Emotion Engine

Prashant Gandhi
Dept. of Electronics & Communication Engineering
Institute of Technology, Nirma University
Ahmedabad, India
13bec029@nirmauni.ac.in

***Abstract*— this paper presents an overview on CPU of the PlayStation2 video game console which is known as the Emotion Engine. The PlayStation2 also known as PS2 is a gaming platform. This paper gives the brief idea of Emotion Engine design and its various CPU components like CPU core, VPU, IPU, DMA unit and GIF. Working of each CPU components is discussed in this paper. This paper also contains the fabrication method and components used to make the Emotion Engine. Also its specifications and theoretical performance is also stated in this paper.**

Keywords-VPU; IPU; DMA units; GIF units

I. INTRODUCTION

The Emotion Engine was a central processing unit developed and manufactured by Sony Computer Entertainment and Toshiba to be used in the Sony PlayStation 2 Video game console. It was also used in early PlayStation 3 models sold in Japan and North America (Model Numbers CECHAxx & CECHBxx), to provide PlayStation 2 game support. Mass production of the Emotion Engine began in 1999 and ended in late 2012 with the discontinuation of the PlayStation 2.

The way Sony designed the PS2, it's not really possible to look only at the Emotion Engine, and ignore the rest of the system. We have got to seem at the Emotion Engine with in the context of the overall design of the PS2 because, unlike a modern PC's CPU, the Emotion Engine is not really a general purpose computing device. The CPU of PC is designed from the ground to run SPEC benchmarks. Err, application code as quick as possible. The PS2 is in a slightly different scenario. The PS2's designers had the luxurious of coming up with hardware whose main purpose is to run one type of application extremely well: the 3D game. The PS2 can run web browsers, mail clients, and other types of software, but that's all secondary. The main thing the PS2 does in 3D gaming which means generating the kind of immersive sound and vision that places us in virtual world. Nearly all of the PS2's hardware is developed to providing some specific portion of that audiovisual gaming experience.

Its specifications are: Clock frequency: 294 MHz, 299 MHz (later versions). Instruction set: MIPS III, MIPS IV subset, 107 vector instructions 2 issue, 2 64-bit fixed point units, 1 floating point unit, 6 stage pipeline Instruction cache: 16 KB, 2-way set associative Data cache: 8 KB, 2-way set associative Scratchpad RAM: 16 KB Translation look aside buffer: 48-entry combined instruction/data Vector processing unit: 4 FMAC units, 1 FDIV unit Vector processing unit registers: 128-bit wide, 32 entries Image processing unit: MPEG2 macro block layer decoder Direct memory access: 10 channels V_{DD} Voltage: 1.8 V Power consumption: 15 W at 1.8 V

II. General PS2 Overview

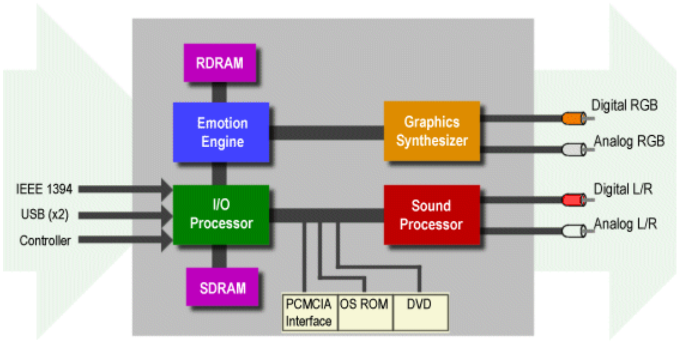


Figure 1. Main Parts of the PS2[1]

From the figure 1, we can see that The I/O processor (IOP) handles all USB, FireWire, and game controller traffic are the main four parts of the device. When we are playing a game on the PS2, the IOP takes our controller input and sends it to the Emotion Engine so that the Emotion Engine is the heart of the PS2, and the part that really makes it unique. The Emotion Engine handles mainly two primary types of calculation and one secondary type: Geometry calculations: transforms, translations, etc. Behavior/World simulation: enemy AI, calculating the friction between two objects, calculating the height of a wave on a pandect. Misc. Function: program control, housekeeping, etc.

