Reed Electronics Group

MICROPROCESSOR REPORT

www.MPRonline.com

THE INSIDER'S GUIDE TO MICROPROCESSOR HARDWARE

CELL MOVES INTO THE LIMELIGHT

ISSCC Begins the Rollout of Cell Architecture

By Kevin Krewell {2/14/05-01}

Cell is real. Cell is out—finally! IBM has been eager to reveal more about a project at which it has been hard at work for almost five years. The Cell processor has been shrouded in secrecy, necessitated by the competitive nature of the multibillion-dollar game console market. With the first production units still at least a year away, Sony, Toshiba, and IBM now feel comfortable to begin revealing the nature of the processor at the 2005 International Solid-State Circuit Conference (ISSCC) but there was little information on the system design. What the announcement at ISSCC did not lack was publicity—the presentations got extensive national and international press coverage. Some of the stories hyped the Cell processor as competition for Intel, but this chip is designed for Sony's next-generation gaming console, not PCs (although there is the open question of would Apple use Cell). The chip designers did tweak Intel by showing that the prototype chips can operate at clock frequencies up to 4.6GHz (Intel had the very public problems reaching 4.0GHz with the Pentium 4). We've made a conscious effort to ignore the hype and focus on the technical aspects of the chip.

Early concepts of the Cell processor have already been glimpsed in patent reports (see *MPR 1/03/05-01*, "New Patent Reveals Cell Secrets"), but, for the most part, the details of the architecture's real implementation were hidden until the ISSCC abstract was released in December. (It was somewhat like trying to predict what a new production car would look like on the basis of the concept car.) At the 2005 ISSCC, the design began to come out, and we can reveal more details about why we chose the Cell Processor for the *MPR* Analysts' Choice Award for Best Technology of 2004. (See *MPR 1/31/05-01*, "Chips, Software, and Systems.")

The Cell development program has involved people located around the world, with a large core team of designers in an Austin, Texas, facility called the STI Design Center. The team had some very lofty goals and a tight schedule to execute against. It started in mid-2000, when IBM, Sony, and Toshiba began exploring the idea of working together. Toshiba had been Sony's partner in the PlayStation 2 chip set and brought its experience in large-scale integration and volume manufacturing. By fall 2000, the three partners had refined the concepts, and by March 2001, the contracts for the design center were signed and work began to accelerate.

Cell is a mix of ideas from the partners that has evolved since the initial patent application, referenced in a previous story, was submitted. While all partners contributed and are involved with the management of the program, IBM will be able to sell the Cell processor to others. So we could see devices using Cell coming from companies beyond IBM, Sony, and Toshiba.

The overarching goal was to create a new architecture that could process the next generation of broadband media and graphics with greater efficiency than the traditional approaches of ultradeep pipelines and the ganging of numerous complex and power-inefficient out-of-order RISC or CISC cores. The designers took a clean-slate approach to the problem but ultimately based the main core on the familiar Power architecture. Using Power gave them a well-known base to extend upon and could jump-start software development. In addition to the Power core, the design includes eight other processing cores called the Synergistic Processor Element (SPE), shown in Figure 1.

The Cell processor is technically a family of processors compliant to the specifications of the Broadband Processor Architecture (BPA), the new architecture designed to process media data. Future implementations could have differing numbers of Power and Synergistic Processor cores. (IBM loves three-letter acronyms, and the company spread plenty of love on the Cell processor—or should I say the BPA design.) In designing the BPA, IBM looked at different workloads in areas of cryptography, graphics transform and lighting, physics, fast-Fourier transforms, matrix math, and other more scientific workloads.

BPA (Cell) design features include the following:

- Extension of the Power Architecture
- Coherent and cooperative off-load processing
- Enhanced SIMD architecture
- Power efficiency improved over that of conventional architectures
- Linux port derived from work on PowerPC
- Resource allocation management
- Locking caches (via replacement management tables)
- Multiple memory page table sizes
- Isolation mechanism for secure code execution

Getting Inside the Cell Processor

Fundamentally, the Cell processor consists of three main units supported by two Rambus interfaces. There is a single Power processor element (PPE) that acts as the main host processor, eight single-instruction, multiple-datastream (SIMD) processors, and a highly programmable DMA controller (see Figure 1). Cell has eight SIMD processing elements and the Power core, so Cell can also be considered a multiple instruction stream, multiple datastream (MIMD) processor. Although SIMD and vector processors have been built in the past, and none too successfully, IBM believes this is the best architecture to process a great deal of media and graphics data with a minimum of power and a reasonable die size. Programming techniques for a processor such as Cell will take some time to develop, but IBM hopes that starting with a Power core and building on that foundation will speed things along.

