

# MT mediaLib™ for Chip MultiThreaded (CMT) Processors

---

**Liang He and Harlan McGhan**  
**Scalable Systems Group**  
**Sun Microsystems Inc.**

---

## 1.0 Background

---

Innovation in processor design tends to happen in waves. The introduction of the first 4-bit/8-bit microprocessor designs (the 4004/8008) by Intel in 1971/1972 triggered a wave of competing designs from other semiconductor companies, including Motorola, Zilog, MOS Technology, TI, Rockwell, RCA, Fairchild, and others. The SPARC® architecture was part of a wave of RISC processor designs that appeared between 1985-92, and included such famous (and not-so-famous) lines as MIPS, PA-RISC, Clipper, ROMP, 29000, 88000, i860, Alpha, and POWER. Even Intel's massive and prolonged Itanium project was just one aspect of a wave of VLIW-style designs that crested in the 1990s, also encompassed Transmeta's x86-compatible low power Crusoe line, and succeeded in deluging the embedded space with a plethora of colorfully named media and signal processors like TriMedia, Mpact, Carmel, ManArray, MAP, StarCore, TigerSharc, Jazz, and Crescendo.

Interspersed between these waves of architectural innovation have been other, vitally important waves of design innovation, that tend to cut across architectural lines. One such wave is occurring now, in the rapidly acceler-

ating industry-wide shift toward new multicore/multithread processor implementations. Another such wave occurred in the mid-90s, when a widespread movement to adopt instruction set extensions aimed at accelerating multimedia and related operations swept through the industry. Just as Sun Microsystem's SPARC processor architecture played a central role in the "RISC revolution" of the 1980s, and SPARC processors today are leading the way to new and still-controversial radical Chip MultiThreading (CMT) designs, SPARC processors in the 1990s were at the forefront of the technology wave of multimedia instruction set extensions that swept across the industry, permanently altering the computing landscape.

The central idea behind the 1990s movement to improve the ability of general-purpose processors to handle multimedia and related applications by extending their instruction sets was the opportunity to adapt new "wide" registers and execution units<sup>1</sup> to handle groups of shorter, fixed-width data values in parallel. Thus, a 64-bit processing unit might be modified to handle either two 32-bit values, four 16-bit values, or eight 8-bit values at the same time. Since exactly the same operation (addition, subtraction, etc.) was performed on each of the parallelized short values, these extensions generally went under the rubric of "SIMD" (Single Instruction, Multiple Data) instructions. And since the short fixed-width data values targeted by these instructions were used most prominently in the encoding formats for graphics, sounds, and images, they also commonly were referred to as "multimedia" instruction set extensions.<sup>2</sup>

Since the overhead involved in retrofitting a wide execution unit to handle SIMD instructions, too, is relatively small (typically, 1-3% added area with no significant impact on cycle time), while the payoff can be huge – usually doubling the speed of affected algorithms, and sometimes multiplying it by a factor of four, eight, or even more – it is hardly surprising that the idea of adding SIMD instruction set extensions to established general-purpose

- 
1. These became standard in processor designs with the multimillion transistor budgets that were available by the early 1990s.
  2. For example, the pixels used to form color images often are encoded as bytes. In the simplest 256-color schemes, each pixel can be rendered as a single byte; while in more complex "TrueColor" schemes, the pixels often are composed of multiple bytes. Thus, a 24-bit scheme might use one byte each to specify a red, green, and blue component value for a pixel; while a 32-bit scheme might add a fourth, "alpha" byte that provides information about a pixel's transparency. Similarly, sound is typically encoded in formats that range from 1 to 4 bytes.

architectures was quickly embraced. Or rather, it was quickly emulated once the practical value of SIMD extensions had been demonstrated.

The pioneering architecture here was Sun Microsystem's family of 64-bit UltraSPARC<sup>®</sup> processors, which first began shipping in 1995.<sup>1</sup> Starting with the original UltraSPARC I processor, every member of the UltraSPARC processor family has been equipped with a set of acceleration extensions, termed VIS<sup>™</sup> (originally, an acronym for Visual Instruction Set), as an integral part of its design. While UltraSPARC was not the first general-purpose processor line ever to incorporate any SIMD-style instructions,<sup>2</sup> it was the first to provide a complete, general set of SIMD instructions across an entire processor family. Thus, it was the first processor to elevate SIMD instructions from a specialized hardware acceleration feature, targeting a particular application (or narrow range of applications), to a general software-visible acceleration feature, inviting use by a broad range of ISVs and other software developers looking to achieve competitive advantage through better performance.

The leading role of UltraSPARC processors in the developing wave of multimedia acceleration technology based on SIMD instruction set extensions was quickly acknowledged. Indeed, the first recognition of Sun's leader-

- 
1. The rollout of the 64-bit UltraSPARC processor family actually occurred over three years. In early 1993, the 64-bit SPARC V9 architecture was finalized [1] and Sun's plans to introduce the UltraSPARC family built on this foundation revealed [2]. In the Fall of 1994, the initial UltraSPARC I implementation was previewed at the Microprocessor Forum [3] [4]. In the Spring of 1995, UltraSPARC I was announced [5], with processors shipping in volume before the end of the year [6] [7].
  2. Specifically, there were three notable RISC precursors to VIS. The Intel i860 series (introduced 1989) included 6 SIMD instructions. The second generation of the Motorola 88000 series (the 88110, introduced 1991) included 9 SIMD instructions. The PA-RISC 7100LC processor (introduced 1994) included 5 SIMD instructions. Since VIS consisted of a total of about 80 instructions, the contrast in simple numbers between VIS and any of these earlier SIMD instruction sets is striking. Corresponding to the very limited number of SIMD instructions included in these earlier processors were equally limited ambitions for these extensions. The PA-7100LC, for example, included its 5 SIMD instructions specifically to speed up MPEG-1 decoding, so HP could cost-reduce its entry-level workstations (by eliminating the expense of a graphics coprocessor). From this perspective, adding SIMD instructions to the 7100LC was completely successful. But, from the perspective of inducing software developers to begin employing SIMD generally to accelerate PA-RISC code, they were not so much a complete failure as an absolute non-starter. Indeed, even the SIMD-enhanced MPEG decoding algorithms developed specifically for the 7100LC were not useful on any other PA-RISC processor, since no other PA-RISC processor included these instructions. To be sure, HP did add an enhanced set of 10 SIMD instructions to its PA-RISC line generally, at the PA-8000 generation in 1996, but that broadening of ambitions came after UltraSPARC processors had shown the way, not before.

