability of the VIRAM architecture is better than that of other architectures like superscalar or VLIW. For example, in the VIRAM compared to the single-lane case, two, four and eight lanes lead to approximately 1.7x, 2.5x and 3.5x performance improvement, respectively.

IV. Programmable Multimedia Processors (PMP)

Media processors are currently only using subword parallelism, ILP, and specialized instructions and coprocessors (DCT, VLC, motion estimation) for providing high performance. Two other approaches to improve performance include higher frequency and improved ILP. ILP typically requires more resources to achieve performance gains. Increasing frequency and adding dynamic scheduling hardware both require increased power, area and processor design time [24]. Additionally, increasing processor frequency leads to deeper pipelines, longer instruction latencies, and increased memory latency, all of which reduce IPC. However, current solutions for media processing do not provide all the features necessary to obtain a high computational, low-cost, low-power, real-time response, and programmability for supporting the needs of MMAs, specially, mobile media applications. The drawbacks of existing multimedia processing approaches are summarized below:

- The function-oriented approach offers the advantages of high speed and low power, but their design and debugging phases involve a significant amount of time. Additionally, they are suitable only for specific functions, and future extensions are not possible without a redesign of hardware. Therefore, they lack the flexibility to accommodate to algorithm modifications.
- Programmable architectures offer flexibility in implementation, but their power dissipation is high, often too high for Mobile applications.
- Only a few special-purpose programmable processors can be programmed in a High-Level Language (HLL), while most of them offer firmware programming only.
- Most architectures for processing MMAs do not offer the facility to exploit TLP, which is essential to support to flexibility at higher levels of the application as in the case of MPEG-4.
- Adapted programmable architectures provide dedicated modules for several multimedia tasks, but they are not suitable for multi-standard and multi-format media applications.
- GPPs equipped with multimedia extensions are suitable for GPAs, but incur overhead instructions and cannot exploit all the DLP in MMAs.

Additionally, the complexity and variety of techniques and tools and the high computation, storage, and I/O bandwidths, multi-format and multi-standard associated with multimedia processing pose challenges, particularly from the points of scalability, high flexibility, resource utilization, dynamic adaptation capabilities, and real-time imple-

mentation. For example, MPEG-4 and MPEG-7 require a great deal of video processing using advanced algorithms. To permit these methods, multimedia is moving away from simple channel and frame-based representations towards an object-based representation of multimedia. These objects describe real-world objects, so each with its own audio, visual, and graphical characteristics specifying its spatial and temporal behavior. The advantages of object-based representation includes more flexibility for object manipulation and an increased ease of user interaction [7], [6], [28]. However, based on [33], for the next generation of multimedia, parallel media processors will prove the best alternative as they can offer both the performance and flexibility through specialized high-speed, highly parallel architectures that are programmable using a HLL. Because a suitable media processor for processing full MMAs eliminate the need for many separate processors within multimedia systems. Additionally, the programmable media processors provide several important benefits over discrete fixed function multimedia solutions. First, programmable media processors have lower cost, because using a single programmable device, the discrete fixed-devices can be replaced [10]. Second, programmable media processors have bigger manufacturing flexibility, because a single hardware platform can be used for different markets by simply changing software. Third, media processors can inherit many features from DSP architectures in their data path architecture. For example, the direct connection of functional units, which is used in the multiply-accumulate architectures of DSPs. The fourth advantage is a large register file that is useful for storing intermediate data in MMAs such as video compression/decompression. Furthermore, media processors can use non-cache memories to store programs and data in the same method as internal memories are used in DSPs [9], [30].

In the following, we provide a list of research questions that need to be investigated in order to design a cost-effective and flexible media processing systems.