The general architecture seems in parts inspired by Sony's experience with the PlayStation 2 processor, IBM's experience with network processors, and a number of vector computers dating back to the mid-1970's. In the '70's and 80's a number of computer systems were built that attached a series of processing functions around a ring network, including the Programmable Signal Processor (PSP) for the Air Force JSTARs program, which was inspired by the CDC Advanced Flexible Processor (AFP). The Project MAC data flow processor developed at M.I.T. in the 1970's used a packet communications architecture that passed a bundle of an instruction and two operands called an "Instruction Cell" to an array of processing elements through a routing network. The complexity of these system designs also meant that programming them was equally complex. So while the Cell processor design is unique today for a proposed mainstream processor, its design is built on research and development that is decades old.

The goal of building a chip with a large amount of parallel processing, and still being able to offer deterministic processing times, meant that some conventional microprocessor concepts had to be discarded. That included out-of-order execution by the Power core, and local store memory for the SPEs do not use hardware cache-coherency snooping protocols avoiding the indeterminate nature of cache misses.

The Cell processor supports Rambus's XDR interface for memory and the Redwood interface for I/O (see *MPR 8/04/03-02*,"Rambus Yellowstone Becomes XDR") and multiprocessor link. Cell doesn't integrate networking, peripherals, or large memory arrays (unlike the IBM BlueGene/L processor) on chip. The reason for this design difference between Cell and the IBM BlueGene chip is that BlueGene could use an older process technology that had embedded DRAM (eDRAM) available; it worked for BlueGene because the goal was to add many thousands of modestly clocked processors (700MHz) into one system to build a powerful supercomputer system. Each Cell processor chip needs to have greater performance, because each system is expected to contain only one or a very few chips. The BPA architecture was designed with certain performance and die-size goals. To reach the price points for the Cell processor, IBM had to consider die size.

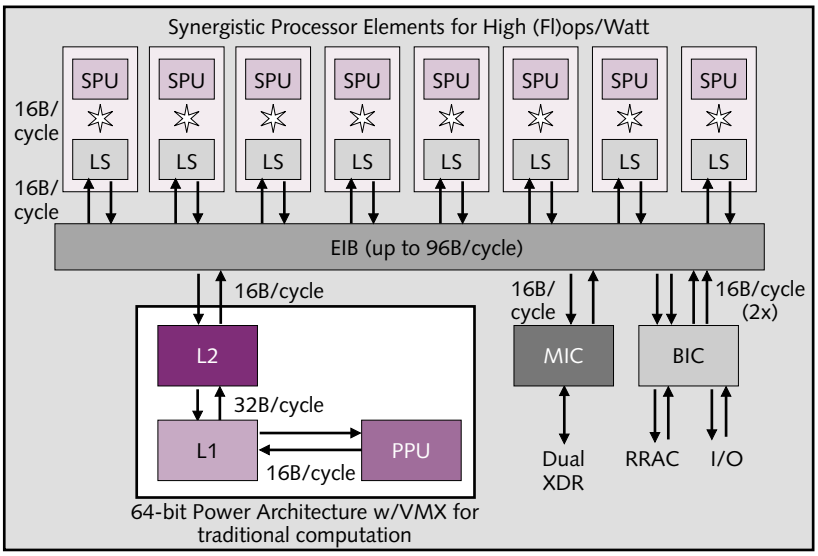

Figure 1. Block diagram of the Cell processor. The synergistic processor element (SPE) is the combination of the synergistic processor unit (SPU) with its local store (LS) memory and memory flow controller (MFC). The Power processor unit and the L2 cache form the Power processor element. The element interconnect bus (EIB) connects the SPE and PPE blocks together. The Cell processor uses I/O and memory interfaces licensed from Rambus.

The die size of the Cell processor is actually slightly smaller than that of the original Emotion Engine in the PlayStation 2 (240mm^2 in a 0.25-micron process). To reach both the right die size and performance, IBM also needed to use its 90nm SOI technology, which at present doesn't support eDRAM. Placing the I/O off chip gave the designers more flexibility with system design. A game console contains different peripherals than do a TV or supercomputer. For Sony's next-generation gaming system, the peripheral chip will likely be Sony proprietary.

The Cell processor can reach its performance goals by using high clock speeds and can be supported by using the high-performance XDR DRAM interface licensed from Rambus. The on-chip memory controller is combined with the high-bandwidth XDR interface, which is capable of 25.6GB/s of bandwidth and helps to compensate for the modest 512K of on-chip L2 cache (see Figure 2).

Power Processor Element Provides Control

The "brain" of the Cell chip is the Power Architecture core. If the Cell processor were a network processor, the Power core would be the control-plane processor. This is a new 64-bit in-order, two-issue superscalar Power core design, optimized for the Cell processor, and not a recycled core from another processor. The Power core combined with the L2 cache is called the Power processor element or PPE core in IBM documentation. The PPE is called the PU (processor unit) in the original Cell patent. This implementation of the PPE includes the Power with AltiVec (which IBM calls VMX) instruction-set extensions. The Power core also has support for simultaneous multithreading, with up to two threads.