ship in this area came even before the new UltraSPARC family began to ship, or any swell of following designs was yet visible:

Although a few others have dabbled in this area, Sun's UltraSparc processor offers the most extensive set of features aimed at improving performance on multimedia data types. With just a small amount of special hardware, UltraSparc increases the speed of some calculations by eight times or more .... These tasks are well beyond the capabilities of current microprocessors .... In fact, it is doubtful that any other next-generation microprocessor will achieve these feats, putting UltraSparc in a class by itself.... This new hardware will give the forthcoming Sun chip an advantage when competing against other next-generation processors, an advantage that is not apparent from basic SPEC [standard performance] measurements. As multimedia applications become more prevalent, other manufacturers are likely to include similar features in future designs. [4]

Intel was the first to fulfill the expectation expressed here, that other microprocessor vendors soon would be forced to respond in kind to the challenge posed by the exceptional new multimedia capabilities of the UltraSPARC processor family. In 1995, they announced plans to add a set of 57 SIMD instructions, dubbed "MMX",<sup>1</sup> to Pentium. Indeed, so anxious was Intel to bring SIMD technology to market, they introduced a significant rework of the original Pentium design, Pentium MMX, just four months before introducing Pentium II (which superseded the Pentium I core design and also incorporated the new MMX technology).<sup>2</sup> However, Intel's quick endorsement of SIMD instructions served only to enhance UltraSPARC's premier place in the new world of SIMD instruction set extensions.

1. Although it is widely supposed that "MMX" is an acronym for "MultiMedia eXtensions," Intel has never acknowledged this derivation of the name – probably to protect their MMX trademark against accusations that it is nothing more than a generic term (that anyone can use). However, they never have volunteered any other origin for it, either.
2. The increasing complexity and sophistication of Pentium processors was reflected in the growing transistor counts of these designs: from 3.3M in the original Pentium I (introduced 3/22/1993), to 4.5M in Pentium MMX (introduced 1/8/1997), to 7.5M in Pentium II (introduced 5/7/1997). The timing of Pentium MMX is puzzling, given Pentium II was so close to production at the time it was introduced. Certainly, Pentium MMX has the dubious distinction of being the shortest-lived of any major new release of an Intel processor. One plausible explanation is that, because MMX technology involved instruction set extensions to Intel's line of x86 architecture processors (the first additions to the x86 instruction set since the architecture was upgraded to 32-bits, back in 1985 with the 80386 generation), Intel believed that it would be awkward if not impossible to sell older Pentium I generation processors alongside new Pentium II generation processors, unless Pentium I was first upgraded with the new accelerations for multimedia software, too. There is absolutely no doubt that Intel perceived MMX as a significant differentiator. The most visible token of their exuberance over "Pentium processors with MMX technology" was a major new ad campaign, memorably featuring dancing figures dressed in colorful "bunny suits" (the spacesuit-like one-piece outer garments worn by workers in the "clean rooms" where semiconductor chips are fabricated). For maximum impact, the first of these spots aired on January 26, 1997, during the Super Bowl [8].

[Intel's] MMX ... [is] outshined by VIS .... HP's recent processors ... operate on only two 16-bit quantities at once. To date, the other leading desktop RISCs - PowerPC, MIPS, and Alpha - have somehow failed to implement multimedia instructions. Microprocessor Report, 3/5/1996, p. 9

By June 1996, barely 8 months after Sun began shipping UltraSPARC-based systems, EE Times was ready to declare:

Sun Microsystems Ultrasparc CPU ... [is] a harbinger of the Next Big Thing in micro-processor architecture. Buried in the logic of the chip ... was a bold experiment – a way to accomplish single-instruction, multiple-data (SIMD) processing .... This was the Visual Instruction Set (VIS) ... [which] quickly progressed from curiosity to industry trend. The ideas in VIS were picked up and extended by Intel ... to re-emerge as MMX. They were explored by other RISC design teams .... VIS and its children are moving into the mainstream of microprocessor instruction sets with a growing inevitability. [9]

Marking the rapidly growing momentum behind this new technology, the August 1996 issue of IEEE Micro devoted its entire feature section to “Media Processing: A New Design Target” [10] with articles on Sun’s VIS [11], Intel’s MMX [12], and HP’s MAX [13]. Unable to ignore the handwriting on the wall any more, the other leading vendors of general purpose processors soon announced their own plans to add multimedia extensions. In fact, without exception, every general purpose architecture moved to implement a set of SIMD instruction set extensions in the period

1995-2000. The following table summarizes the history of SIMD instruction sets for general purpose processors.

**TABLE 1: SIMD Instruction Sets for General Purpose Processors**

PROCESSOR	ANNCD	AVLBL	NAME	SIZE	GENRL	COMMENTS
SPARC	1994	1995	VIS	80	Very	First complete set of SIMD instruction set extensions, modest upgrade to VIS 2.0 in UltraSPARC III (2000).
PA-RISC	1996	1996	MAX-2	10	Some	MAX-1 (1994) was 5 SIMD instructions added to the 32-bit PA-7100LC (only), to lower the cost of graphics in HP's entry-level workstations; MAX-2 was the first generalized extension common to all 64-bit PA-RISC processors.
Pentium I & II Pentium III Pentium IV	1995 1999 2000	1997 1999 2000	MMX SSE SSE2	57 70 144	Very	SSE (Streaming SIMD Extensions) is a superset of MMX. This renamed and augmented SIMD instruction set was the only upgrade of note in Intel's Pentium III generation (which was strictly a fill-in product, based on the older Pentium II core, while waiting for the delayed Pentium IV to arrive).
MIPS	1996	2000	MIPS V MDMX	8 19	Yes	Neither of these announced SIMD extension sets reached market as intended: MIPS V was folded into the overarching MIPS64 ISA standard (1998), MDMX (MIPS Digital Media eXtensions) was largely superseded by MIPS-3D (1999).
Alpha	1996	1997	MVI	5	No	A few Motion Video Instructions added just to boost MPEG-2 decoding; Alpha's designers argued Alpha was fast enough for multimedia as is, more SIMD instructions only would hurt its high clock rate.
AMD x86	1997	1998	3DNow!	21	Some	A response to MMX, focused largely on needs of PC game industry for fast 3D graphical rendering.
PowerPC	1998	1999	AltiVec	162	Very	AltiVec is a Motorola trademark; the same set of extensions is called VMX (Video and Multimedia eXtensions) by IBM, and The Velocity Engine by Apple. Whatever its name, this extension set was added to Motorola's PowerPC line for Apple desktops, but not to IBM's POWER line for servers.