When all is said and done, the Emotion Engine's job is to produce display lists (sequences of rendering commands) to send to the Graphics Synthesizer. The Graphics Synthesizer is sort of a soaped-up video accelerator. It can do all the standard video acceleration functions, and its job is to render the display lists that the EE sends it. Finally, the Sound Processor is the "soundcard" of the PS2. It lets we do 3D digital sound using AC-3 and DTS.

The Emotion Engine is sort of a combination CPU and DSP Processor, whose main function is simulating 3D worlds.

II. Basic Architecture

As was expressed above, the Emotion Engine's primary piece of output is the display list. Generating those display lists involves a variety of steps besides simply the plain pure mathematics geometry calculations. For example, if the software we are running is a racing game, then we have got to first calculate the virtual friction between a car's tires and the road once the car turns a corner before we can draw the next scene. Or if the game is an FPS, we have to run the enemy AI's path-finding code so we will know where to place them on each frame. So there is a lot of stuff that goes on behind the scenes and affects the output on the screen. All of these labours are geometry calculations, physics calculations, AI, data transfers, etc. -- is divided up among the following units:

- MIPS III CPU core
- Vector Unit (which is actually two vector units, VU0 and VU1).
- floating-point coprocessor, or FPU
- Image Processing Unit (The IPU is basically an MPEG2 decoder with some other capabilities).
- 10-channel DMA controller
- Graphics Interface unit. (GIF)
- RDRAM interface and I/O interface (for connecting to the two RDRAM banks and the I/O Processor, respectively)

All of the above components are integrated onto one die and are connected (with the exception of the FPU) via a shared 128-bit internal bus.

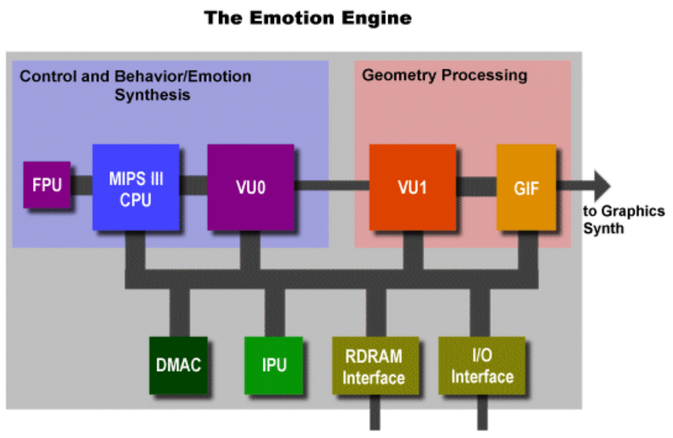


Figure 2. The Emotion Engine Architecture[2]

As we noted in the bullet list, the VU can further into two independent, 128-bit SIMD/VLIW vector processing units, VU0 and VU1. These units, though they're micro architecturally identical, are each intended to fill a specific role. Toshiba, who designed the Emotion Engine and licensed it to Sony, didn't feel that it was optimal to have three pieces of general purpose hardware (a CPU and two vector processors) that could be assigned to any task that was needed. Instead, they fixed the roles of the devices in advance, customized the devices to fit those roles, and organized them into logical units. In that respect, they're sort of like employees who've been grouped together on the basis of talent and assigned to teams. Let's look at the division of labour amongst the components:

- CPU + FPU: basic program control, housekeeping, etc.
- CPU + FPU + VU0: behavior and emotion Synthesizers, physics calculations, etc.
- VU1: simple geometry calculations that produce display lists which are sent directly to the Graphics Synthesizer (via the GIF).
- IPU: image decompression.

