The Mpact™ Media Processor Redefines the Multimedia PC

Pete Foley
Director of Product Planning
Chromatic Research

Abstract

By implementing all 7 key multimedia functions (MPEG video, 2D graphics, 3D graphics, audio, fax/modem, telephony, and videophone) using a single media processor, Chromatic Research has redefined the multimedia PC. Utilizing a high bandwidth RAMBUS RDRAM for media memory and a VLIW SIMD internal architecture, the Mpact M1 media processor is able to deliver 2 BOPS of sustained integer performance at a price point significantly below alternative discrete hardware implementations. By cooperatively solving multimedia tasks with the host processor, by supporting Win '95 APIs, and by supporting PC legacy needs such as VGA graphics and FM audio, the M1 raises the bar on base PC multimedia functionality.

Enabling technologies

The intersection of several key technology trends has enabled the creation of media processors such as the Mpact M1 that are capable of eliminating most of the limitations of first generation multimedia PCs. Incorporating media processors such as the M1 onto the motherboard creates a second generation of multimedia PCs with significantly enhanced base level capabilities (see Table 1).

The first key technology is the development of very high speed memories with narrow (low pin count) interfaces, such as RAMBUS RDRAM, that have costs approaching standard commodity DRAM. Using an 18-pin interface, a single chip 2-MB media buffer provides 500 MB/s of peak bandwidth to implement all of the memory needs of the media processor, including framebuffer, program store, data store, wavetable memory, etc. Such a pin-efficient media memory interface allows the M1 to be manufactured in a low cost 240-pin plastic package, while providing sufficient free pins to implement the other necessary chip interfaces (see Figure 1). The narrow RAMBUS channel also permits Mpact systems to have media memory sizes of 1MB granularity, something impossible to implement in a cost effective manner for typical 64-bit wide EDO RAM or SDRAM based systems.

A second key technology is the development of consistent API's which allow media procesors to transparently accelerate the multimedia needs of application software. This need has been met with the introduction of the DirectX API's from Microsoft; DirectDraw, Direct3D, DirectPlay, and DirectVideo. The media processor and the host processor can now cooperatively solve the multimedia compute needs of the PC.

A third key technology is the constantly improving silicon fabrication technology, which has provided silicon feature sizes that allow the creation of cost effective media processors with performance to match concurrent multimedia compute needs. Implementation of the M1 in a .35 micron 3-layer metal CMOS technology allows the processor to pack 2 BOPS of sustained integer performance into 1.5M transistors on a single 65mm sq. die.

Dedicated motion estimation hardware

The M1 incorporates dedicated logic designed to accelerate one of the most compute intensive multimedia functions, MPEG motion estimation. The motion estimation block of the M1 pushes the sustained integer capability of the M1 to 20 BOPS, as shown in Figure 2, while utilizing only a few percent of the die area.

Programmable paradigm advantages

The adoption of a programmable paradigm for multimedia function implementation provides obsolescence protection for the consumer through upgradeability, and provides economies of scale for PC OEMs by allowing the creation of common hardware platforms that support late binding of product feature sets.

Video	MPEG1 encode/decode 30fps, MPEG 2 decode (with AC-3 audio), JPEG
2D Graphics	VGA emulation,GDI, DCI, DirectDraw with all bit BLTs, ternary ROPs, and a hardware cursor
3D Graphics	3D-DDI and Direct3D with z-buffering, double-buffering, texturing
Audio	MIDI, Wavetable, Waveguide, 3D sound, Dolby AC-3, and DirectSound
Fax/Modem	Up to V.34bis (33.6Kbps); supports DSVD
Telephony	Full-duplex speakerphone, Win 95 TSPI and DirectPlay, and AT+V command set
Videophone	H.320 over ISDN and H.324 over POTS