## 2.0 Value of SIMD Instruction Set Extensions

Although not part of the 64-bit V9 SPARC standard,<sup>1</sup> adding the VIS ISA extensions to UltraSPARC was highly synergistic with the upgrade of the SPARC architecture from 32- to 64-bits. One objection to making general-purpose processors 64-bits wide was that this increase in functionality had to be purchased at the expense of efficiency, since many integer data values are fixed at shorter lengths. Manipulating an 8-bit value on a 64-bit

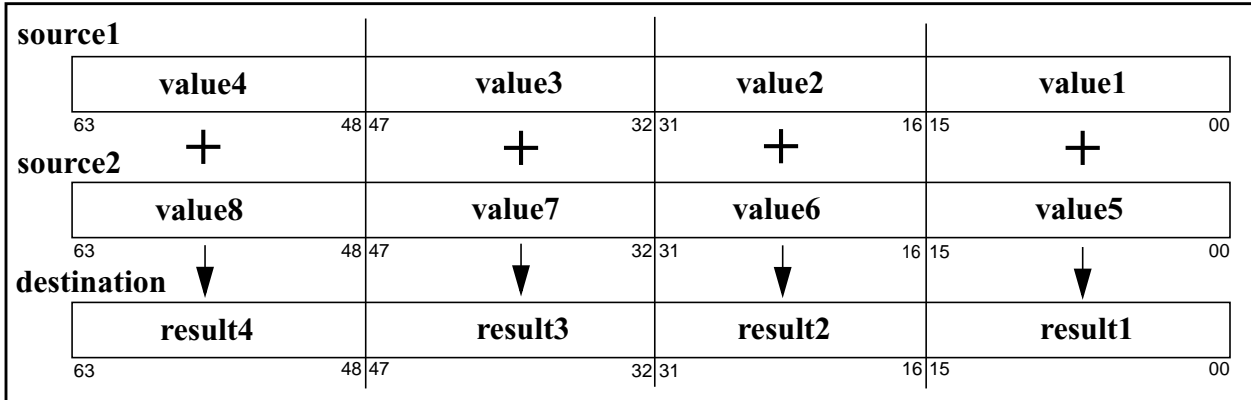
processor, for example, involves fetching, calculating, and storing numbers comprised of 8 significant bits padded out with 56 leading zeros. This need to accommodate large numbers of meaningless zeros when computing short integer values on a 64-bit processor consumes resources, time, and power – all to no gain.

VIS is a powerful remedy for this problem, since (where used) it converts the 64-bit width of UltraSPARC from a liability into an asset. At its heart, VIS provided a new set of RISC-style 3-operand “partitioned” instructions for a new type of “packed data,” composed of multiple short fixed-width data values tightly inserted into a 64-bit register. Depending on the length of the values to be manipulated, packed data can be composed of 2 32-bit, 4 16-bit, or 8 8-bit elements. For instance, the partitioned add instruction, `fpadd16`, specifies two source registers, each packed with 4 16-bit values, and a destination register to store the packed result, employing UltraSPARC’s 64-bit wide functional units to perform four 16-bit computations in parallel, during a single cycle (see Figure 1). In addition to a far more efficient use of UltraSPARC’s 64-bit wide registers and computational units (with a corresponding reduction in its overall power consump-

- 
1. V9 is short for Version 9 of the SPARC Instruction Set Architecture (ISA), the version that upgraded the SPARC standard to 64-bits. The VIS instructions leverage a set of opcodes reserved by the V9 standard for “implementation dependent” instructions. Thus, while it is true that the SPARC V9 ISA does not specify any VIS instruction, it is also true that every VIS instruction implemented by any UltraSPARC processor is specified within the overall V9 framework. Although the new VIS instruction set extensions were described in the 1994 disclosures of the UltraSPARC design [3] and discussed at some length in [4], the first full account of VIS [14] and its effectiveness at decoding MPEG video streams [15] was given at Comcon 40, held in San Francisco the week of March 5-9, 1995 – the very same conference, incidentally, at which HP was presenting details of their achievements in MPEG decompression using MAX-1 on the PA-7100LC [16].

tion), the result of executing this new type of operation is a 4X speed-up, compared to performing the same four 16-bit adds sequentially.

**FIGURE 1 Operations on Packed Data by the Partitioned Add Instruction, fpadd16**



In addition to providing a way to efficiently use wide hardware to compute with short data values, adding VIS instructions to UltraSPARC processors also improves utilization of the available functional units. While all UltraSPARC processors provide the ability to fetch and execute a total of four Instructions Per Clock cycle (4 IPC), the instructions cannot all be of the same type. Instead, they must be selected a mix that includes a maximum of two integer ALU operations, 1 load/store, 1 branch, and two floating-point operations.

Although the potential peak performance of this four-issue design is impressive – 5.4 Billion Instructions Per Second or BIPS (at 1.35 GHz) – achieved performance often will fall far short of potential performance. Ignoring any other impediments to peak performance, computational speed can be limited simply by the mix of available instructions. This is particularly true of integer programs that contain little or no floating-point math, and therefore leave the floating-point hardware (and its associated two issue slots) largely or entirely idle.<sup>1</sup> Specifically, when executing integer-only code, the best case IPC for UltraSPARC drops from 4 to 2.<sup>2</sup>

1. Although it became common practice in the 1990s to integrate Floating-Point Units (FPUs) into processor chips, making floating-point capability both cheap and ubiquitous, an often overlooked side-effect of this design shift was a noticeable reduction in the efficiency of processors, as measured by the ratio of transistors to instructions executed per clock. In fact, FPUs add a large number of transistors to a design that, in the absence of any floating-point instructions to execute (the common case in integer code), are left to sit idle.

By designing the VIS instructions to execute in the FPU – or, rather, by designing an FGU (Floating-Graphical Unit) instead of an FPU – the UltraSPARC designers helped address the inefficiency of underused floating-point hardware. Since VIS instructions execute in the FGU rather than the Integer Unit (IU), the typical mix of instructions in a VIS-enhanced algorithm frequently will peak at an IPC of 4 rather than just 2: two VIS instructions, issued into the two FGU issue slots, and either two standard integer instructions, issued into the two integer issue slots, or one integer instruction and a load (to keep the execution units supplied with data). Thus, even where SIMD cannot be applied, VIS still has the potential to double performance on integer code by opening the FGU for use, in addition to the IU (operating in SISD or Single Instruction, Single Data mode in both functional units).