A. The CPU/FPU/VU0 team.

The FPU and VU0 are coprocessors for the MIPS III CPU core. This means that the CPU, the FPU, and VU0 all form a logical and functional unit (or a team, if we will) where the CPU is the primary, controlling device and the other two components extend its functionality. This CPU/FPU/VU0 team has a common set of goals: emotion Synthesizers, physics, behavior simulation, etc. .

The other important component that ties the CPU core and VU0 closely together is the Scratch Pad RAM. The SPRAM is 16K of very fast RAM that lives on the CPU, but that both the CPU and VU0 can use to store data structures. The SPRAM also acts as a staging area for data, before it's sent out over the 128b internal bus. So the SPRAM is kind of like a shared workspace, where the CPU and VU0 collaborate on a piece of data before sending it out to wherever it needs to go next.

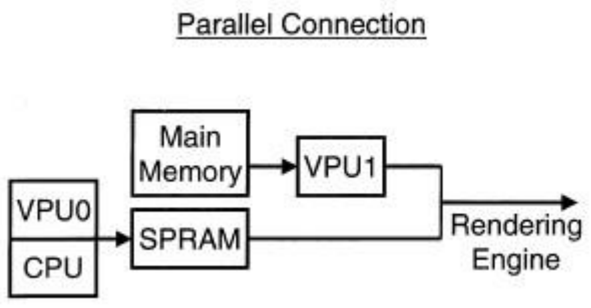
B. The VU1/Graphic Synthesizer team

The other main team is composed of VU1 and the Graphics Synthesize resizer (which communicate via the GIF). Just as VU0 has a dedicated bus to the CPU core, VU1 has its own 128-bit dedicated path to the GIF. However, VU1 and the Graphics Synthesizer aren't as closely tied together as are the CPU/FPU/VU0 group. VU1 and the GS are more like equal partners, and one doesn't control the other. We're probably wondering at this point just who does control VU1. This is an interesting question, and we will discuss the answer when we talk about VU1's architecture in detail.

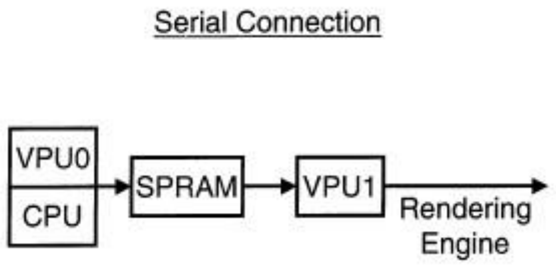
C. *Putting the team to gather*

Though the roles of the components were fixed by the PS2's designers, the overall design is still quite flexible. We divvy up an application's work amongst the teams however we like. For instance, the CPU/FPU/VU0 group can generate display lists and do geometry processing in parallel with VU1, so both groups can send display lists to the GIF at the same time.

Or, the CPU/FPU/VU0 group can act as a sort of pre-processor for VU1. The CPU and co. process conditional



branches in the rendering code and load data from main memory. They then generate world information that VU1 takes as input and turns into a display list. This flexibility allows a developer to customize the process of generating and rendering the 3D environment to suit the needs of the specific application we're working with.



Now that we have gone over the basics of the Emotion Engine's operation, it's time to get hard-core. For the remainder of this article, I'll go in-depth on the MIPS III CPU core, VU0, and VU1. I'll give we the straight scoop on how these components are designed, and how they're integrated with each other. If terms like instruction latency, pipelining, SIMD make wear eyes glaze over, then we might want to check out here. If, however, we're an architecture enthusiast who eats CPU internals for breakfast, then hang on, because what follows is quite fascinating.