Table 1: Mpact M1 System Functional Capabilities

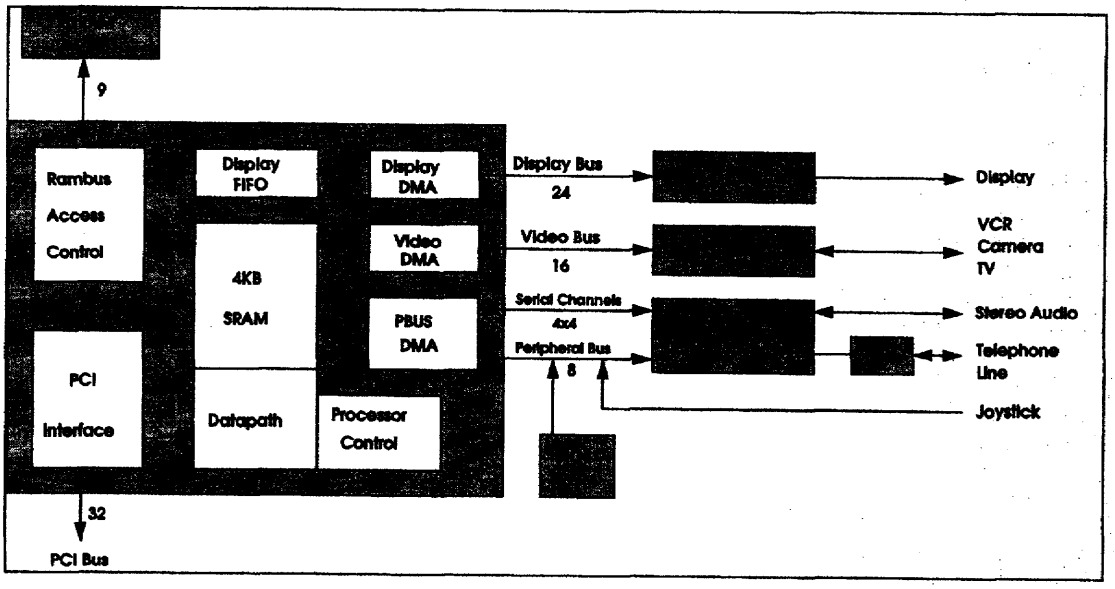

Figure 1: Mpact system block diagram

Cooperative computing

The Mpact system is designed to cooperatively solve multimedia compute needs with the host processor. By load balancing with the host processor, an Mpact system takes advantages of the strengths offered by each type of processor. Multimedia or "natural data types" are characteristically streaming integer data with stringent real time processing constraints. The M1 excels at processing these data types with its efficient real-time kernel, vector operations, multiple ALUs, SIMD architecture, fast private memory system, and hand crafted multimedia algorithm inner loops. The M1 is typically 10x faster than an Intel P100 at such calculations.

The host processor is optimized for executing large applications requiring large virtual address spaces with a paged virtual memory addressing system. It is also quite efficient at floating point calculations and calculations involving table lookups. The Intel P100 is perhaps 10x faster than the M1 at these types of calculations.

Windows and third party applications can arbitrarily disable interrupts, which can create serious headaches for latency sensitive data streams, particularly audio. The human ear is extremely sensitive to audio discontinuities. The Mpact system is less exposed to interrupt latency effects because

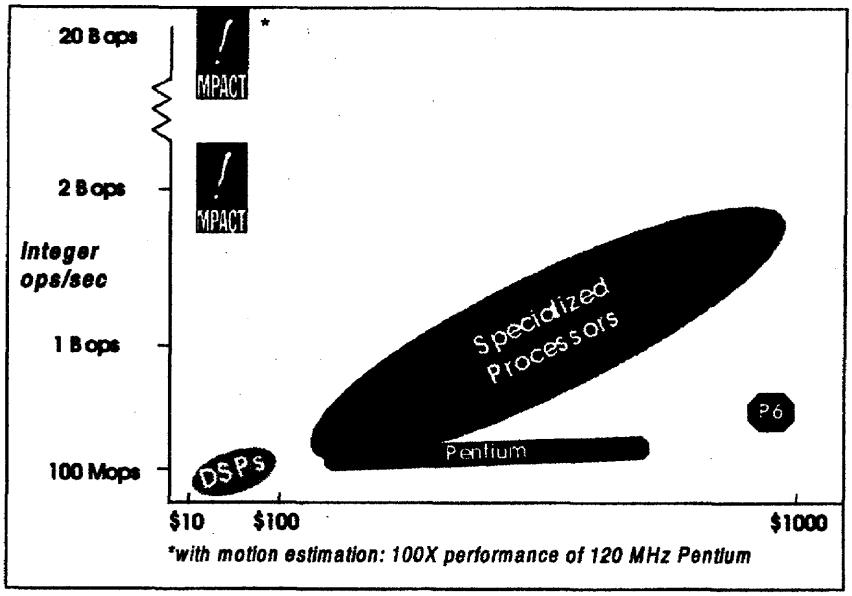

Figure 2: Price/performance chart

there is less processing performed on the host than would be in an NSP based system. In general, the Mpact system architecture isolates the real-time processing burden from the host CPU.