Where SIMD can be applied, of course, the potential performance benefit is much greater. Consider a case where two SIMD VIS instructions are issued in a clock cycle, in addition to two standard integer instructions (or an integer instruction and a load). If each of the two VIS instructions performs (say) 4 16-bit operations in parallel, that's a total of  $4+4+1+1 = 10$  things done in parallel. 10 parallel operations is an acceleration of 2.5X, compared to the best case of 4 things done in parallel without VIS. Compared to an integer-only peak of 2 parallel instructions, it's an acceleration of 5X.

Even an acceleration of 5X is not the limit for a VIS-accelerated algorithm. Consider, for example, the highly specialized VIS instruction, PDIST, which provides a specific combination of the operations used for motion estimation in video-compression schemes like MPEG. More precisely, the PDIST instruction operates on two registers that each have been packed with eight bytes, performing the following sequence of operations in parallel: it subtracts each of the 8 bytes in one register from the corresponding byte in the other register, performs an absolute value operation on the 8 differences (to eliminate any negative numbers), adds all 8 differences together, and stores the sum in a register (see Figure 2 for a graphical representa-

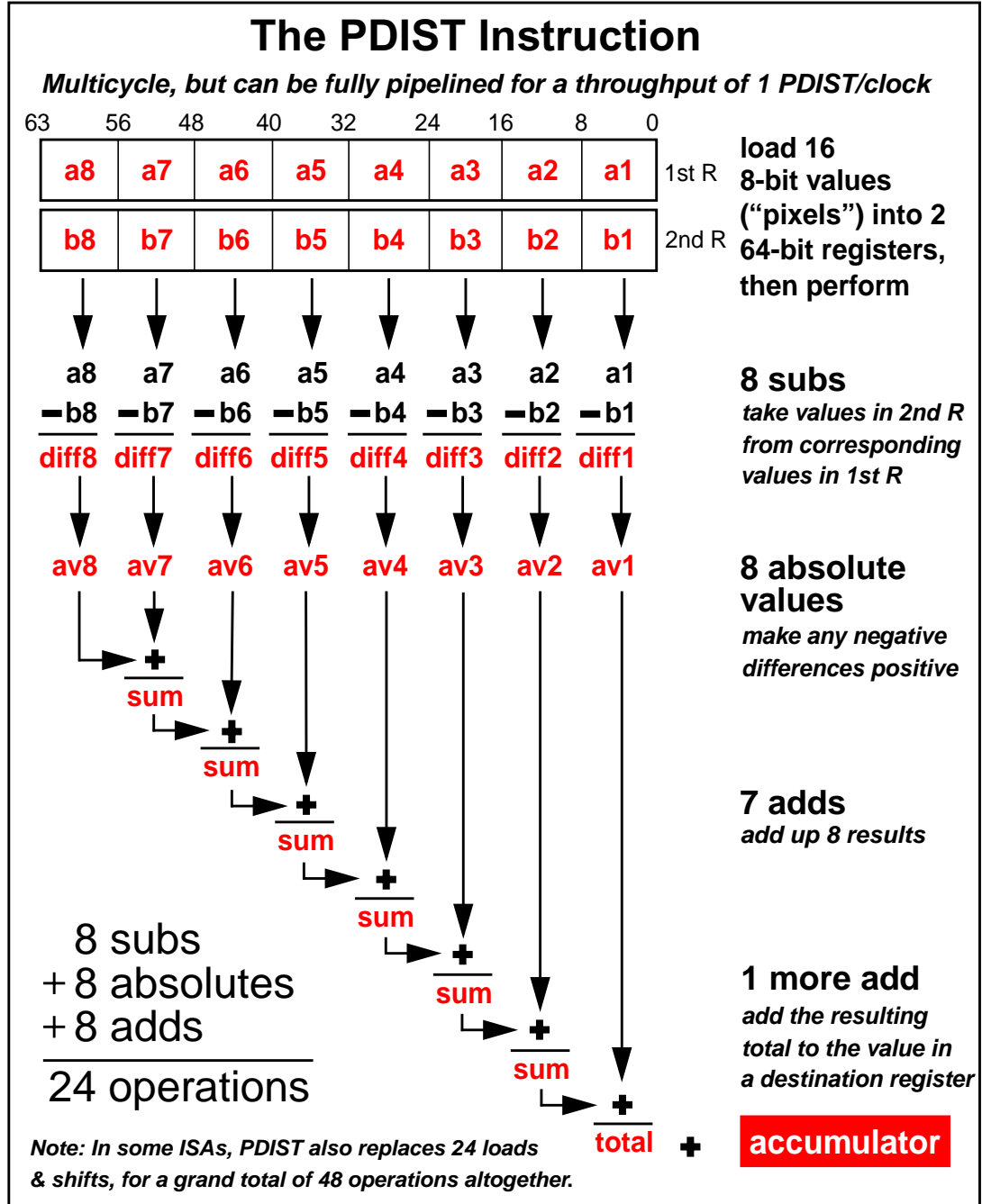
- 
2. Regrettably, where a load and/or a branch can be issued in addition to 2 integer instructions, the overall effect on IPC tends to be negative rather than positive. Even a (best case) correctly predicted-taken branch will cause a pipeline "bubble" of several clocks, while every load that misses in primary cache will stall the pipeline for as many clocks as are needed to fetch data from a more remote location (for data as far away as main memory, the stall on a load miss can stretch into hundreds of clock cycles).

tion of PDIST). Without VIS, this would be a total of 24 separate operations, namely:

- 8 subtractions
- 8 absolute values
- 8 additions

But by applying the VIS PDIST instruction, all 24 operations are performed in a single clock cycle.

FIGURE 2 The Highly Parallel VIS Instruction, PDIST



That's a phenomenal amount of processing to get done in a single clock cycle, but it's still just executing 1 VIS instruction. Now consider that UltraSPARC processors can sustain execution of 4 instructions at a time, i.e., can execute 3 more instructions in conjunction with PDIST. Further, since PDIST uses just one of the UltraSPARC processor's two FGUs, the second FGU is available for a second VIS operation, say, a PADD, performing 4 parallel 16-bit adds. Also, during the same clock cycle, one of UltraSPARC's integer units could be doing an address calculation (integer add), and the load/store unit could be executing a load (to keep up a steady stream of data to feed the VIS operations of the two FGUs).