D. *The MIPS III CPU core*

The CPU core is a two-way superscalar in-order RISC processor. Based on the MIPS R5900, it implements the MIPS-III instruction set architecture (ISA) and much of MIPS-IV, in addition to a custom instruction set developed by Sony which operated on 128-bit wide groups of either 32-bit, 16-bit, or 8-bit integers in single instruction multiple data (SIMD) fashion (i.e. four 32-bit integers could be added to four others using a single instruction). Instructions defined include: add, subtract, multiply, divide, min/max, shift, logical, leading-zero count, 128-bit load/store and 256-bit to 128-bit funnel shift in addition to some not described by Sony for competitive reasons. Contrary to some misconceptions, these SIMD capabilities did not amount to the processor being "128-bit", as neither the memory addresses nor the integers themselves were 128-bit, only the shared SIMD/integer registers. For comparison, 128-bit wide registers and SIMD instructions had been present in the 32-bit x86 architecture since 1999, with the introduction of SSE. The MIPS-based core consists of two arithmetic logic units (ALUs) and a floating point unit (FPU). The integer units are 64-bit, but the FPU was single-precision, or 32-bit. The custom instruction set was implemented by grouping the two 64-bit integer units. Both the integer and floating-point pipelines are both six stages long. To feed the execution units with instructions and data, there is a 16 KB two-way set associative instruction cache, an 8 KB two-way set associative non-blocking data cache and a 16 KB scratchpad RAM. Both the instruction and data caches are virtually indexed and physically tagged while the scratchpad RAM exists in a separate memory space. A combined 48 double entry instruction and data translation is provided for translating virtual addresses. Branch prediction is achieved by a 64-entry branch target address cache and a branch history table that is integrated into the instruction cache. The branches mispredict penalty is three cycles due to the short six stage pipeline:

Pretty standard RISC stuff. As usual, the execute stage can take a couple of cycles depending on the latencies of the instructions.

The FPU coprocessor, COP1, doesn't really deserve its own section. It's pretty much a straight-up, floating-point coprocessor that's a throwback to the classic RISC days when the FPU was a separate unit from the core. It executes basic, 32-bit MIPS coprocessor instructions using one FMAC unit and one FDIV unit, each of which is the same as the FMACs and FDIVs in the vector units. I'll talk about what these units do when we get to the vector unit discussion.

As we can tell from the above, the CPU core isn't really all that exciting. The only really cool things in it are the SPRAM and the 128-bit integer SIMD capabilities. Other than that, there's not much out of the ordinary going on. This unit is mostly here to handle program control flow by processing branch commands. It also does other stuff, but it doesn't do any of the real heavy lifting--that's reserved for the vector units.

E. *The Vector Unit*

Both VU0 and VU1 are micro architecturally identical, but they are not functionally identical. VU1 has some extra features tacked onto the outside of it that help it do geometry processing, and VU0 has some features that it doesn't normally use. Toshiba did things this way to make the units easier to manufacture. Since VU0 is simpler, we will start with it first.

1) Vector Unit 0

VU0 is a 128-bit SIMD/VLIW design. Since VU0 is a coprocessor for the MIPS III core, it spends most of its time operating in Coprocessor Mode. This means it looks like just another logical pipe (along with the integer ALUs) to the programmer. The instructions that make VU0 go are just 32-bit MIPS COP instructions, mixed in with integer, FPU, and branch instructions. In this respect, VU0 looks a lot like the G4's Altivec unit. Often, in the rendering process, the CPU maintains a separate thread that controls VU0. The CPU places FP data on the dedicated bus in 128b chunks (w,x,y,z), which the VIF unpacks into 4 x 32 words for processing by the FMACs. VU0 has its own set of 32, 128-bit FPRs (floating-point registers), each of which can hold 4, 32-bit single precision floating-point numbers. It also has 16, 16-bit integer registers for integer computation.

Here are the computational units available to VU0 (and VU1):

- 4 FMACs
- 1 FDIV
- 1 LSU
- 1 ALU
- Random number generator.