It was noted earlier that Sony and Toshiba do not have exclusive rights to Cell. While IBM cannot comment on our speculation, it would seem possible that the Cell processor could be used by Apple Computer in a Power Mac G-series computer (as a G6 perhaps). The Power processor core in Cell has some significant architectural differences from the PowerPC 970FX, but it's not clear whether those differences would preclude running Apple-compatible software on the Cell processor. With the larger power envelope of the G5 computer case, the Cell processor could even be clocked at higher frequencies than in a home-entertainment console. Imagine what Apple could do to the graphics user interface and media capabilities of the Mac with this much processing-power capability!

Although the team did look at other processor cores, using Power was the obvious solution for IBM because of the company's investment in the architecture and experience with it. Development-time constraints were also a factor in the choice. But the team didn't just take an existing core like the PowerPC 970FX and build an SoC around it. The core for Cell is new and appears to have been designed before the clock-frequency-is-dead era. The core was designed to reach certain power and die-size goals and is designed to be able to run at clock frequencies in the 4+GHz range (see Figure 3). The engineering team did simplify some of the core design (for example, it's an in-order design and only a dual-issue superscalar) and used some dynamic logic in the design in certain critical timing areas.

The core complies with the PowerPC instruction-set architecture version 2.02 (and the 2.01 public version of the specification). The core was designed with a particular balance of die size, clock speed, and architectural efficiency that is different from that of the PowerPC 970. (See *MPR 10/28/02-02*, "IBM Trims Power4, Adds AltiVec.") This instantiation of the Power Architecture still has a relatively long pipeline, much like the Power 4 and PowerPC 970, but the Cell design does not have a very wide issue pipeline or out-of-order execution, nor does it have as many functional units. The Cell Power core has hardware fine-grain multithreading. The multithreading design supports fine-grained multithreading with round-robin thread scheduling. If both threads are active, the processor will fetch an instruction from

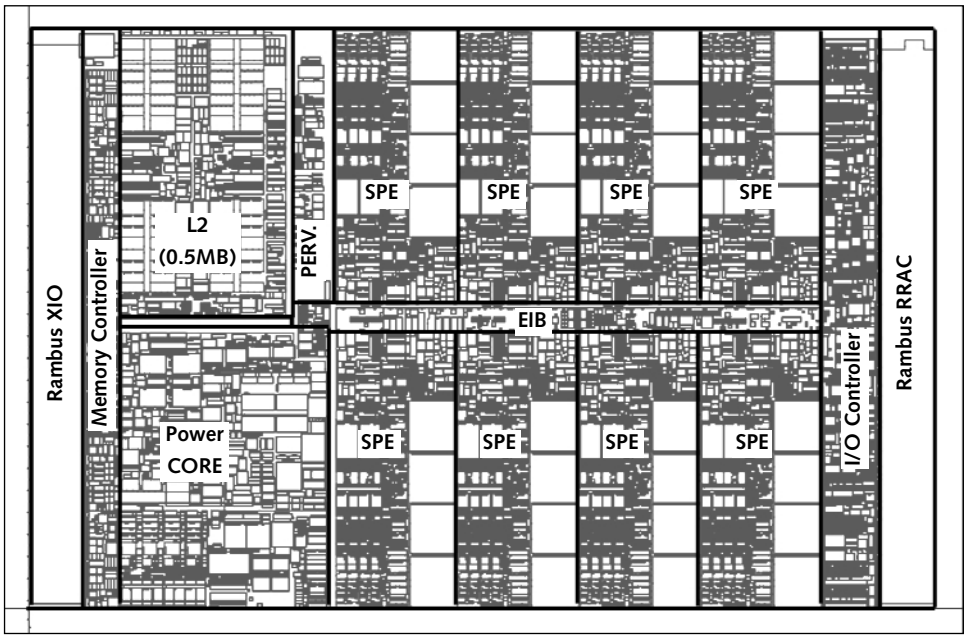

Figure 2. The layout of the Cell processor places the Rambus XDR DRAM memory and Redwood I/O interfaces at each side of the die. The processor cores in the chip surround the EIC internal bus. In IBM's 90nm SOI process, the Cell processor production die has 235 million transistors and is 235mm^2. The prototype processor presented at ISSCC is slightly smaller at 221mm^2.

each thread in turn. When one thread cannot issue a new instruction or is not active, the other active thread will be allowed to issue an instruction every cycle. Threading does add some burden to the die size (around 7% in this case), as there must be duplicated register files, program counters, and parallel instruction buffers (before the decode stage). The benefit of threading is widely known, and with broadband and gaming content, multiple concurrent threads should be easy to extract from software. A mispredicted branch in the Power core has an eight-cycle penalty, and a load has a four-cycle data-cache access time. When one thread is stalled, possibly by a mispredicted branch or a cache miss, the second thread often can execute to fill the execution stall. This leads to greater architectural efficiency and higher utilization of processor resources.

The Power core contains a conventional 32K two-way instruction cache and a 32K four-way set-associative data cache. The design is an in-order execution pipeline, and there is also a one-cycle cache-use penalty.