The example mix comprises a sum of 24 PDIST + 4 PADD + 1 Integer Add + 1 Load, for a total of 30 operations, all done in a single clock cycle. In a 1350 MHz UltraSPARC processor, that's an indefinitely sustainable processing rate not merely of:

$$4 \text{ Instructions} \times 1.35 \text{ GHz} = 5.4 \text{ BIPS}$$

But of:

$$30 \text{ Operations} \times 1.35 \text{ GHz} = 40.5 \text{ BOPS}$$

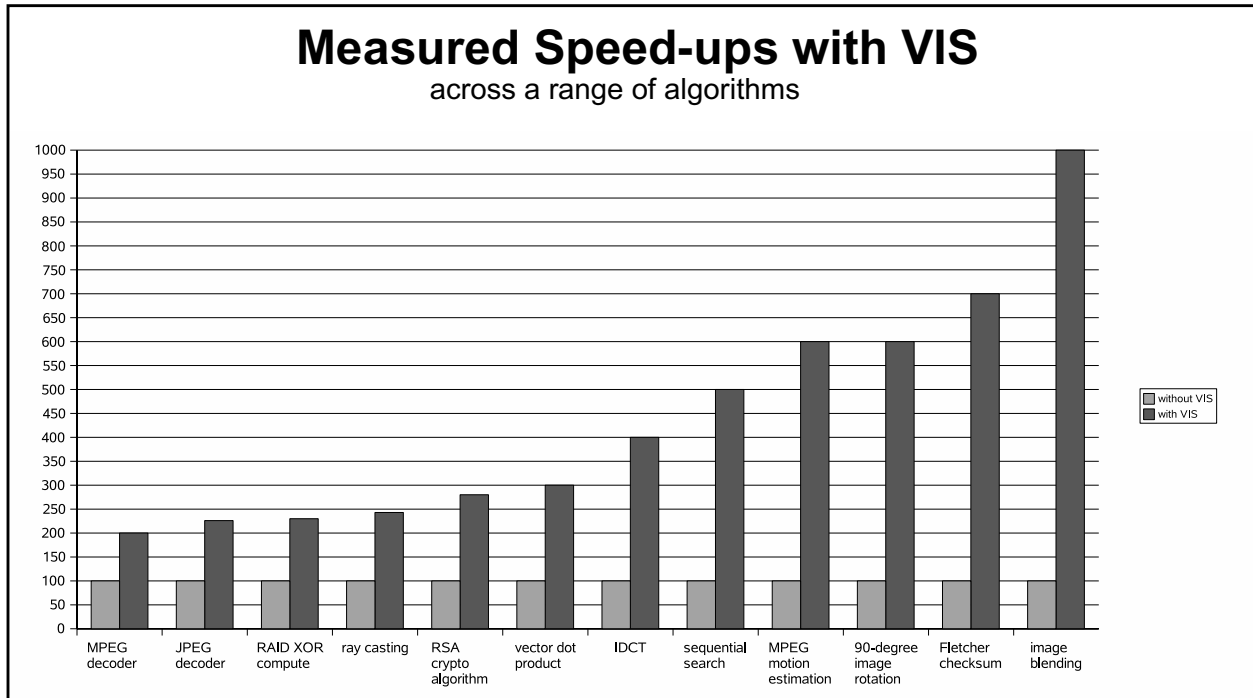
While this might seem a frantic rate of instruction execution to keep up, maintaining this sort of execution pace for an extended period is more than just an abstract possibility. In fact, the instruction mix given here is typical of decompression algorithms, like those used in MPEG-2 decoding. In a wide variety of specialized problems like this one, where VIS can be brought to bear,<sup>1</sup> the total processing power of UltraSPARC is truly staggering.

---

1. Of course, VIS cannot be used everywhere, for a variety of reasons. Some applications simply lack SIMD parallelism. In other cases, VIS may lack a specialized instruction needed to speed a particular function. In still other cases, it may be impossible to deploy VIS for implementation-dependent reasons, of the sort often encountered in legacy code (alignment requirements, edge conditions, formatting and ordering restrictions). Conversely, as mentioned above, it may be possible to exploit VIS outside the boundaries of normal SIMD parallelism, as a means (in effect) of accessing the FGUs simply as two more integer units. In fact, one of the very first commercial uses of VIS (a medical imaging application, demonstrated at RSNA 1995 in Chicago, the annual meeting of the Radiological Society of North America) applied VIS in SISD mode, to access the FGU just as two more integer units conveniently blessed with a fast multiplier.

Figure 3 shows the sorts of performance gains that have been measured for VIS-enhanced algorithms across a range of application areas.

FIGURE 3 The VIS Performance Advantage Across a Range of Applications



### 3.0 Using SIMD Instructions

The fact that SIMD instructions have proven themselves to be highly effective accelerators for a wide variety of applications does not mean that it is either easy or simple to apply them effectively. Since ease of use is perhaps the most vital factor in getting VIS widely accepted and adopted by ISVs and other application developers, facilitating the use of VIS is itself an issue of great practical importance. Corresponding to its role as a pioneer of general SIMD instruction set extensions, Sun Microsystems also has been a pioneer in developing effective, programmer-friendly interfaces to these extensions.

In some ways, the best possible solution to encouraging use of VIS wherever applicable would be to remove the burden of its use from program-

mers altogether, and place it instead on the compilers supplied for UltraSPARC processors. Then, whenever any program was compiled with a VIS-knowledgeable compiler, VIS instructions automatically would be inserted into the code sequence where and as appropriate – all without any requirement that the programmer even know about the existence of VIS, let alone have mastered any of its subtleties. Unfortunately, this alternative suffers from several practical drawbacks, including the fact that many of the opportunities to use VIS simply can't be identified automatically.<sup>1</sup> Hence, as a practical solution to the problem of facilitating the use of VIS, automatic compiler-based deployment has not proven a significant means of realizing the potential benefits of SIMD-enhanced code.

At the opposite end of the ease-of-use spectrum lies the prospect of requiring programmers to write VIS instructions directly in assembler. In many ways, this alternative is the reverse image of the compiler solution. By placing the burden of use entirely on the programmer, it maximizes the opportunities for VIS-literate code writers to use VIS where and as appropriate. At the same time, it is the least programmer-friendly way to deploy VIS. Indeed, since writing code in assembler is slow, and the code created is non-portable, assembly language programming simply is not an option for most software developers. Hence, the bottom line for the assembler alternative is very similar to the bottom line for the compiler alternative: it is not a significant means of deploying SIMD functionality and reaping its benefits.