- An investigation of the trade-offs between processing power and processor-memory bandwidth for a restricted area and low-power implementation based on the requirements of MMAs.
- Detailed analysis of the computational complexity, variety of techniques, formats and standards associated with multimedia processing.
- Evaluation of real-time constraints for supporting next generations of MMAs.
- Investigation of different types of parallelism such as DLP, ILP, and TLP.
- Understanding of the inter-processor communication patterns for system on chip (SOC).
- To consider the best memory organization based on temporal and spatial behavior of MMAs.
- More on-chip memory for supporting streaming data.
- Add more functional units based on requirements of MMAs.
- Provide greater flexibility using a HLL programmability for accelerating several multimedia functions simultaneously.
- Investigation of data representation, data type, datapath width, and memory hierarchy for supporting MMAs.
- Selection the best architecture for supporting future MMAs (MIMD, SIMD, microSIMD, superscalar, and VLIW architectures).

Recently, many new architectures have specially been proposed for processing MMAs. In [8] has been proposed an ISA extension called Complex Streamed Instructions (CSI) for increasing parallelism by processing of two-dimensional data streams. The CSI has several advantages. First, CSI does not put an architectural limitation on the number of subwords that are processed in parallel, because CSI processes data streams of arbitrary length. Thus, the number of bits or data elements that will be processed in parallel is not visible to the programmer. Second, CSI minimizes the overhead caused by data alignment by performing alignment in a hardware. The CSI does not also need any loop control instructions, because CSI processes streams of arbitrary length.

Matrix registers with accumulators are introduced in the Matrix Oriented Multimedia (MOM) ISA [11], [12]. The MOM architecture investigates combining traditional pipelined vector processing with subword processing. The MOM architecture relies on having a vector register file where every element contains subwords that are processed in parallel. The addressing mode is extended to stride-n access, where every element is loaded separated by an n-byte gap. Two key features distinguish MOM from CSI. First, MOM is a register-to-register architecture that use sectioning when the data do not fit into the MOM registers. The second, MOM requires overhead instructions for data conversion.

Another related architecture for processing MMAs is the Imagine processor [34], [51], which has a load/store architecture for one-dimensional streams of data records. Imagine is a stand-alone multimedia coprocessor. The focus of the Imagine project is to develop a programmable architecture that achieves the performance of special purpose hardware on graphics and image/signal processing. This is accomplished by exploiting stream-based computation at the application, compiler, and architectural level. The Imagine stream architecture is a novel architecture that executes stream-based programs. It provides high

performance with 48 floating-point arithmetic units and a area and power-efficient register organization. A streaming memory system loads and stores streams from memory. A stream register file provides a large amount of on-chip intermediate storage for streams. Eight VLIW arithmetic clusters perform SIMD operations on streams during kernel execution. Kernel execution is sequenced by a micro-controller.

A. The need for reconfigurable media processing

A designer who wants to implement an application has to decide between performance or generality. On the one hand, there are hardware realizations that are optimized to specific problems and exploit parallel and spatial execution of operations or functions such as ASICs. On the other hand, there are flexible software solutions, which are slow, but suitable to solve variety of applications such as DSPs, and GPPs [57]. A relatively new development in integrated circuits, Field-Programmable Gate Arrays (FPGAs) with static RAM programming, offer a third option. They allow more customization to either different MMAs or an adaptation to specific functions during run-time. Additionally, reconfigurable solutions provide low cost, low power, low chip area and high flexibility [14]. In traditional programmable multimedia processors, flexible software is executed on a fixed hardware architecture. In reconfigurable computing, the hardware is flexible as well. Hence, the main advantage of reconfigurable architectures is the combination of almost software flexibility with high performance. This is particularly useful for object-based video coding using MPEG-4. MPEG-4 employs a variety of algorithms and coding modes requires a more generic approach in hardware than the dedicated approach.