Although the ISSCC presentations talked about 4.6GHz operation, don't expect the final product to run at that speed. To fit within the space, power, and noise constraints for a game console, IBM will likely run the processor at lower voltages and frequencies. The clock frequency should still be in the 4GHz range, as the processor core will still be quite busy running processing overhead and the operating system(s), despite having eight SPEs to accelerate media processing. At this point, the final core frequency has not been finalized.

One side note is that the BPA/Cell architecture is big-endian. The Power architecture supports both big- and little-endian data formats, but with Cell, little-endian support was jettisoned. For the PlayStation implementation, it might have been considered extraneous. As a self-contained system, with dedicated software, the designers didn't feel compelled to offer interoperability with PC software (which is little-endian). The byte ordering of Apple Computer's software is also big-endian.

Interrupt management is similar to that of standard PowerPC. Interrupts arising from SPEU and memory flow controller (MFC) events in the Cell processor are treated as external interrupts to the PPE. The PPE is also capable of running multiple operating systems through a hypervisor (low-level software below the operating system for virtual processing support).

SPE Provides the Brawn

If the PPE is the brain of the processor, the Synergistic Processing Elements (SPE) are the muscles that do the heavy lifting. In the network-processor analogy for the Cell processor, the SPEs are the data-plane processors. In fact, with a little change to the SPE architecture and some integrated packet parsing, Cell would make a very interesting network processor. The SPE includes the Synergistic eXecution Unit (SXU) and 256KB of local-store (LS) SRAM. Each of the eight SPEs has its own private local store, and the local-store memory is aliased to main memory but does not participate in a cache-coherency protocol. Software must manage the movement of data and instructions in and out of the LS and is controlled by the MFC. The LS has data-synchronization facilities but does not participate in hardware cache coherency. The eight local stores do have an alias in the memory map of the processor, and a PPE can load or store from a memory location that is mapped to the local store (but it's not a high-performance option). Similarly, another SPE can use the DMA controller to move data to an address range that is mapped onto a local store of another SPE or even to itself. LS memory, if cached in the system, is not coherent with respect to modifications made by the SPE to which it belongs.

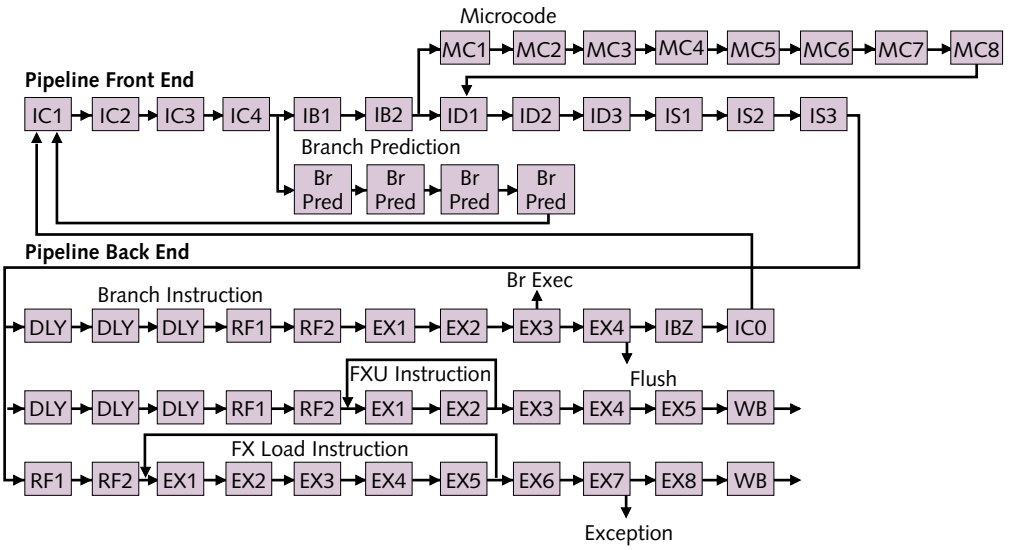

Figure 3. Pipeline of the Power Architecture core of the Cell processor is 21 clock cycles long and was designed with 11 F04 delays between clock periods.

The eight SIMD units on Cell are identical and can process both integer and floating-point numbers. The SPE can handle 8-, 16-, or 32-bit integer and single-precision (32-bit) or double-precision (64-bit) data formats. These SPEs provide a coherent off-load engine for the PPE. As such, the SPE is more independent than what is typically considered a coprocessor. And the eight SPEs are not directly tied to the PPE core. They take command streams from the memory

allocated to them, and the movement of commands and data are controlled by the MFC.