Clearly, some middle ground is required. If the burden of using VIS cannot be entirely removed from programmers and placed instead on automated code-generation tools for the SPARC/Solaris™ platform, it is still possible

1. Although attempts have been made to automate the generation of SIMD instructions at the compiler level [17]. Even if these attempts were to become more successful in future than they have been to date, however, the fact remains that the compiler is an awkward place to generate SIMD instructions – at least, as long as these extensions remain outside the umbrella of the SPARC architectural ISA. Since (by definition) every 64-bit SPARC processor implements the SPARC V9 ISA, compilers that generate V9-compliant code are guaranteed their code will run correctly (though perhaps not quickly) on any 64-bit SPARC processor, regardless of its implementation. But since VIS is not part of V9, different SPARC processors have freely implemented distinct subsets and supersets of the VIS extensions. Hence, for the SPARC compilers to generate VIS-enhanced code, they would have to be knowledgeable about the particular set of VIS instructions implemented on each possible target processor, and support the necessary switches to enable use of the corresponding set of VIS instructions. The resulting binaries could not be guaranteed portable to new implementations of SPARC (breaking SPARC's long-standing guarantee of upward binary code compatibility); and could be made portable across known implementations only by being encumbered with branches to multiple alternative sequences of code.

to minimize the barriers programmers face when attempting to use VIS, and so maximize the use they make of VIS in their code. Specifically, for programmers that develop application code in C and other high-level languages, ways must be provided to access VIS directly from those high-level languages, rather than through assembler.

The simplest way to up-level access to VIS for use in high-level languages is by providing in-line macros: essentially, names for common sequences of VIS assembler instructions that are automatically expanded when the program is compiled for execution.<sup>1</sup> However, while it is far easier to use macros than to write out entire sequences of instructions directly in assembler, this solution still suffers from the problem that it requires programmers to be knowledgeable about when and where to invoke VIS in their code. Hence, while macros both simplify access to VIS and make its use less painful, they do not succeed at making either access to VIS completely simple or its use entirely pain free.

In practice, it turns out that the best and most helpful way to facilitate use of VIS is not (merely) to encapsulate common VIS assembler sequences as macros, but rather to incorporate VIS into library routines. This has several significant advantages.

- It lifts the burden of knowing about VIS off the shoulders of the application developer. The application developer simply calls whatever library routine is needed to perform a specific function. If the routine can be and has been enhanced with VIS, then the application benefits from whatever speedup is achieved thereby; if not, not. Either way, the application developer need know nothing about how the routine has been coded.
- Assuming the library is dynamically linked into the program, applications automatically stay current with the latest version of the library. Thus, if a formerly unenhanced routine is later enhanced with VIS (or a newer version of VIS), the application immediately benefits from the upgrade.
- It makes VIS equally accessible from any language that links into the library. It doesn't matter if code is being developed in C, C++, or the Java™ programming language, they all share the same benefits of VIS acceleration.
- It maximizes the speedup achieved, while hiding details of the underlying target machine. UltraSPARC libraries are not only enhanced by use of VIS where possible, they are finely tuned for maximum performance on each version of the UltraSPARC processor where they reside. Thus, they

---

1. A set of VIS macros for C language programming is included in the VIS Software Developer's Kit (VSDK), available for download from the Sun web site, [www.sun.com](http://www.sun.com).

serve to insulate programs from hardware changes both when moving across different families of UltraSPARC processors, or from generation to generation within the same family.

- It minimizes the cost of using VIS. Calling library routines saves programmers the time and effort of writing the routine themselves. In the Solaris 10 OS, many libraries (including mediaLib™) are bundled into the operating system itself; in earlier versions of the Solaris OS, these libraries are freely available for download (at [www.sun.com](http://www.sun.com)).
- It speed the deployment of VIS. In particular, Sun's mediaLib library was developed specifically to exploit the potential of VIS, while making deployment of VIS-enhanced routines fast and painless. It now contains thousands of VIS-enhanced routines, covering a wide variety of functions for algebra, graphics, imaging, signal processing, video, and other uses.

## 4.0 The mediaLib Library for VIS Instructions

---

The heart of Sun's campaign to foster the use of VIS, wherever appropriate, is the mediaLib library, a performance-tuned collection of application subroutines for building portable, high-performance multimedia applications.<sup>1</sup> The MediaLib library implements key algorithms for:

- Imaging - data format conversion, spatial operations, image generation and copying, arithmetic and logical operations, color space conversion, geometric and radiometric operations, image statistics, and Fourier domain processing.
- Volumetric Imaging - volume data processing and volume visualization.
- Video Processing - DCT, motion compensation, motion estimation, and color conversion that can be used for video conferencing and JPEG and MPEG processing.
- Signal and Audio Processing - digital signal filtering, signal generation, codecs and transformations.
- Graphics - 2D and 3D primitives, rendering and texturing.

---

1. The mediaLib library focuses on providing performance-critical functions needed in VIS-intensive application areas, primarily imaging, graphics, audio/video and linear algebra. This library is continually revised and expanded, as Sun's partners and software developers identify additional or other key functions that can be accelerated using VIS instructions. However, the benefits of VIS are by no means confined just to the functions in mediaLib. Other libraries, including OpenGL, XIL, and XGL also contain routines that have been enhanced with VIS instructions. For that matter, the Solaris Operating System itself included components that are VIS accelerated, so all Solaris OS users on SPARC derive some degree of benefit from VIS, too.

- Linear Algebra - vector and matrix algebra.

Work on what ultimately would become mediaLib started in 1995, as part of the rollout plan aimed at facilitating adoption of the initial generation of UltraSPARC processors. Initially, it was known as VDI (for “Virtual Device Interface”). In June 1996, in one of the first accounts to both raise and address the issue of VIS usability, VDI is circumspectly described as “a library of thin algorithms precoded and optimized in the VIS instructions ... a library of inner loops for signal processing algorithms” [9]. By September 1996, however, VDI had been rechristened as “mediaLib” and recharacterized in a joint press release issued by Sun and Apple as a general API for “multimedia-capable processors” enabling “software and hardware developers ... to access the advanced multimedia capabilities” of processors like UltraSPARC and PowerPC<sup>1</sup> via “functions highly tuned for imaging, video, audio, algebra and graphics” [18]. Although this announced joint ambition to position mediaLib as a general API for multimedia-capable processors ultimately proved fruitless, the description of the functionality to be provided by mediaLib is essentially accurate as of this September 1996 statement of intention.

