Microarchitecture and implementation of the synergistic processor in 65-nm and 90-nm SOI

B. Flachs
S. Asano
S. H. Dhong
H. P. Hofstee
G. Gervais
R. Kim
T. Le
P. Liu
J. Leenstra
J. S. Liberty
B. Michael
H.-J. Oh
S. M. Mueller
O. Takahashi
K. Hirairi
A. Kawasumi
H. Murakami
H. Noro
S. Onishi
J. Pille
J. Silberman
S. Yong
A. Hatakeyama
Y. Watanabe
N. Yano
D. A. Brokenshire
M. Peyravian
V. To
E. Iwata

This paper describes the architecture and implementation of the original gaming-oriented synergistic processor element (SPE) in both 90-nm and 65-nm silicon-on-insulator (SOI) technology and introduces a new SPE implementation targeted for the high-performance computing community. The Cell Broadband Engine™ processor contains eight SPEs. The dual-issue, four-way single-instruction multiple-data processor is designed to achieve high performance per area and power and is optimized to process streaming data, simulate physical phenomena, and render objects digitally. Most aspects of data movement and instruction flow are controlled by software to improve the performance of the memory system and the core performance density. The SPE was designed as an 11-FO4 (fan-out-of-4-inverter-delay) processor using 20.9 million transistors within 14.8 mm^2 using the IBM 90-nm SOI low-k process. CMOS (complementary metal-oxide semiconductor) static gates implement the majority of the logic. Dynamic circuits are used in critical areas and occupy 19% of the non–static random access memory (SRAM) area. Instruction set architecture, microarchitecture, and physical implementation are tightly coupled to achieve a compact and power-efficient design. Correct operation has been observed at up to 5.6 GHz and 7.3 GHz, respectively, in 90-nm and 65-nm SOI technology.

Introduction

As gaming develops into an immersive experience with more realistic rendering, object movement, and strategy, it is becoming increasingly similar to high-performance computing (HPC). To achieve high levels of realism, traditional HPC algorithms and those with characteristics much like HPC algorithms are being used in gaming. These algorithms often process massive amounts of data in a manner that can be partitioned to enable parallel execution. Throughput is often more important to HPC and gaming than the general-purpose thread with complex branching schemes. Gaming and HPC are also similar in that they are limited by factors such as component cost and power dissipation. More often than not, the question for HPC and gaming is not the speed at which a single thread can run, but the sustainable throughput per unit of system cost.

Two algorithmic methods of partitioning are popular. Vectorization has long been a staple of simulation and solution within the HPC community, while today's media-rich application software is often characterized by multiple lightweight threads and software pipelines. The trend in gaming is to merge these two characteristics. This trend in software design favors processors that utilize characteristics of vector processing and multiple threads. Software running on these processors can efficiently drive the improved memory bandwidth becoming available from commodity memory systems, while software that runs on processors designed to accelerate a single thread of execution by taking advantage of instruction-level parallelism is much less able to derive benefit from the new memory systems.

Note: Portions of this paper are based on earlier publications by the authors. ©2006 IEEE. Reprinted, with permission, from References [1] and [2].

0018-8646/07/$5.00 © 2007 IBM

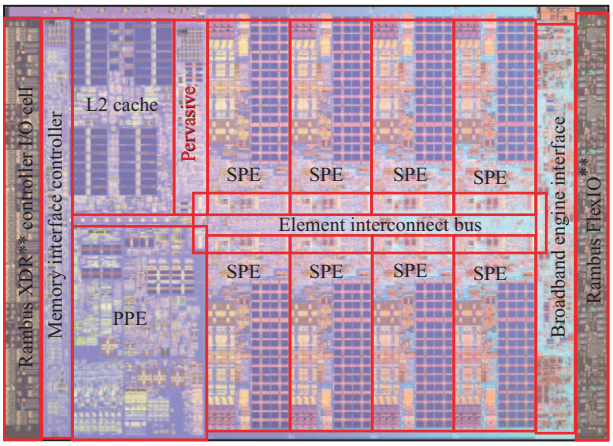


Figure 1

The 65-nm Cell Broadband Engine processor.