Each Synergistic Processor Unit (SPU) is a four-way SIMD unit optimized for single-precision (32-bit) floating point but can support double-precision math at reduced performance (about ten times slower). Figure 4 shows the pipeline of the SPU. It has a rich set of 128 registers that are 128 bits wide and are low latency. The large number of registers is purposeful and was added to hold more data values closer to the SIMD unit and reduce the need for LS accesses. The instruction set of the SPU can be said to have been "inspired" by the VMX/AltiVec instruction extensions and are a superset of the vector processor unit instruction set in the Emotion Engine processor for PlayStation 2. The SPE instruction set (see Figure 5) supports a multiply-add operation with three sources and one destination. At a 4GHz clock frequency, the eight SPEs would be capable of a theoretical peak single-precision floating-point performance of *256MFLOPS*. For one chip, that is quite an impressive number, but is short of some expectations for the chip.

Latencies	
Simple Fixed (FX)	2
Shift (FX3)	4
Single Precision (FP1-6)	6
Floating Integer (FP7)	7
Byte	4
Permute	4
Local Store (LS)	6
Channel (not shown)	6
Branch (not shown)	4

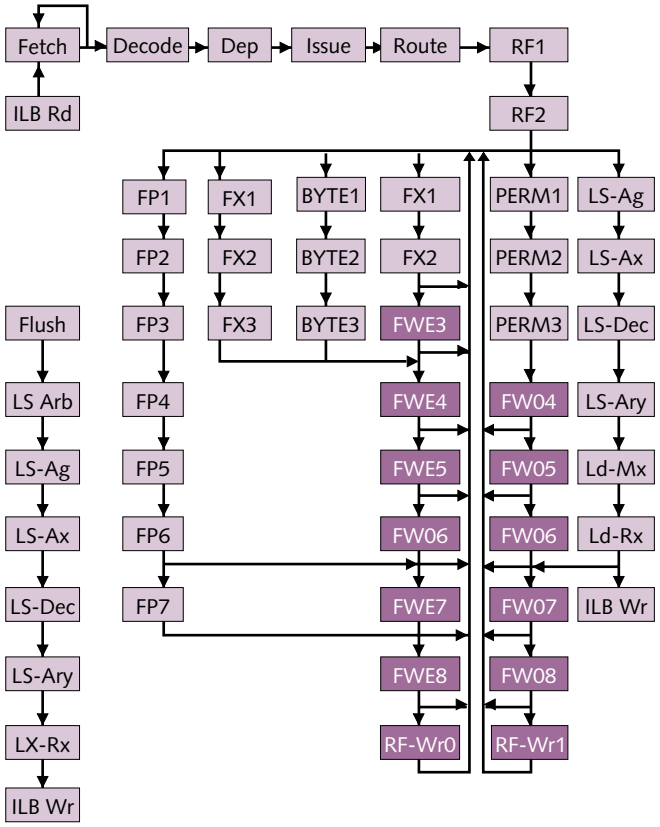

Figure 4. The SPU pipeline diagram shows that the SPU has a relatively short pipeline but can still keep pace with the Power core clock speeds and was designed to have the same level of gates per cycle as the Power core.

IBM engineers admitted that they considered a very long instruction word (VLIW) architecture for the SPE (somewhat like the Trimedia TM32). However, they considered VLIW code expansion a problem, and their existing experience with the VMX instruction extensions showed that media operations could use SIMD operations very effectively. Once the media data is loaded into the local store, the SIMD units can be very efficient by processing multiple data words at the same time. The issue with SIMD units added onto conventional processors is the overhead of gathering operands into the registers. With the SPUs, the MDF can gather the data into the local store.

The SPEs were not multithreaded for a couple of good reasons. The one stated by IBM was that the goal of the design was to provide the SPU with enough memory and registers for all the required data to be processed to be present without a having a miss penalty. The second reason we deduce is that multithreading the SPUs would have complicated the issue of scheduling and isolating operations from each other and significantly increased the die area (times eight). In programming an SPE operation, the programmer can be assured that all the resources are available to the task and are not shared with another thread. Sharing an SPE would have made isolating a critical process, such as data encryption/decryption, from another process task it shared with the same local store. At the same time, the SPE can double buffer tasks. The MFC can begin to transfer the data set of the next task while the present task is running (within the bounds of the bandwidth of the local store).

The floating-point operation is presently geared for throughput of media and 3D objects. That means, somewhat like AMD's 3DNow SIMD extensions, that IEEE correctness is sacrificed for speed and simplicity. Especially with these workloads, exact rounding modes and exceptions are largely unimportant. For overflows and underflows, saturation results are desirable, rather than a precise exception or undetermined results. A small display glitch in one display frame is tolerable; missing objects, tearing video, or

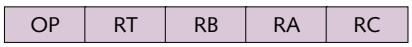

Figure 5. The instruction format of the SPE supports three source operands and one destination.

incomplete rendering due to long exception handling is objectionable.

On the other hand, the double-precision calculations, while considerably slower on the Cell processor, do include more complete support for IEEE 754 rounding modes and exceptions. It doesn't, however, support IEEE 754 precise mode. This instantiation of Cell was not optimized for DP math, as most of the target workloads can use SP, so for a supercomputer application, the SPE might need further development. The Power core has the VMX (AltiVec) SIMD extensions available for additional processing.