The first 5 units here, the 4 FMACs and the 1 FDIV, are sort of the heart of both VU0 and VU1 (which are they the heart of the Emotion Engine, which is itself the heart of the PS2). So this is where the magic happens. Each of the FMACs can do the following instructions:

Floating-Point Multiply-Accumulate	1 cycle
Min/Max	1 cycle

The FDIV unit does the following instructions:

Floating-point Divide	7 cycles
Square Root	7 cycles
Inverse Square Root	13 cycles

The bulk of the processing that the PS2 does to make a 3D game involves performing the above operations on lots and lots of data. Now, those last three units in my list (LSU, ALU, and RNG) aren't normally shown in most charts as being part of VU0. I suspect this is because they aren't used in coprocessor mode.

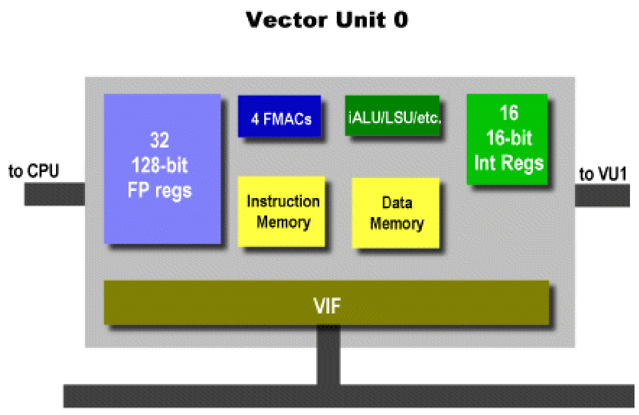


Figure 3. The Vector Unit 0

When VU0 is acting like a MIPS Coprocessor, it only uses the 4 FMACs. "Wait a minute," we're saying, "isn't VU0 always a MIPS coprocessor--we know the 128-bit dedicated bus and stuff? We went to great lengths to make that point in the first half of the article." Yeah, I did kind of insist that VU0 is on the CPU's "team," and that they share the same goals, and that it's bound to the CPU, etc... This *is* kind of misleading but all will become clear in the final section. For now, just understand that VU0 mostly operates as a MIPS Coprocessor that handles any FP SIMD instructions that show up in the CPU's instruction stream.

2) Vector Unit 1

VU1 is a fully independent SIMD/VLIW processor that includes all the architectural features of VU0, plus some additional mojo. These additions relate directly to VU1's role as a geometry processor for the Graphics Synthesizer, and they help bind it more tightly to the GS. The primary addition is an extra functional unit, the Elementary Functional Unit (EFU). The EFU is just 1 FMAC and 1 FDIV, just like the CPU's FPU. The EFU performs some of the more basic calculations required for geometry calculation.

Another big difference between VU1 and VU0 is that VU1 has 16K of data memory and 16K of instruction memory (as opposed to VU0's 8K data/8K instruction sizes). This most of the amount of data memory is needed because VU1's role as a geometry processor requires that it handle much more data than VU0.

Finally, VU1 has multiple paths it can take to get data to the GIF (and on to the GS). Like VU0, VU1 can send display lists to the GIF via the main, 128b bus. Or, VU1's VIF can send data directly to the GIF. Finally, there's a direct connection between VU1's 16K data memory and the GIF, meaning that VU1 can work on a display list in data memory, and the DMAC can transfer the results directly to the GIF.