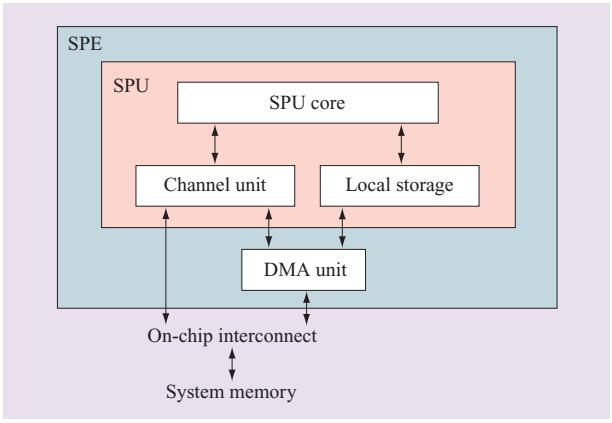


Figure 2

SPE architecture. (©2006 IEEE. Reprinted, with permission, from Reference [1].)

Memory latency is a key limiter to processor performance. Modern processors lose up to 4,000 instruction slots while they wait for data from main memory. Previous designs emphasized large caches and reorder buffers to reduce the average latency and to maintain instruction throughput while waiting for data from cache misses. However, because the reuse rate for much of the data is low, caches that can deliver high hit rates on large data structures consume large amounts of area that might have been used by computational elements. When the cache misses, processing can continue through branch prediction to fill a reorder buffer. However, it would be difficult to build a reorder buffer large enough to continue through a main memory access.

Transistor oxides are now only a few atomic levels thick, and the channels are extremely narrow. These features improve transistor performance and increase transistor density, but they tend to also increase leakage current. As processor performance becomes power limited, leakage current becomes an important performance issue. Since leakage is proportional to area, processor designs must extract more performance per transistor. Since the performance efficiency of caches and reorder buffers diminishes as they increase in size, another approach is necessary.

Architecture

Figure 1 shows the 65-nm Cell Broadband Engine[†] (Cell/B.E.) processor, a heterogeneous shared-memory multiprocessor [3] featuring a multithreaded 64-bit IBM PowerPC* processor element (PPE) and eight synergistic processor elements (SPEs). Performance per transistor is the motivation for heterogeneity. Software can be divided into general-purpose computing threads, operating system (OS) tasks, and streaming media threads. Each of these tasks can be targeted to a processing core customized for that particular task. For example, the PPE is responsible for running the OS and coordinating the flow of data processing threads through the SPEs. This differentiation allows the architectures and implementations of the PPE and SPE to be optimized for their respective workloads and enables significant improvements in performance per transistor. For example, branch history tables are valuable to general-purpose code and are present in the PPE design, but have little value for data processing loops and are not present in the SPE design. Thus, heterogeneity allows the Cell/B.E. processor to include nine processors in the same area as a dual-core solution based on an industry-competitive general-purpose processor core and the L2 cache that would be necessary to achieve good performance.

Figure 2 shows the major entities of the SPE architecture and their relationships. Local storage (LS) is a private memory for SPE instructions and data. The synergistic processor unit (SPU) core is a processor that runs instructions from the LS and can read from or write to the LS with its load and store instructions. The direct memory access (DMA) unit transfers data between LS and system memory. The DMA unit is programmable by SPU software using the channel unit. The channel unit is a message-passing interface that allows the SPU core to communicate with both the DMA unit and other units in the Cell/B.E processor. The channel unit is accessed by the SPE software through channel access instructions.

The SPU core is a single-instruction multiple-data (SIMD) reduced instruction set computing (RISC) processor [4]. All instructions are encoded in 32-bit fixed-length instruction formats, and no instructions are

difficult to pipeline. The SPU features 128 general-purpose registers (GPRs) that are used by both floating-point (FP) and integer instructions. The shared register file allows for a high level of performance for various workloads using only a small number of registers. Loop unrolling, which is necessary to fill functional unit pipelines with independent instructions, is possible with the 128 registers. Most instructions operate on 128-bit-wide data. For example, the FP multiply–add instruction operates on vectors of four 32-bit single-precision (SP) FP values. Some instructions, such as FP multiply–add, consume three register operands and produce a register result. The SPU includes instructions that perform SP FP integer arithmetic, logical operations, loads, stores, compares, and branches. Some of the instructions of the SPU are intended specifically to support media applications. The instruction set is designed to simplify compiler register allocation and code schedulers. Most SPE software is written in C or C++ with intrinsic functions.