The 256KB local storage is a good size, but there were some design compromises. The memory is only single ported and can be accessed only on aligned quadword (128-byte) transfers. The DMA transfers support 1,024-bit transfers with quadword enables. Any DMA transfer less than a quadword is performed by a read-modify-write transaction. Although the local store is single ported (to save area and power), it does support both a wide 128-byte access and a narrow 16-byte access. DMA commands are aggregated into a 128-byte latch and then written in a single local store cycle; similarly, DMA reads occupy only a single cycle for 128 bytes. Instruction fetches also (pre) fetch 128 bytes (64-byte aligned to guarantee at least 17 instructions on a sequential path). Thus, the number of cycles that the local store is unavailable for loads and stores is minimized. IBM expects to see very high utilizations of the local store on optimized code (often near 85–90%).

The access to the local store is prioritized, with the highest priority being DMA transfers of PPE loads and stores. The secondary priority is SPE loads and stores, and an SPE instruction prefetch gets the lowest priority. With the large local store and a relatively long branch-mispredict penalty in the SPE (18 cycles), software programmers will be encouraged to do loop code unrolling. The SPE does have some help for branches that include branch hint instructions, a three-source bitwise select. The SPE defaults to not-taken branch prediction without a penalty, and no penalty is imposed for branches taken and hinted correctly.

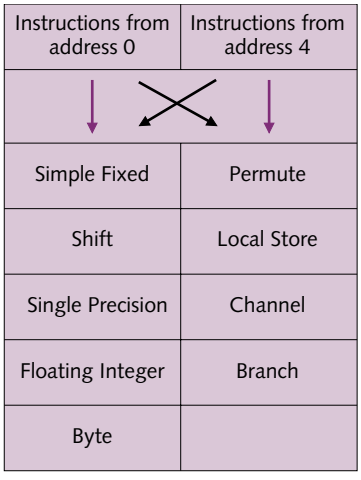

Figure 6. The SPE is capable of limited dual-issue operation. If the instruction is not properly aligned, the instruction swap operation will force single-issue operation.

The architecture allows the local storage of the SPE to be mapped into the real address space of the memory system. Privileged software will control the mapping of the memory. Memory mapping in Cell is a two-stage process. First, the effective address is converted to a virtual address, using the segment lookaside buffer (SLB). Then, the virtual address is converted to the real address using the translation lookaside buffer (TLB). In the PowerPC architecture, privileged software manages the SLB, and hardware manages the TLB. The Cell memory management is a superset of the PowerPC. In the BPE, privileged software manages the SLB and there is an option to manage the TLB either by the hardware or by the privileged software. The latter option offers greater page table flexibility, but at the cost of more software overhead. The Cell Power core's implementation of the MMU is still consistent with the Power architecture.

The SPE is capable of limited dual-instruction issue when an integer or floating-point operation is even-word aligned and a load instruction is odd-word aligned (see Figure 6). In this regard, it is reminiscent of the original Pentium processor and the Intel i860 microprocessor. In this way, there is still a hint of LIW to the SPE.

The SPU units are designed for aggregation into an array of processing elements. The programming model for the SPU has not been predetermined, and the BPA can support both process pipelining and parallel processing. In a pipeline implementation, a transformation on a data set performs a specific task at each SPU and then passes the intermediate results to the next SPU. The advantage of this technique is that the code in each SPU is usually quite small and the operations are easy to manage. The pipeline is also more predictable. This predictability is important to the design goals of the Cell processor and is one of the reasons the SPEs have local memory and not cache memory. The disadvantage is that it's often difficult to make each stage equal in complexity and time. The weak-link-of-a-chain effect is that the overall throughput of the pipeline is limited by the time taken by the slowest stage. Faster stages will often be stalled.

The parallel implementation is more flexible, but completion times are more stochastic. Each program routine is expected to run to completion on an SPE. This has the benefit of better data locality, less data copying, and overall better throughput. The problems include the need for more thread overhead and management as well as data coherency management. With so many SPU units, it is also possible to program with a mix of both methods. The processing of media data is different than general-purpose computing. The data structures have more parallelism. In addition, IBM found there are shorter distances between dependencies.

Either way, the memory model of the Cell processor supports the sharing of memory locations, and the effective

address of the local store may be aliased among different processing units. And it is highly recommended that each SPU be assigned a task that is allowed to run to completion of the task, because a context switch can be costly in both time and storage space, due to large number of wide registers and memory translation buffers. The programming goal should be that the SPE has all the data and instructions to complete its assigned task.

The communications between the MFC and the SPEs is through an SPE channel. Each channel is a unidirectional queue of varying depth and can be configured as blocking or nonblocking. SPU commands must be executed by the SPU in the order received. Through the channel and an external event controller, the SPU can be directed to perform an action or can be set to a pending state. Short communications between the SPUs and the PPE can be accomplished using two mailbox queues provided by the MFC. These 32-bit mailbox messages are user defined. In the PPE, the mailbox is in memory mapped I/O space.