V. CONCLUSIONS

Recently, multimedia processing is the technology for a wide variety of applications. Multimedia processing poses very high demands on devices for transmission, storage, and computation. The growing number of international multimedia standards such as MPEG-1, 2, 4, and 7 presents challenges for both hardware and software that should perform complex multimedia processing in real-time, because every application in a multimedia environment requires different algorithms, processing techniques, and hardware. MMAs have some features such as real-time response, intensive computation for highly regular operations, and organized by small loops. The efficient processing of MMAs is currently one of the main bottlenecks in the field of media processing. Many architectures have been proposed for processing MMAs such as function-oriented approaches, adapted programmable architectures, VLIWs, GPPs enhanced with a multimedia extension, vector architectures, SIMD architectures. The function-oriented approach offers the advantages of high speed and low power, but they are suitable only for a specific application or applications , and future extensions are not possible without a redesign of the hardware. Programmable architectures offer more flexibility, but their power dissipation is usually high. Adapted programmable architectures provide dedicated modules for several multimedia tasks, but they are not suitable for multi-standard and multi-format of media applications. GPPs equipped with multimedia extensions are suitable for GPAs, and have overhead instructions and cannot exploit all DLP present in MMAs.

Superscalar processors with dynamic out-of-order scheduling provide higher performance than VLIW and superscalar processors with in-order scheduling. Vector processors provide performance benefits for both partially vectorizable programs with short vectors and highly vectorizable benchmarks. For example, the VIRAM architecture is 2 times faster than a 4-way superscalar processors running at a 5 times higher clock frequency and consumes 10 times more power. The VIRAM architecture, despite issuing only a single instruction per cycle, is also 10 times faster than a 5- to 8-way VLIW architecture. Moreover, the VIRAM microarchitecture has significantly lower complexity than superscalar processors.

The complexity and variety of techniques and tools and the high computation, storage, and multi-formats and multi-standards associated with multimedia processing pose challenges, particularly from the points of scalability, high flexibility, high performance, resource utilization, dynamic adaptation capabilities, and real-time implementation. However, parallel media processors are expected to be best candidate for processing the next generation of multimedia applications, because they can provide both the performance and flexibility through specialized high-speed, highly parallel architectures that are programmable using a HLL.

REFERENCES

[1] *"3DNow Technology Manual"*, 2000.

[2] N. Aron, H. Wijaya, A. Singh, and V. Malhotra. "Study of Multimedia Application Characteristics". http://www.stanford.edu/class/ee392c/ handouts/apps/media-long.pdf, 2003.

[3] R. Asokan and S. Nazareth. "Processor Architectures for Multimedia". In *Proc. of the 14th Annual Workshop on Architecture and System Design*, pages 589–594, November 2001.

[4] P. Bannon and Y. Saito. "The Alpha 21164PC Microprocessor". In *IEEE Proc. Compcon 97*, pages 20–27, February 1997.

[5] C. Brooks. "Developing High-Performance Multimedia Ap-

plications". www.cs.unc.edu/Admin/Exams /IP/Samples/brooks-ip.pdf, 2003.

[6] M. Burlacu and S. Kangas. "MPEG-4 Technology Strategy Analysis". Technical Report 2.4.2003, Telecommunications Software and Multimedia Lab. Helsinki University of Technology, 2003.

[7] J. Chen, U. Koc, and K. J. Ray Liu. *"Design of Digital Video Coding Systems A Complete Compressed Domain Approach"*. Marcel Dekker Inc., 2002.

[8] D. Cheresiz, B. Juurlink, S. Vassiliadis, and H.A.G. Wijshoff. "The CSI Multimedia Architecture". *IEEE Trans. on VLSI Systems*, 2004. To appear.

[9] Inc Chromatic Research. "Media Processors Software-Driven Multimedia". http://www.vxm.com/21R.84.html.

[10] T. M. Conte, P. k. Dubey, M. D. Jennings, R. B. Lee, A. Peleg, S. Rathnam, M. Schlansker, P. Song, and A. Wolfe. "Challenges to Combining General-Purpose and Multimedia Processors". In *IEEE Computer*, pages 33–37, December 1997.