A good example of cooperative computing in an Mpact based system is 3D graphics. The host processor, with its efficient floating point hardware, is used for the geometry calculations, while the M1 is used to render the polygons into the framebuffer located in RDRAM. Another example is MPEG decode. The host processor is used to parse the incoming video and audio data streams, and to perform the variable length decoding, while the M1 is used to perform the Inverse DCT (IDCT), motion compensation, colorspace conversion, and image scaling calculations.

Concurrency

A key aspect of quality delivery of multimeda, and something that is currently not well benchmarked in the market, is the ability of the multimedia PC to process concurrent tasks. If a significant percentage of the host processor cycles are consumed performing a software MPEG-I decompression algorithm, how well can the host be expected to take on additional tasks such as a V.34 modem? Just adding the integer computation requirements for multiple multimedia tasks is only the first step in determining the concurrency capabilities of a system. Memory bandwidth, memory latency, interrupt latency, and a number of other issues must also be considered.

Chromatic believes that rather than eclipsing the need for a media processor, the increasing compute capabilities of anticipated next generation host processors, such as the Intel P55C, when combined with an Mpact media processor, will offer the increased concurrency capabilities the consumer will expect.

PCI bus insulation

One of the advantages offered by an Mpact based multimedia system is that the PC is insulated from most bandwidth and latency limitations of the PCI bus as the bulk of the high bandwidth traffic is between the M1 and the RDRAM.

Due to the fact that arbitrary third party devices can potentially be plugged into the main PCI bus, there can be significant variations in bus latency and available bandwidth. This is particularly true with devices that are not PCI 2.1 compliant. Another concern is bus arbitration variability.

In the case of systems with multiple PCI bus masters, a master can asynchronously steal bus cycles without the other master necessarily being aware of it. This can lead to problems in systems expecting isochronous data transfers. An example of this is joystick read data, where the joystick read count proportionality to time may become jeopardized.

Why does the M1 perform so well?

How is it the M1 can provide so much multimedia performance in 1.5M transistors running at 62.5 Mhz compared to a typical general purpose CPU running at 100Mhz? It is instructive to examine where the transistors are expended. A general purpose CPU is optimized for executing large applications requiring large virtual address spaces with a paged virtual memory addressing system, and to provide a robust protected execution environment for a wide variety of application software. Consequently there are a lot of transistors expended in multiple large set associative cache memories and their tag memories, virtual paged memory support such as a translation lookaside buffer, a branch target cache, a floating point unit, and multiprocessing/bus snooping support for cache coherency. If the processor is deeply pipelined there may be extensive hardware interlock support, and if the processor is superscalar there are likely to be reservation stations, a reorder buffer, a register alias table, and much more complex instruction issue logic. None of these transistor expenditures apply to the M1, where more than 50% of the 1.5M transistors are expended in the multi-ALU integer datapath and a 4Kbyte multiported SRAM.

The Mpact M1 execution environment is more controlled and more application specific than a general purpose CPU. With its emphasis on streaming data types and its close high speed RDRAM memory, the M1 can dispense with large on-chip cache memories. There is no need for floating point logic when the focus is multimedia data types, and the resulting small die size allows for the addition of fixed function logic blocks, such as the motion estimation unit, with very small increases in die cost. The M1 instruction set was targeted for multimedia data types; the M1 employ saturation arithmetic, Multiply Accumulate (MAC) instructions with guard bits, and specialized pixel manipulation instructions, for example.

But these silicon efficiencies do not come without some cost. The M1 affords little in the way of hardware protection mechanisms as on-board memory is primarily managed by software. The M1 pipeline does not guarantee hazard free execution instruction. Any statically predictable hazard conditions must be managed by the compiler.

Managing RDRAM latency

One of the criticisms leveled against RDRAM is that the first access latency, particularly if the first access is a sense amp cache (or bank) miss, significantly reduces the net effective bandwidth. The short answer as to how RDRAM latency effects are minimized is that the Mpact system deals primarily with streaming data types such as video and display refresh data, so that data is transferred to/from RDRAM in large bursts.