The mediaLib library, which has been continually refined and extended since its 1996 announcement, has recently (May 2005) been upgraded once again, to version 2.3 or MultiThreaded (MT) mediaLib. This latest MT mediaLib library provides about 2000 functions – primarily for linear algebra, and image, signal and video processing – newly tuned for the UltraSPARC IV generation of dual-core processor (which implements VIS 2.0). A

---

1. One of the odder things about this joint press release was that, at the time of the announcement, PowerPC was not a “multimedia-capable processor” either in need of or able to benefit from a library like mediaLib. Indeed, in September 1996, given even the MAX-2 enhancements to HP’s PA-8000 series processors were far too limited to support a wide-ranging functional set like mediaLib, Sun was the only vendor shipping a processor line fitting this description. Even in terms of announced plans, the only vendor at that time to have revealed definite intentions to add general SIMD functionality was Intel (with MMX) – but Intel was notably absent from this joint press release, and there is no evidence they ever supported the idea of a cross-platform library of SIMD acceleration functions. As for PowerPC, specifically, by the time word about VMX (AltiVec) first leaked out in mid-1997, Apple had decided to enable its use by enhancing QuickTime with its own independent set of “Media Layer acceleration APIs” [19]. Hence, the larger Sun+Apple initiative to create a common cross-platform API for SIMD-enhanced processors quickly came to nothing. Albeit – a point that may not have been apparent as early as September 1996 – in the search for competitive advantage, the several sets of SIMD extensions that ultimately were implemented varied significantly from each other. This fact, in turn, meant each set of SIMD extensions supported its own distinct range of potential routines, greatly diminishing the value of a common API (which could cover only a limited set of core functions).

later version of MT mediaLib will support the initial generation of the radical Chip MultiThreaded (CMT) 32-way Niagara processor (which implements a subset of the VIS 2.0 instructions). This later version of MT mediaLib should be available by the time systems based on the Niagara processor begin shipping.

As was true with previous upgrades of mediaLib, the new MT mediaLib retains the same API as earlier mediaLib versions, so there is no need to modify any source code to take advantage of it. As a result, all UltraSPARC IV systems are positioned to benefit from this upgrade immediately,<sup>1</sup> since the Solaris OS treats each dual-core UltraSPARC IV processor as two logically distinct processors. Thus, even a uniprocessor UltraSPARC IV box behaves as if it were a 2-way system, while a two processor box acts as a 4-way system, a four processor box acts as an 8-way system, and so on.

In essence, the MT part of MT mediaLib is a software layer developed on top of the mediaLib library, using OpenMP. When used with a sufficiently large data set on a multithreaded system, this new software layer will partition the data set into independent subsets, and process each subset in parallel. Obviously (excluding any overhead costs associated with partitioning the data set), the potential speed-up for affected algorithms, relative to processing the same data set in a single thread (unpartitioned), can approach or (in some cases) even exceed the number of separate threads invoked.

The number of threads used to execute a MT mediaLib routine either can be set explicitly by the system user, before running the application, or automatically at application runtime. If the number of threads is set automatically at runtime, the new MT software layer attempts to determine the largest number that makes sense, in order to maximize the multithreaded capability of the host system. Specifically, if the function can't be parallelized or its target data set is too small to support subdivision, then the number of threads is set to 1, and the mediaLib function is called in single thread mode. If the function can be parallelized and the associated data

---

1. On a Solaris 10 (or later) system. Systems running Solaris 8 OS or Solaris 9 OS require installation of a patch to use MT mediaLib, respectively:  
117557-01 SunOS 5.8: Microtasking libraries (libmtsk) patch  
117560-01 SunOS 5.9: Microtasking libraries (libmtsk) patch  
These patches are freely available for download from the SunSolve web site at:  
<http://sunsolve.sun.com/>

set is large enough to make parallelization profitable, then the thread number is set to the number of on-line processors.

Alternatively, the system user can explicitly control the number of threads by setting it to the PARALLEL environment variable, where PARALLEL is any positive integer between 1 and the number of on-line processors.<sup>1</sup> If PARALLEL is not set at all, is set to 0 (or less), or is set to a number larger than the number of available processors, then the number of threads defaults to the automatically determined value (either 1 or the number of on-line processors).

In any case where the number of threads is set above 1, mediaLib checks the size of the associated data set to see if it is large enough to support that many threads. If not, the number of threads is adjusted downwards to the maximum number that is supported by the data size. The data set is then partitioned into the given number of subsets, and calls are made to the specified function to process each subset in a separate thread.

Although applications can be rebuilt to link with MT mediaLib, ensuring that MT mediaLib always will be used any time the application is started, it is also possible for an application to take advantage of MT mediaLib without being rebuilt. In the later case, however, it is necessary to explicitly pre-load MT mediaLib at runtime, by setting the LD\_PRELOAD environment variable appropriately before the application is invoked.<sup>2</sup>

- 
1. The syntax here will vary, depending on the command shell being used. In the C Shell, the command is just:  
`setenv PARALLEL NTHRS`  
In the Bourne/Korn Shell, it takes two lines to set this equivalence:  
`PARALLEL=NTHRS`  
`export PARALLEL`  
In addition to PARALLEL, there are other environment variables that can be set to change OpenMP runtime behavior. See section 5.3 on "OpenMP Environment Variables" in *Sun Studio 10: OpenMP API User's Guide* for more details.
  2. Again, the syntax for setting LD\_PRELOAD varies, depending on the command shell being used. In the C Shell, the command is:  
`setenv LD_PRELOAD libmllib_mt.so`  
In the Bourne/Korn shell, the commands are:  
`LD_PRELOAD=libmllib_mt.so`  
`export LD_PRELOAD`

## 5.0 Future Directions

---

VIS itself already has been upgraded once, from VIS 1.0 to VIS 2.0, with the release of the UltraSPARC III processor in 2000. The major addition to VIS in version 2.0 is inclusion of a “byte shuffle” instruction, intended to make the reordering of packed data fast and easy. This speeds operations where packed data must be handed off from one VIS instruction to another, when the two instructions assume incompatible orderings of the given data.