The SPE architecture reduces area and power while facilitating improved performance by requiring software to solve difficult scheduling problems, such as data fetch and branch prediction. Software solves these problems by including explicit data movement and branch prediction directives in the instruction stream. Because the OS is not run on the SPE, it is optimized for user-mode execution. SPE load and store instructions are performed within a local address space, not in system address space. The local address space is untranslated, unguarded, and noncoherent with respect to the system address space, and it is serviced by the LS. The LS is a private memory, not a cache, and does not require tag arrays or backing store. Loads, stores, and instruction fetch can complete with a fixed delay and without raising exceptions. These properties of LS greatly simplify the SPU core design and provide predictable real-time behavior. This design reduces the area and power of the core while allowing for operation at a higher frequency.

Data is transferred to and from the LS in 128-byte lines by the SPE DMA engine similar to the way a supercomputer uses vector load and store instructions to move data to and from the vector register file. Data moves between system memory and the DMA engine using the on-chip interconnect. The on-chip interconnect sources and sinks 16 bytes of data every other cycle. The SPE DMA engine allows SPE software to schedule data transfers in parallel with core execution. **Figure 3** shows a timeline of how software can be organized into cooperative coarse-grained threads that take advantage of nonblocking DMA commands to facilitate overlap between data transfer and core computation. Thread management can be done by the software itself or through a task manager application programming

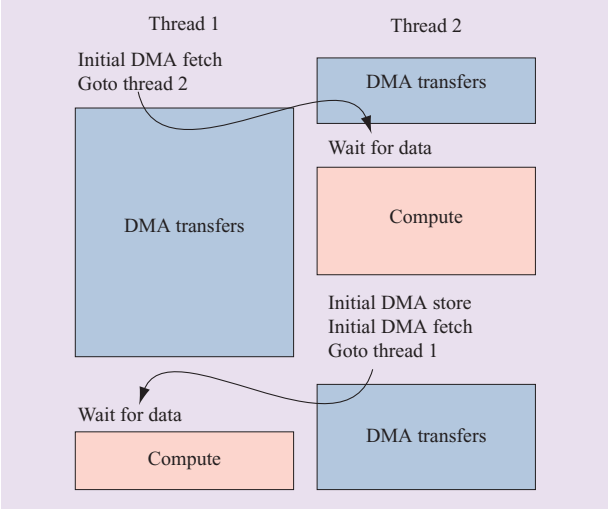


Figure 3

Example of the timeline of concurrent computation and memory access. (©2006 IEEE. Reprinted, with permission, from Reference [1].)

interface. In this example, the threading is directly encoded into the software so that as thread 1 finishes its computation, it initiates a DMA fetch of its next data set and branches to thread 2. Thread 2 begins by waiting for its previously requested data transfers to finish and begins computation while the DMA engine obtains the data needed by thread 1. When thread 2 completes the computation, it programs the DMA engine to store the results to system memory and fetch the next data set from system memory. Thread 2 then branches back to thread 1. Techniques such as double buffering and coarse-grained multithreading allow software to overcome memory latency and achieve high memory bandwidth and improved performance. The DMA engine can process up to 16 commands simultaneously, and each command can fetch up to 16 KB of data. These transfers are divided into 128-byte packets for the on-chip interconnect. The DMA engine can support up to 16 packets in flight at a time. DMA commands are richer than a typical set of cache prefetch instructions because they can perform scatter or gather operations from system memory or set up a complex set of status reporting and notification mechanisms. Software can achieve much higher bandwidth through the DMA engine than it could with a hardware prefetch engine; in addition, with the DMA engine, a much higher fraction of that bandwidth is useful data than with the speculative prefetch engine design.