Tiling is employed to improve framebuffer memory performance[1]. Tiling improves the likelyhood that sequential framebuffer accesses will hit in the RDRAM sense amp caches. Additional techniques such as aggressively pipelining RDRAM memory interfaces, and double buffering RDRAM access are also employed to increase net RDRAM bandwidth.

The Mpact system is capable of driving a 1280 x 1024 x 18bpp display @ 75 Hz. Such a display uses 270Mbytes of sustained bandwidth from RDRAM while still having bandwidth left over for other functions.

Mpact system hardware

A single 18-Mbit RDRAM provides the standard 2MB of media memory through the 9-bit RAMBUS channel (see the block diagram in Figure 1). The other chip interfaces include a 32-bit PCI bus interface, a 24-bit RGB graphics interface, the 16-bit YUV video interface, and the peripheral device interfaces. The peripheral device interfaces include four 4-bit serial interfaces and a microprogrammable byte parallel peripheral bus. The peripheral device interfaces facilitate connection to external devices such as BIOS ROMs and audio/modem CODECs. The flexibility of the video and peripheral device buses, and in particular the microprogrammability of the byte parallel peripheral bus, allow support for a wide variety of present and future peripheral devices.

The internal architecture of the chip mirrors the external bus structure. Referring again to Figure 1, the M1 includes a very fast multiported SRAM, called the SMEM. The SMEM is both an I-cache, a D-cache, a DMA buffer memory, and a register file. The SMEM has 4 read and 4 write ports which operate every clock cycle. Each port is 8 bytes, or 72 bits wide. The SMEM contents are managed by software.

The RAMBUS channel operates at 250 Mhz, and the processor core is slaved to 1/4 that clock rate, or 62.5 Mhz. Data on the RAMBUS channel is transferred on both rising and falling clock edges (every 2ns), giving the 500MB/s peak transfer rate. Since the smallest unit of transfer to RDRAM is 8 bytes, it takes 16ns to transfer 72-bits, or one DWORD

of data. The DWORD is the fundamental unit size in the Mpact system. Since the M1 core clock rate is also 16ns, this gives rise to the natural internal bus size of one DWORD, or 72-bits.

Instruction Set Architecture (ISA)

The daily feedback of the Chromatic software team with the M1 architects was critical in shaping the M1 ISA. The M1 ISA was refined through many iterations as the multimedia algorithm inner loops were simulated on the evolving instruction set.

The M1 instruction pipeline is a classic four stage pipeline: fetch, decode, execute, and writeback. Generally all instruction execute in one cycle, however multiply instructions incur a two cycle latency (with one clock pipelined throughput), and Load/Store instructions can transfer from 1 to 256 bytes between the SMEM and RDRAM.

The SMEM contents are generally software managed, however there is a small I-cache within the SMEM consisting of 2, 4, or 8 lines of 128 bytes/line. A line will automatically be fetched from RDRAM into SMEM on an I-cache miss, however the I-cache line fetch latency can be explicitly minimized by software via the I-Load instruction, which will initiate an I-cache line fill while allowing instruction execution to continue.

Instructions are organized as instruction pairs, with an instruction pair always taking up one DWORD (eight 9-bit bytes) in memory. However, individual instructions themselves vary in length from 3 to 5 bytes. Hence a 5 byte instruction can be paired with a 3 byte instruction to form an I-Pair. This provides improved code density, while still maintaining "fixed length instructions" that are DWORD aligned in memory. A single I-Pair is dispatched every cycle. Each I-Pair is interpreted as a sequential I-Pair, a concurrent I-Pair, or a vector I-Pair. The M1 supports interpreting virtually any I-Pair as a vector version of that instruction pair (except I-Pairs that include control flow and long latency instructions, such as Load or Store). A single I-Pair can be repeated up to 127 times with auto-incremented addressing. Interrupts occur at I-Pair boundaries.

The M1 also supports a repeat instruction that allows a block of instructions (minimum size 2) to be repeated up to 512 times without any branching or counting overhead.

Branches are statically predicted by the compiler, with a 0 cycle penalty for predicted branches, and a 2 cycle penalty for mispredicted branches.

M1 datapath

Figure 2 shows the M1 datapath, which consists of five functional units called ALU Groups. Each ALU group consists of an 8 byte (72-bit) arithmetic unit that always operate on DWORD sized quantities. Each DWORD can operated on in 1, 2, 4, or 8 byte chunks. Use of 9-bit bytes in the Mpact system lends itself nicely to 16-bit audio applications, providing two bits of extra precision for intermediate results without the expense of going to a 24 or 32-bit system. Extensive early algorithm simulation indicated 18-bits provided sufficient precision for 16-bit audio.