F

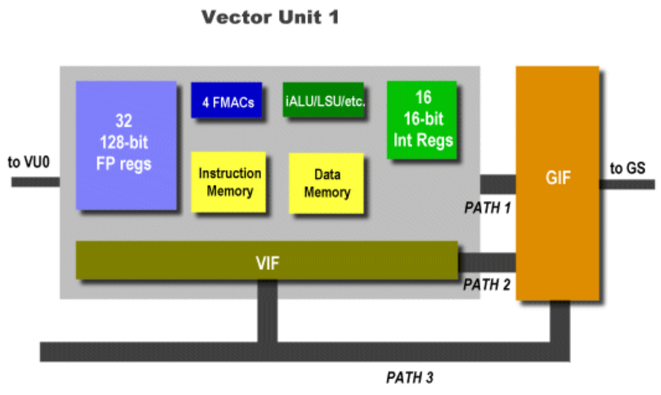


Figure 4. Vector Unit 1[4]

As we will remember from the discussion of VU0, VU0 is controlled by the CPU, and VU0 gets its instructions from whatever program the CPU is currently running. VU1, however, doesn't work that way. VU1's VIF plays a much more noticeable role in VU1's life than VU0's VIF does in its. VU1's VIF takes in and analyses what Sony confusingly calls a 3D display list. This 3D display list is not VU1's program. Rather, it's a data structure that contains two types of information, and some particular commands that tell VU1 how to handle this information. The two types of info are

a. The actual VU1 program instruction, which on in VU1's instruction memory.

b. The data that said program operates on. This goes in VU1's data memory.

The VIF decodes and parses the 3D display list commands, and makes sure that VU1 program code and data find their way into the correct spots. In this manner, VU1 can operate independently of the CPU to generate display lists. Executing these VU1, "VLIW mode" programs brings into play those three units that VU0 often neglects: the LSU, the iALU, and the RNG. There three units, along with the EFU (which acts as a general FPU), all function to make VU1 a full-blown SIMD/VILW coprocessor.

F. *Image Processing Unit (IPU)*

The IPU allowed MPEG-2 compressed image decoding, allowing playback of DVDs and game FMV. It also allowed vector quantization for 2D graphics data.

G. *DMA, DRAM and Memory Management Unit (MMU)*

The memory management unit, RDRAM controller and DMA controller handle memory access within the system.

H. *Internal data bus*

Communications between the MIPS core, the two VPUs, GIF, memory controller and other units is handled by a 128-bit wide internal data bus running at half the clock frequency of the Emotion Engine but, to offer greater bandwidth, there is also a 128-bit dedicated path between the CPU and VPU0 and a 128-bit dedicated path between VPU1 and GIF. At 150 MHz, the internal data bus provides a maximum theoretical bandwidth of 2.4 GB/s.

I. *External Interface*

Communication between the Emotion Engine and RAM occurs through two channels of DRDRAM (Direct Rambus Dynamic Random Access Memory) and the memory controller, which interfaces to the internal data bus. Each channel is 16 bits wide and operates at 400 MHz DDR (Double Data Rate). Combined, the two channels of DRDRAM have a maximum theoretical bandwidth of 25.6 Gbit/s (3.2 GB/s), about 33% more bandwidth than the internal data bus. Because of this, the memory controller buffers data sent from the DRDRAM channels so the extra bandwidth can be utilised by the CPU. The Emotion Engine interfaces directly to the Graphics Synthesize resizer via the GIF with a dedicated 64-bit, 150 MHz bus that has a maximum theoretical bandwidth of 1.2 GB/s. To provide communications between the Emotion Engine and the Input Output Processor (IOP), the input output interface interfaces a 32-bit wide, 37.5 MHz input output bus with a maximum theoretical bandwidth of 150 MB/s to the internal data bus. This interface provides vastly more bandwidth than what is required by the PlayStation's input output devices.

III. Programming The VU

To covering up, we are going to take a look at the VU's instruction format, and talk about its microarchitecture in a bit more detail. We will look at VU1 first, because it always runs in "VLIW mode." Then we will talk about VU0, and how it's different.

Both VU1 and VU0 are 2-issue, VLIW processors. The basic instruction format for VU1 is a 64-bit, VLIW instruction packet (or "bundle," or whatever VLIW term we want to use) that can be divided into two, 32-bit COP2 instructions.