[11] J. Corbal, R. Espasa, and M. Valero. "MOM: a Matrix SIMD Instruction Set Architecture for Multimedia Applications". In *Proc. of the IEEE/ACM SC99 Conf. on Supercomputing*, pages 1–10, November 1999.

[12] J. Corbal, M. Valero, and R. Espasa. "Exploiting a New Level of DLP in Multimedia Applications". In *Proc. Int. Sympo. on Microarchitecture*, 1999.

[13] A. Dasu and S. Panchanathan. "A Survey of Media Processing Approaches". *IEEE Trans. on Circuits and Systems for Video Technology*, 12(8):633–644, August 2002.

[14] A. Dasu and S. Panchanathan. "Reconfigurable Media Processing". *Parallel Computing*, 28(7):1111–1139, August 2002.

[15] K. Diefendorff and P. K. Dubey. "How Multimedia Workloads Will Change Processor Design". In *IEEE Computer*, pages 43–45, September 1997.

[16] K. Diefendorff, P. K. Dubey, R. Hochsprung, and H. Scales. "AltiVec Extension to PowerPC Accelerates Media Processing". *IEEE Micro*, pages 85–95, March-April 2000.

[17] J. T. J. Van Eijndhoven, F. W. Sijstermans, K.A. Vissers, E.J.D. Pol, and M.J.A. Tromp. "Trimedia CPU64 Architecture". In *Proc. ICCD Inter. Conf. on Computer Design*, pages 1–7, October 1999.

[18] A. H. M. R EL-Mahdy. *"A Vector Architecture for Multimedia Java Applications"*. PhD thesis, University of Manchester, 2001.

[19] R. Espasa and M. Valero. "Exploiting Instruction- and Data-Level Parallelism". *IEEE Micro*, pages 20–27, September-October 1997.

[20] R. Espasa, M. Valero, and J. E. Smith. "Vector Architectures: Past, Present and Future". In *Proc. of the Int. Conf. on Supercomputing ACM*, pages 425–432, July 1998.

[21] P. Faraboschi, G. Desoli, and J. A. Fisher. "The Latest Word in Digital and Media Processing". *IEEE Signal Processing Magazine*, pages 59–85, March 1998.

[22] F. Ferrand. "Optimization and Code Parallelization for Processors with Multimeida SIMD Instructions". Master's thesis, ENST Bretagne, 2003.

[23] J. Fritts. *"Architecture and Compiler Design Issues in Programmable Media Processors"*. PhD thesis, University of Princeton, 2000.

[24] J. Fritts and W. Wolf. "Evaluation of Static and Dynamic Scheduling for Media Processors". In *Proc. 2nd Workshop on Media Processors and DSPs*, pages 34–43, December 2000.

[25] J. Fritts and W. Wolf. "Instruction Fetch Characteristics of Media Processing". In *Proc. of SPIE Photonics West Media Processing*, 2002.

[26] J. Fritts, W. Wolf, and B. Liu. "Understanding Multimedia Application Characteristics for Designing Programmable Media Processors". In *Proc. SPIE Photonics West Media Processors*, pages 2–13, Jan 1999.

[27] B. Furht. "Processor Architectures for Multimedia: A Survey (Invited paper)". In *Proc. of Multimedia Modeling Conf.*, pages 89–109, November 1997.

[28] M. Ghanbari. *"Standard Codecs: Image Compression to Advanced Video Coding"*. The Institution of Electrical Engineers London, 2003.

[29] K. Guttag, R. J. Gove, and J. R. Van Aken. "A Single-Chip Multiprocessor For Multimedia: The MVP ". *IEEE Computer Graphics and Applications*, pages 53–64, November 1992.

[30] C. Hansen. "Architecture of a Broadband Mediaprocessor". In *COMPCON96*, Februry 1996.

[31] H. Y. Hen. *"Programmable Digital Signal Processors: Architecture Programming and Applications"*. New York Dekker, 2002.