ALU Group 1 is a shift and align unit, ALU Group 2 is a general purpose ALU, and ALU group 3 is also a general purpose ALU that implements a ternary operation superset of ALU2. Multiplication sequentially utilizes ALU Groups 4 and 3. ALU Group 4 uses 2-bit booth encoding and 4:2 compactors in a Wallace tree structure to compress the partial products down to two results for input to ALU Group 3 for the final addition in the second cycle of the multiply. The multipliers can be configured to produce eight 9 x 9 ->18-bit results, four 18 x 18 -> 44-bit results, or two 24 x 24 -> 44-bit results. Group 5 is a dedicated motion estimation unit consisting of some 400 arithmetic units. This ALU group, which consumes only a few percent of the die area, provides 20 BOPs of compute performance for MPEG motion estimation.

To achieve the sustained 2.0 BOPs compute rate, the first 4 ALU groups must compute eight byte wide results per 16 ns cycle. Like the multiply instruction, there are other more complex instructions that trigger operations in multiple ALU groups.

All of the functional units, as well as the SRAM, are interconnected with a 792-bit wide bus that functions as a crossbar, allowing any result to be used as an input or written to the SMEM in the following cycle. In other words, any of eleven 72-bit result outputs can be input to any of nineteen 72-bit inputs. Another way to look at this is that the crossbar supports a bandwidth of 19 sinks x 8 bytes/sink x 62.5Mhz = 9.5 GB/s of bandwidth.

M1 graphics datapath

The display refresh is a DMA process that does not require M1 intervention. Data is fetched from RDRAM and placed directly into the display FIFO. The M1 can support 8, 15, 16, 18, and 24bpp display formats, but the preferred format is 18bpp. Using just two 9-bit bytes for the18bpp (6,6,6) color, allows the Mpact system to provide near 24-bit color quality

without the expense of a third byte. Display sizes of up to 1280 x 1024 x 18bpp @ 75Hz and 1024 x 768 x 24bpp @ 75Hz are supported.

M1 video datapath

The 16-bit half-duplex YUV databus supports video data at either PAL or NTSC data rates (approximately 27MB/s). Video data is buffered in SMEM before being transferred to the RDRAM under DMA control.

The flow of video data through the Mpact system utilizes offscreen YUV surfaces in RDRAM to buffer the YUV data as it is brought in from the video bus. For display in a window, for example, the YUV surface data would be colorspace converted, converted to the required color depth, and probably scaled before being written to the framebuffer in RDRAM.

M1 peripheral device interfaces

The M1 Peripheral Bus externally consists of four serial channels and one byte parallel bus. Internally, the bus resources are organized as 14 virtual channels, where each channel is unidirectional and half-duplex. Each channel provides a DMA resource for transfer of data to/from RDRAM and a peripheral device via the SMEM, and can be assigned to any of the serial channels or parallel bus devices. 6 channels are read channels and 6 are write channels. Four of the read channels and four of the write channels can be configured either for serial or parallel operation. Of the additional two channels, one is dedicated to reading the BIOS ROM, and another is used for reading and writing the Peripheral Bus internal control registers.

The byte parallel bus interface is microprogrammed using a microprogram store of modest size. Most frequently used transaction protocols for active devices are kept resident in the microprogram store, and when an unusual event occurs, such as an error condition in a WAN controller, then the appropriate error transaction routine is dynamically loaded into the microprogram store from RDRAM. This greatly increases flexibility, lowers design risk, and minimizes the amount of fixed function logic needed in the peripheral bus controller.