While the SPE is quite capable of processing various data types, handling loads and stores, and branching, it was not designed to run an operating system. This simplifies its design compared with that of a multiprocessor, such as the Cavium Octeon with sixteen 64-bit MIPS processors. The SPU has been designed for high clock speeds, as was the PPE. The use of an in-order design did simplify logic design and dependency checking, but a branch mispredict costs a painful 18 cycles of latency. IBM also mixed in some dynamic logic in the quest for speed and lower power. This is another example of a growing trend of mixing more dynamic logic into designs that need speed and low power.

Security Built In

Instead of the addition of a dedicated security processor, the SPUs can be used to perform security algorithms. Details of how the SPE can be used for security processing have not been released.

One unique design feature of Cell is the ability to fence off SPE processing units from each other through hardware protection features. This way, SPUs that are dedicated to security processing can be isolated from the rest of the system and have special reserved and protected memory that cannot be accessed by another process. This feature will be essential in future digital rights management (DRM) implementations, where the SPU creates a trusted environment.

Memory Flow Controller Moves the Data

To continue the biology analogy further, the memory flow controller (MFC) is the heart and the Element Interconnect Bus (EIB) is the vascular system, keeping the components fed with digital lifeblood of data. To keep the Cell processor running, the MFC can support more than 128 outstanding requests to memory. Keeping the SPE unit utilization high requires the MFC to support many transaction flows simultaneously.

The MFC has its own memory management unit that is a subset of the Power core's MMU. The MFC, like the Power core, supports 64-bit virtual address space. The flow controller's MMU supports the same page sizes as the Power core and includes the new 64K and 16M page sizes. Transfer sizes can range from one byte up to 16KB, but transfers of less than one cache-line size (128B) are discouraged. The controller supports scatter gather and interleaved operations.

IBM's experience with a DMA-list command scheme (where the list of DMA commands is placed in the local store and processed asynchronously by the DMA unit), showed that programs that would have benefited from strided memory addressing and prefetch become memory-bandwidth bound before they become SPE-compute bound. The DMA command-list processing used by the MFC can be considered similar to display-list processing used in graphics.

The MFC transfers data to and from the SPE and memory using get and put commands. Each command can have an instruction modifier (an "s" prefix) that instructs the SPC to begin processing instructions from the next program counter register after the data transfer is complete. The MFC can also take the data from the SPE and load it directly into the Power core L2 cache as well. This is a way to get critical data to the PPE more quickly.

The Memory and I/O by Rambus

The memory and I/O channels are based on licensed Rambus technology. The XDR controller actually consists of two independent controllers, offering more flexibility than one. The memory interface can support an incredible 25.6GB/s of bandwidth. To meet performance goals for the Cell processor, the design requires a tremendous amount of memory and I/O bandwidth. Otherwise the Cell architecture will be stalled by memory or slowed by an inability to feed the graphics buffer fast enough.

Rambus presented a paper at ISSCC on the RRAC. It supports up to 6.4GB/s per RX or TX byte. The Cell processor has five RX bytes and seven TX bytes. The I/O RAC can also be used to connect two Cell processors, using dual unidirectional channels. The design is optimized for one- and two-way systems; for more than two Cell processors to be clustered together, an outside hub is required.

EIB Ties It All Together

To support this many units and the requirements for bandwidth, IBM made a design decision that traded off lowest latency for greater concurrency and better bandwidth. The center of the chip layout shows the element interconnect bus (EIB), which is not actually one bus—it is a data-ring structure with a control bus. Each ring is 16 bytes wide and runs at half of the core clock frequency, so some IBM graphics call it an 8-byte interface (running at full core frequency).

There are four unidirectional rings, but two rings run counter to the direction of the other two. In this way, the worst-case maximum latency is only half the distance of the

ring, not the complete ring. The ring network is able to support up to three simultaneous transfers when transactions are between neighbors on the bus. Because ring transactions are deleted by the destination node, multiple neighbor transactions can occur simultaneously. Resource allocation is provided by a token exchange mechanism, with varying allowable transfer rates, depending on class. The EIB bus manages the token transactions.

Power Under Control

Many of the design decisions were driven by power conservation. The Power core and SPE designs could have added more speculative operations, but those can waste power when the speculative result is not used. While the design doesn't offer a large number of power-management modes, it is capable of clocking at one-eighth the normal speed when idling.

Multiple power-management states are available to privileged software in the BPA specification. The five major states are active, slow, pause, state retained and isolated (SRI), and state lost and isolated (SLI). Each is progressively more aggressive in saving power. Software controls the transitions, but it can be linked to external events. The slow and pause states have numerical modifiers that can represent varying levels of aggressive power management. The SRI state maintains component state information, whereas SLI does not. In SLI, the device is effectively shut off from the system.