[32] M. D. Jennings and T. M. Conte. "Subword Extensions for Video Processing on Mobile Systems". *IEEE Concurrency*, pages 13–16, July-September 1998.

[33] K. Karadayi, V. Markandey, J. Golston, R. J. Gove, and Y. Kim. "Strategies for Mapping Algorithms to MediaProcessors for High Performance ". *IEEE Micro*, pages 58–70, July - August 2003.

[34] B. Khailany, W. J. Dally, S. Rixner, U. J. Kapasi, P. Mattson, J. Namkoong, J. D. Owens, B. Towles, and A. Chang. "Imagine: Media Processing with Streams". In *IEEE Micro*, pages 35–46, Mar-April 2001.

[35] K. Konstantinides. "VLIW Architectures for Media Processing". *IEEE Signal Proc. Magazine*, pages 16–19, March 1998.

[36] C. Kozyrakis and D. Patterson. "Vector Vs. Superscalar and VLIW Architectures for Embedded MUltimedia Benchmarks". In *In the Proc. of the 35th int. Symposium on Microarchitecture Instabul Turkey*, November 2002.

[37] Christoforos E. Kozyrakis and David Patterson. "A New Direction for Computer Architecture Research". In *IEEE Computer*, pages 24–32, November 1998.

[38] I. Kuroda and T. Nishitani. "Multimedia Processors". *Proc. of the IEEE*, 86(6):1203–1221, June 1998.

[39] V. Lappalainen, T. D. Hamalainen, and P. Liuha. "Overview of Research Efforts on Media ISA Extensions and Their Usage in Video Coding". *IEEE Trans. on Circuits and Systems for Video Technology*, 12(8):660–670, August 2002.

[40] C. Lee, M. Potkonjak, and W. H. Mangione-Smith. "MediaBench: A Tool for Evaluating and Synthesizing Multimedia and Communicatons Systems". *IEEE Micro*, pages 330–335, June 1997.

[41] C. G. Lee and D. J. DeVries. "Initial Results on the Performance and Cost of Vector Microprocessors". In *Proc. Thirtieth Annual IEEE/ACM Inter. Sympo. on Microarchitecture*, pages 171–182, December 1997.

[42] C. G. Lee and M. G. Stoodley. "Simple Vector Microprocssors for Multimedia Applications". In *Proc. of the 31st annual ACM/IEEE inter. sympo. on Microarchitecture*, pages 25–36, 1998.

[43] R. B. Lee. "Accelerating Multimedia With Enhanced Microprocessors". In *IEEE Micro*, pages 22–32, April 1995.

[44] R. B. Lee. "Subword Parallelism with MAX-2". *IEEE Micro*, pages 51–59, August 1996.

[45] R. B. Lee. "Multimedia Extensions for General-Purpose Processors". In *Proc. of the IEEE Signal Processing Systems Design and Implementation*, pages 9–23, November 1997.

[46] R. B. Lee. "Efficiency of MicroSIMD Architectures and Index-Mapped Data for Media Processors". In *Proc. Media Processors IS&T/SPIE Symposium on Electric Imaging*, pages 34–46, January 1999.

[47] R. B. Lee and M. D. Smith. "Media Processing: A New Design Target". In *IEEE Micro*, pages 6–9, August 1996.

[48] L. A. Lev, B. A. Frederick, D. L. Wendell, H. K. Hingarh, M. Allen, R. J. Melanson, and V. Reddy. "A 64-b Microprocessor with Multimedia Support ". *IEEE Journal of SOLID-State Circuits*, 30(11):1227–1238, November 1995.

[49] Inc. MIPS Technologies. "MIPS Extension for Digital Media with 3D". www.mips.com.

[50] J. Oliver, V. Akella, and F. Chong. "Effi cient Orchestration of Sub-Word Parallelism in Media Processors". In *Proc. Sympo. on Parallel Algorithms and Architectures*, 2004.