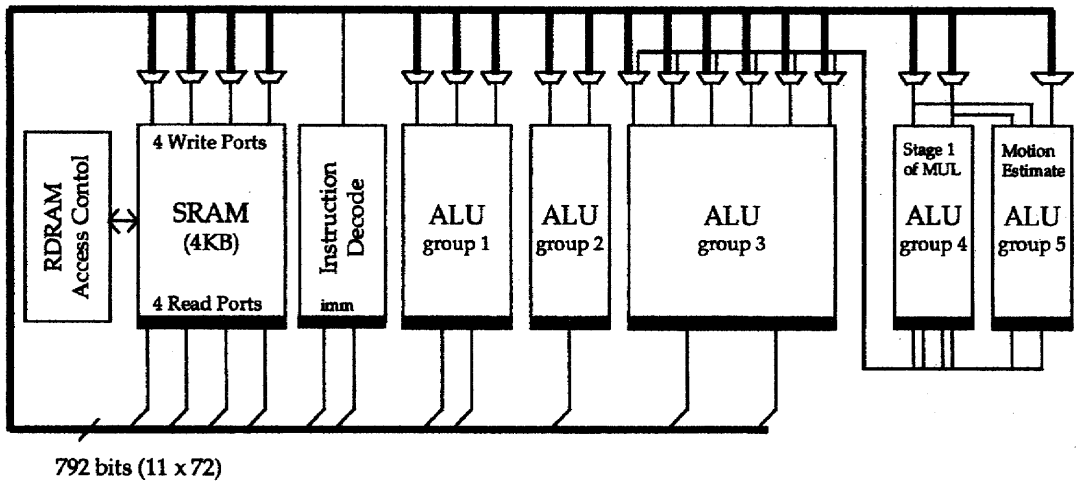

Figure 2: M1 datapath block diagram

Mpact system software

The Mpact system software consists of the host resident Resource Manger (RM), the M1 resident Mpact Real Time Kernel (MRK), and the associated mediaware modules that implement each of the multimedia functions. Figure 3 shows the Mpact system software architecture.

Mpact mediaware

The dark grey boxes in Figure 3 indicate software that is "host independent" and tied to the M1. A mediaware module consists of both host resident and M1 resident code that reside under Win 95 APIs and interact with the Chromatic Resource Manager (RM) and the Mpact Real-time Kernel (MRK) to implement a multimedia function. A mediaware module is a

vertical "slice" through Figure 3 of the Chromatic software components.

A good way to think of a mediaware module is as virtual hardware. When a function (say MPEG1 encode) is enabled on a motherboard, although the function is implemented by software running on both the host and the M1, if the motherboard is removed and put into another machine, the ability to encode MPEG1 goes with the motherboard, hence the "virtual hardware" aspect.

Mediaware modules have relatively small footprints in RDRAM. Some examples include FM audio synthesis (10KB), Dolby AC3 (50KB), V.32bis modem (100KB), and V.34 modem (200KB). The V.34 modem has one of the larger mediaware RDRAM memory footprints.

Resource Manger

The Mpact Resource Manager (RM) is a host-resident resource manager that:

1) Controls invocation of new real time tasks.
2) Performs dynamic real-time task linking and loading
3) Performs dynamic RDRAM heap management
4) Provides host to M1 IPC services
5) Establishes and manages the backoff framework

The RM must allocate tasks while managing all of the key Mpact system resources, RDRAM memory, RDRAM memory bandwidth, and M1 processor compute bandwidth. The RM must also insure that the host itself is not oversubscribed. PCI bandwidth and latency is almost never a critical resource as virtually all of the high bandwidth traffic occurs between the M1 and RDRAM.

Backoff management is one of the key functions of the RM. If the user should ask the system to do too much, and the Mpact multimedia system becomes oversubscribed, there are a variety of backoff/graceful degradation strategies that can be employed. Examples include reducing the amount of 2D GUI acceleration, backing off on video frame rates, reducing the number of audio/MIDI voices, or reducing modem speeds. The default backoff strategies make resource management issues transparent to the consumer in all but extreme cases, with the user still having the option to customize the backoff strategies to suit his or her tastes.

Mpact Real-time Kernel

The Mpact Real-time Kernel (MRK) is a nearest deadline scheduler that assigns thread deadlines dynamically where precise deadline timing is guaranteed by programmable interrupt services. Interrupts themselves are triggered by events such as timer interrupts, CODEC FIFO watermarks, and host-originated semaphores.

The MRK offers several advantages over alternative real-time kernels.

1) Task scheduling is dynamic and pre-emptive.
2) MRK allows dynamic reassignment between real-time and non-realtime status.
3) Supports asyncronous CODECs and other asyncronous devices.
4) Task prioritization need not be re-evaluated every time a task is added or deleted.
5) Programmers need not be aware of the entire system when subscribing for resources.

The MRK has a memory footprint in RDRAM of only 20KB.