While one of the ISSCC papers mentions clock speeds on the order of 4.6GHz, these are unlikely to be the system goals of the Sony game system. A lower clock frequency, with an associated lower core voltage, will be needed to get the Cell processor into a relatively small game-console enclosure.

Programmed for Success?

The processing capability of the Cell processor is still being explored. One demo that IBM has showed was a detailed 3D contour map with satellite images imposed on the geography. The Cell processor can render the ray-cast graphics at around an order of magnitude faster than a contemporary PC processor. The performance of only the 4GHz Power core without the SPEs is harder to compare with the slower 2.5GHz G5 processor, because of the Cell core's simpler microarchitecture. The higher clock speed of the Cell processor can still process many math operations faster than the G5 can, so we would give the performance edge to the Cell processor. IBM and its partners are not disclosing performance data at this time.

One design goal of the Cell processor was predictable execution times, so that programmers could better estimate the processing time of their software to meet frame rates. While this provided another reason for in-order execution processors, there are also extensive timers and counters that can be used to manage the real-time response of the system.

The tool chain for Cell is built on PowerPC Linux. The programming of the SPE is based on C, with limited C++ support. Software research is under way for Fortran and other languages. Debugging tools include extensions for P-Trace and extended Gnu debugger (GDB). The ultimate goal of the software research is to build an abstraction layer on top of the hardware that can scale with additional Cell processors or Cell processors with differing amounts of resources. Programming the Cell processor will be unlike programming any other processor in mainstream use. It will require new tools, and possibly a new programming paradigm, because programs for the SPE should be self-contained with data and instruction bundles (or Cells). This is not the same programming model used for languages with strict class structures like Java. Because the first system implementation of the Cell processor is a game and media console, programmers will craft custom and optimized code for it. To gain more widespread use, the STI partners will need to develop a more mainstream solution, possibly a software virtual-machine architecture on top of the processor.

	Sony Emotion Engine	Cell Processor
CPU Core ISA	MIP64	64-bit Power Architecture
Core Issue Rate	Dual	Dual
Core Frequency	300MHz	~4GHz (est.)
Core Pipeline	6 stages	21 stages
Core L1 Cache	16KB I-Cache + 8KB D-Cache	32KB I-Cache + 32KB D-Cache
Core Additional Memory	16KB scratch	512KB L2
Vector Units	2	8
Vector Registers (#, width)	32, 128-bit + 16, 16-bit	128, 128-bit
Vector Local Memory	4K/16KB I-Cache + 4K/16KB D-Cache	256KB unified
Memory Bandwidth	3.2GB/s peak	25.6GB/s peak (est.)
Total Chip Peak FLOPS	6.2GFLOPS	256GFLOPS
Transistor Count	10.5 million	235 million
Power	15W @ 1.8V	~80W (est.)
Die Size	240mm^2	235mm^2
Process	250nm, 4LM	90nm, 8LM + LI

Table 1. A comparison between the Cell processor and the Emotion Engine from the Sony PlayStation 2 shows well over an order of magnitude performance improvement, but it looks to be a bit short of two orders of magnitude.

The Cell processor will be fabricated in IBM's 90nm SOI process, with eight levels of metal and one layer of local interconnect, at IBM's Fishkill fab and Sony's fab. The production version of the Cell processor will have a die size of 235mm^2. At ISSCC, IBM presented material on a prototype version of the chip that is only 221mm^2. Power dissipation levels have not been released, but we estimate that 80W at 1.2V and 4GHz is not unreasonable. The schmoo plot shown at ISSCC indicated the processor could run at 3GHz at a 0.9V core voltage, which would reduce power considerably. As mentioned earlier, the final specifications of the chip are still being determined for system integration. The die also has 2,965 C4 bumps.

IBM, Sony, and Toshiba have put together a unique processor that could signal the return of the vector processor. This time around, the vector

design has the support of some heavy hitters. A look back at the Sony PlayStation 2 design shows there were two vector units in the Emotion Engine chip as well, so the SPEs can be considered an extension of the older design (see Table 1). But this chip is a quantum leap over the Emotion Engine and with the parallel floating-point performance is a virtual supercomputer on a chip.

Despite the excessive hype surrounding the Cell processor at ISSCC, the possibilities of processing this much media and graphics data are still being explored. The potential impact is very exciting if the right programming model can be found to utilize the capabilities. We have only scratched the surface of the chip and its potential. Luckily, we can count on an army of game programmers to dig deep into this chip, and we'll hopefully be able to learn from them how much capability can be exploited. The next challenge for the STI Design Center (after building the chip and systems) will be to find a way to make this architecture accessible to programmers beyond the aforementioned gamer developers.

Price & Availability

Cell will be used in Sony's next-generation gaming console and in Toshiba TVs. Those products are expected in 2006.

Editor's Note: *The Cell architecture deserves more exploration and* MPR *will have additional coverage on it in the near future, including more details on the architecture and programming model issues.*