With a number of new CMT processors scheduled to appear over the next several years, including Niagara 2 and Rock, opportunities exist to continue to improve both mediaLib and the underlying VIS instruction set itself. In particular, VIS 3.0 will be implemented in future versions of Sun’s SPARC processor, providing the most significant set of enhancements since VIS 1.0 was added to the first generation of UltraSPARC processors. VIS 3.0 will add about 50 new instructions, further expanding the range of VIS applicability. These will include added general SIMD integer functions, special-purpose instructions for cryptography and video decoding, and SIMD operations on dual single-precision floating-point values (in addition to current SIMD operations on dual 32-bit integers). To support VIS 3.0, the MT mediaLib library will experience a corresponding expansion, providing new routines to fully exploit the new capabilities of this upgrade to Sun’s SIMD instruction set extension [20].

## 6.0 Bibliography

---

[1] An excellent contemporary overview of the upgrade of the SPARC architecture from the 32-bit V8 standard to the compatible 64-bit V9 standard is given in “SPARC V9 Adds Wealth of New Features” by Brian Case, *Microprocessor Report*, Vol. 7, No. 2, February 15, 1993, pp. 1,6-11.

[2] See “Sun Details Extensive Plan for SPARC CPUs” by Linley Gwennap, *Microprocessor Report*, Vol. 7, No. 5, March 29, 1993, pp. 20-21.

[3] See “UltraSPARC Unleashes SPARC Performance” by Linley Gwennap, *Microprocessor Report*, Vol. 8, No. 13, October 3, 1994, pp. 1,6-9.

[4] See “UltraSparc Adds Multimedia Instructions” by Linley Gwennap, *Microprocessor Report*, Vol. 8, No. 16, December 5, 1994, pp. 16-18.

[5] See “UltraSPARC Rolls Out at Target Clock Speed” by Linley Gwennap, *Microprocessor Report*, Vol. 9, No. 7, May 30, 1995, pp. 1,5-7.

[6] Sun announced immediate availability of a complete line of workstations and servers built around the UltraSPARC I processor on November 7, 1995. Among the dozen press releases bearing that date, see “SUN CHANGES RULES OF THE GAME WITH NEW SYSTEMS THAT WILL LAUNCH USERS INTO NETWORK AGE” for details.

[7] Also see “UltraSparc to Pick Up Speed in 1996” by Linley Gwennap, *Microprocessor Report*, Vol. 9, No. 15, November 13, 1995, for information about the new UltraSPARC-based systems together with related disclosures about new versions of the UltraSPARC processor scheduled to appear within the next year.

[8] See the Intel press release, “Intel Launches New Ad Campaign For MMX Technology That Puts The Fun In Computing – “Seinfeld” Star Jason Alexander and Intel “Bunny People” Featured in New Commercials” dated Santa Clara, CA January 22, 1997, for details of this multi-million dollar ad campaign.

[9] From “MPUs embrace multimedia instructions” by Ron Wilson, *EE Times*, June 10, 1996.

[10] Quoting the title of the “Guest Editor’s Introduction” to this special feature section, “Media Processing: A New Design Target” by Ruby B. Lee and Michael D. Smith, *IEEE Micro*, Vol. 16, No. 4, August 1996, pp. 6-9.

[11] See “VIS Speeds New Media Processing” by Marc Tremblay, J. Michael O’Connor, Venkatsh Narayanan, and Liang He, *IEEE Micro*, Vol. 16, No. 4, August 1996, pp. 10-20.

[12] See “MMX Technology Extension to the Intel Architecture” by Alex Peleg and Uri Weiser, *IEEE Micro*, Vol. 16, No. 4, August 1996, pp. 42-50. The other two articles in this special feature section did not describe instruction set extensions for general purpose processors but rather a couple of specialized media processors, one from Texas Instruments (the TMS320C82, a DSP optimized for multimedia processing) and the other

from MicroUnity (the much anticipated but never realized MediaProcessor).

[13] MAX (an acronym for “Multimedia Acceleration eXtensions”) was the name HP gave to the small set of SIMD instructions included in their PA-RISC processors. MAX-1 was a set of 5 SIMD instructions for MPEG-1 decoding, exclusive to the PA-7100LC (first shipped in January 1994). MAX-2 was an expanded set of 10 SIMD instructions included in the 64-bit PA-RISC 2.0 standard (first shipped in April 1996). A thorough account of MAX-2 and its minimalistic design philosophy is given in “Subword Parallelism with MAX-2” by Ruby Lee, IEEE Micro, Vol. 16, No. 4, August 1996, pp. 51-59.

[14] See “The Visual Instruction Set (VIS) in UltraSPARC” by L. Kohn, et. al., Proc. Comcon, IEEE CS Press, 1995, pp. 462-469.

[15] See “MPEG video decoding with the UltraSPARC visual instruction set” by C. Zhou, et. al., Proc. Comcon, IEEE CS Press, 1995, pp. 470-475.

[16] See “Realtime MPEG Video via Software Decompression on a PA-RISC Processor” by Ruby Lee, Proc. Comcon, IEEE CS Press, 1995, pp. 186-192.

[17] For example, see “Optimization and code parallelization for processors with multimedia SIMD instructions” by Francois Ferrand, August 27, 2003. This paper presents work done for the author’s master’s thesis to support the automatic generation of code on processors optimized with multimedia instructions.

[18] See the joint Sun/Apple press release, “Apple and Sun Join Forces to Create High-Performance Enterprise Intranet and New Java-Based Component Development and Multimedia Technologies” dated September 18, 1996 from the Networld+Interop conference held in Atlanta, GA.

[19] See “AIM to fire back at Intel with MMX-style chips” by David Morgenstern, MacWeek, June 2, 1997.

[20] For a complete survey and discussion of the improvements made to VIS over the years, with many examples of its use and effectiveness in

accelerating target algorithms, see the Sun Microsystems White Paper, "The VIS Instruction Set" Version 1.0 dated June 2002.

Copyright © 2005 Sun Microsystems, Inc. All Rights reserved. "Sun, Sun Microsystems, the Sun logo, UltraSPARC, Solaris, mediaLib, VIS, Java are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries." All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the US and other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

"Sun, Sun Microsystems, le logo Sun, UltraSPARC, Solaris, mediaLib, VIS, Java sont des marques déposées ou enregistrées de Sun Microsystems, Inc. aux Etats-Unis et dans d'autres pays." Toutes les marques SPARC, utilisées sous licence, sont des marques déposées ou enregistrées de SPARC International, Inc. aux Etats-Unis et dans d'autres pays. Les produits portant les marques SPARC sont basés sur une architecture développée par Sun Microsystems, Inc.