Context switching

A key factor for high quality support of real-time tasks is the processor context switch rate and associated overhead. Audio processing in particular is very sensitive to context switching and interrupt response latency. In the Mpact system, interrupt latency for audio tasks is designed to be less than 2ms under all conditions.

Because of the closeness of the primary memory (RDRAM) to the processor, the high bandwidth bursting capability of the RAMBUS memory system (500 MB/s), and the lack of significant on-chip state to be saved per context, context switching on the M1 is quite fast. Although the context switch rate is dependent on the timing properties of the threads running on the system at any given time, 3K is a typical switch rate number. The total amount of state saved and restored per context switch can vary on a thread to thread basis from 96 to 3168 bytes, with an average observed context switch time of 12 μs. This gives a total M1 processor overhead for context switching of 3.6%.

The total M1 processor overhead for the MRK can be closely approximated by adding the context switching overhead to the interrupt servicing overhead (about 3%). This puts the total real-time OS overhead on the M1 at approximately 7% of processor cycles.

Audio subtask management

Even though MRK offers very fine thread control and rapid context switching, there is an unnecessary amount of overhead associated with using the MRK for audio subtask management. The XAPM audio subtask manager was created to minimize audio processing overhead. XAPM is a dedicated audio subtask manager that runs as a single thread under the MRK. XAPM itself uses round robin scheduling.

Using XAPM running under the MRK, the FM synthesis audio subtask (for example) has been implemented utilizing just 5% of the M1's compute bandwidth.

Status and future work

First silicon on the Mpact processor was received in early July of 1995. The Mpact system is scheduled for release in alpha form to key OEMs in January. Production shipment of the M1 and associated Mpact mediaware package is expected in June 1996.

Work is well underway on the follow-on Mpact processor, which will take advantage of the increase in RDRAM sense amp cache banks and split transaction protocols offered in the next generation RDRAMs to further reduce memory latency and increase net RDRAM bandwidth.

Additional information on the Mpact M1 can be obtained at www.mpact.com.

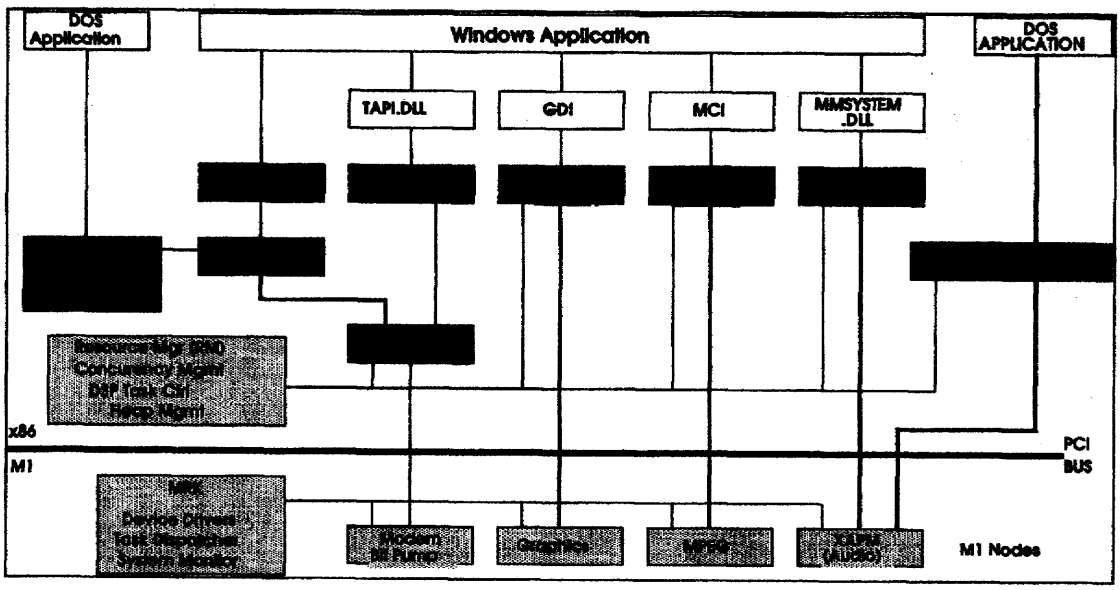

Figure 3: Mpact system software block diagram

References:

1. B. Garrett, "Applying RAMBUS Technology to Graphics", RAMBUS Inc., pp. 31-50.