These two instructions are executed in parallel by two execution units: the upper execution unit and the lower execution unit. (Refer back to the two VU block diagrams on the last page. The upper unit is blue and the lower unit is green.) These two units have the following functionality:

Upper instructions	Lower instructions
• 4 parallel FP ADD/SUB • 4 parallel FP MUL	• FP DIV/SQRT/RSQRT • Load/Store 128b

• 4 parallel FP ADD/MSUB • 4 parallel MAX/MIN • Outer product calculation • Clipping detection	data • EFU (1 FMAC + 1 FDIV) • Jump/Branch • Random number generator/misc.

Now, what we should note is that the upper execution unit is a SIMD unit, while the lower is not hence the term "VLIW/SIMD." So the code is "64-bit VLIW," of which 32 bits are SIMD. Cool, eh? I thought so.

To see this in action, let's look at a code example which is revised from one of Sony's slides.

Upper Instruction	Lower Instruction
MUL VF04, VF03, Q	DIV Q, 1.0, VF02.w
MUL ACC, VF10, VF01.x	MOVE VF03, VF02
MADD ACC, VF11, VF01.y	ADD VI03M VI03, -1
MADD ACC, VF12, VF01.z	NOP
MADD VF02, VF13, VF01.w	LQ VF01, VI01++
NOP	BGTZ VI03, LOOP
NOP	SQ VF04, VI02++

The instructions on the left are executed by the upper execution unit, while the ones on the right are executed by the lower unit.

VU0 is a bit different from VU1 in that, instead of operating in VLIW mode all the time, it normally runs in "MIPS Coprocessor Mode." A MIPS Coprocessor instruction is a 32b instruction and not a 64-bit VLIW instruction. So this means that when it's in COP mode, VU0 can crunch 4, 32-bit FP SIMD numbers in parallel, using the 4 FMACs in the upper execution unit. (Here it is assumed that in this situation, the upper opcode contains the SIMD FP instruction op and the lower opcode a NOP.)

VU0 doesn't have to stay in COP mode though. It can operate in VLIW mode by calling a micro-subroutine of VLIW code. In this case, it takes a 64-bit instruction bundle and splits it into two 32-bit MIPS COP2 instructions, and executes them in parallel, just like VU1.

As we can see, having two operating modes for VU0 is a bit complex, but it gives the unit a lot of flexibility.

VI. Conclusion

As we can see from the above analysis, with a total of 10 FMACs, 4 FDIVs, and all the other integer, branch, and load/store resources available, the Emotion Engine is a boss.

VII. References

1. Hennessy, John L.; Patterson, David A. (29 May 2002). Computer Architecture: A Quantitative Approach (3 ed.). Morgan Kaufmann.ISBN 978-0-08-050252-6. Retrieved 9 April 2013.
2. Diefendorff, Keith (19 April 1999). "Sony's Emotionally Charged Chip". Microprocessor Report (Microdesign Resources)
3. K. Kutaragi et al., "A Micro Processor with a 128b CPU, 10 Floating-Point MACs, 4 Floating-Point Dividers, and an MPEG2 Decoder," ISSCC (Int'l Solid-State Circuits Conf.) Digest of Tech. Papers,Feb. 1999, pp. 256-257.
4. F.M. Raam et al., "A High Bandwidth Superscalar Microprocessor for Multimedia Applications," ISSCC Digest of Technical Papers,Feb. 1999, pp. 258-259.
5. A. Kunimatsu et al., 5.5 GFLOPS Vector Units for "Emotion Synthesizeresis", (Slide show and presentation.) System ULSI Engineering Laboratory, TOSHIBA Corp. and Sony Computer Entertainment Inc.
6. Masaaki Oka Masakazu Suzuoki. Designing and Programming the Emotion Engine, Sony Computer Entertainment. IEEE Micro, pp. 20-28
7. Berkeley Design Technology, Inc. DSP Processors and Cores -- the Options Multiply. Reprinted from Integrated System Design, June, 1995.