[51] J. D. Owens, S. Rixner, U. Kapasi, P. Mattson, and B. Towles. "Media Processing Applications on the Imagine Stream Processor". In *Proc. IEEE Intr. Conf. on Computer Design: VLSI in Computers and Processors (ICCD'02)*, September 2002.

[52] S. Panchanathan. "Architectural Approaches for Multimedia Processing". In *Proc. of the 4th Inter. ACPC Conf. on Parallel Numerics and Parallel Computing in Image Processing Video Processing and Multimedia*, pages 196–210, 1999.

[53] D. Patterson, T. Anderson, N. Cardwell, R. Fromm, K. Keeton, C. Kozyrakis, R. Thomas, and K. Yelick. "A Case for Intelligent RAM". *IEEE Micro*, pages 34–44, March-April 1997.

[54] A. Peleg, , and U. Weiser. "MMX Technology Extension to the Intel Architecture". *IEEE Micro*, pages 42–50, August 1996.

[55] A. Peleg, S. Wiljie, and U. Weiser. "Intel MMX for Multimedia PCs". *Communications of the ACM*, pages 25–38, January 1997.

[56] Philips. "TriMedia TM-1000: Programmable Media Processor". http://www.semiconductors.philips .com/pip/75003407.html.

[57] P. Pirdch, C. Reuter, J. P. Wittenburg, M. B. Kulaczewski, and H. J. Stolberg. "Architecture Concepts for Multimedia Signal Processing". *Journal of VLSI Signal Processing*, 29:157–165, August 2001.

[58] P. Pirsch, A. Freimann C. Klar, and J. P. Wittenburg. "Processor Architectures for Multimedia Applications". In *Embedded Processor Design Challenges: Systems Architectures, Modeling and Simulation*, pages 188–206, February 2002.

[59] S. K. Raman, V. Pentkovski, and J. Keshava. "Implementing Streaming SIMD Extensions on the Pentium 3 Processor". *IEEE Micro*, pages 47–57, July-August 2000.

[60] P. Ranganathan, S. Adve, and N. P. Jouppi. "Performance of Image and Video Processing with General Purpose Processors and Media ISA Extensions". In *Proc. Inter. Sympo. on Computer Architecture (ISCA 26)*, pages 124–135, 1999.

[61] H. Sasaki. "Multimedia Complex on a Chip". In *IEEE Inter. Solid-State Circuits Conf.*, pages 16–19, 1996.

[62] K. Scott and J. Davidson. "Exploring the Limits of Sub-word Level Parallelism". In *Proc. of the 2000 Inter. Conf. on Parallel Architectures and Compilation Techniques*, October 2000.

[63] N. T. Slingerland and A. J. Smith. "Multimedia Instruction Sets for General Purpose Microprocessors: A Survey". Technical Report UCB//CSD-00-1124, University of California, December 2000.

[64] D. Talla. *"Architectural Techniques to Accelerate Multimedia Applications on General Purpose Processors"*. PhD thesis, University of Texas at Austin, 2001.

[65] D. Talla, L. K. John, V. Lapinskii, and B. L. Evans. "Evaluating Signal Processing and Multimedia Applications on SIMD VLIW and Superscalar Architectures". In *Proc. IEEE Inter. Conf. Computer Design: VLSI in Computers and Processors*, page 163, September 17 - 20 2000.

[66] D. Talla, L. Kurian John, and D. Burger. "Bottlenecks in Multimedia Processing with SIMD Style Extensions and Architectural Enhancements". *IEEE Trans. on Computers*, 52(8):1015–1031, August 2003.

[67] S. Thakkar and T. Huff. "The Internet Streaming SIMD Extensions". *Intel Technology Journal*, pages 1–8, 1999.

[68] M. Tremblay, J. Michael 0'Connor, V. Narayanan, and L. He. "VIS Speeds New Media Processing". *IEEE Micro*, pages 10–20, August 1996.