The SPE programs the DMA engine through the channel interface, a messaging-passing interface intended to overlap input/output (I/O) with data processing and

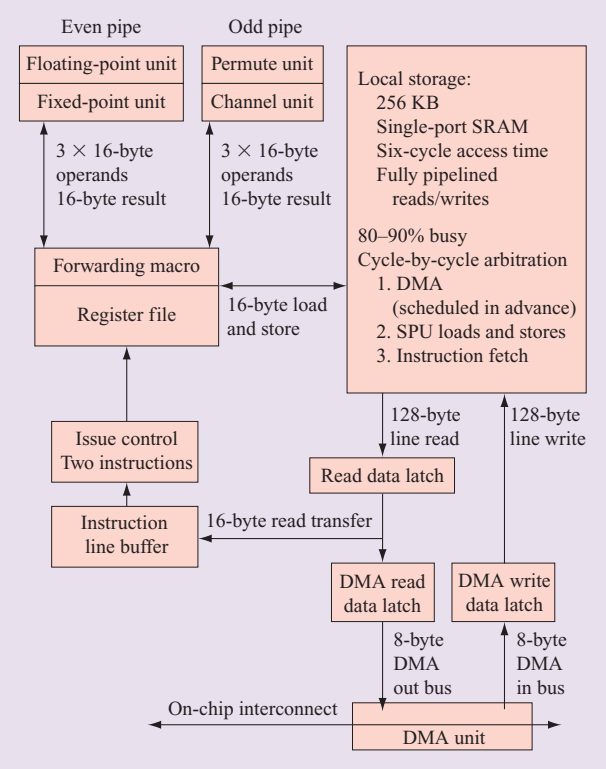
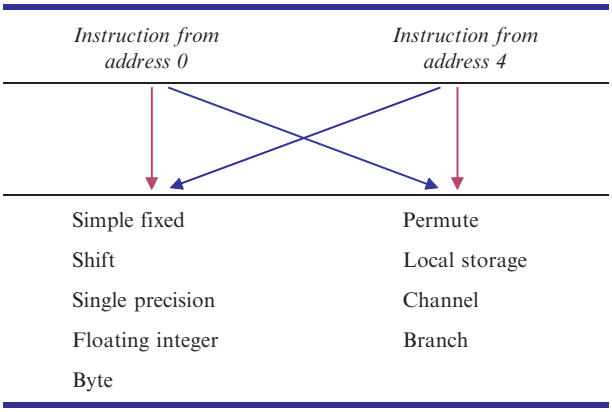


Figure 4

SPE organization. (©2006 IEEE. Reprinted, with permission, from Reference [1].)

Table 1 Dual-issue unit assignments.



minimize the power consumed by synchronization. Each device is allocated one or more channels through which messages can be sent to or from the SPU core. Channels have sufficient capacity for multiple messages to be queued and for the SPU to send multiple commands to a device in pipelined fashion without incurring delay, until the channel capacity is exhausted. When a channel is exhausted, the write or read instruction stalls the SPU in a low-power wait mode until the device becomes ready. Channel wait mode can often substitute for polling and represents significant power savings. Channel facilities are accessed with three instructions: read channel, write channel, and read channel count, which measures channel capacity. The SPE architecture supports up to 128 unidirectional channels. The channel architecture of the SPE enables pipelined computation and communication in a latency-tolerant and power-efficient manner.

The SPE has separate 8-byte-wide inbound and outbound data buses. The DMA engine supports transfer requests generated by both the local SPU and the requesters external to the local SPE. External requests can be programmed into the external request queue or received as incoming element interconnect bus (EIB) read or write requests to addresses within the system real-address range assigned to the LS. The SPE request queue supports up to 16 outstanding transfer requests. Each request can transfer up to 16 KB of data to or from the local address space. DMA request addresses are translated by the memory management unit (MMU) before the request is sent to the bus. Software can check or be notified when requests or groups of requests are completed.

Microarchitecture

The microarchitecture of the SPE is designed to support very high frequency operation. One measure of delay that can be used to compare designs in different technologies is the fan-out-of-4 (FO4) inverter delay, which is the time required for an inverter to drive four copies of itself. In these terms, the SPE is an 11-FO4 design. This allows four to eight stages of logic per cycle, depending on the distance that must be traveled during the cycle. Many signal distribution paths feature about 35% of the cycle time in wire delay. At first glance, high frequency would not seem to be a good low-power option because, relative to a 22-FO4 design [5], there are twice as many registers per pipeline. However, a 22-FO4 design would require two pipelines to achieve the same throughput as the 11-FO4 pipeline. From this point of view, the number of registers might be about equal, but the 11-FO4 pipeline would have only about half as many logic transistors as the two pipelines in the 22-FO4 design. In the 11-FO4 design, some of the registers in the pipeline can be implemented as pulsed latches, reducing the 11-FO4 to 22-FO4 latch ratio from 2:2 to 1.5:2, a net clock load savings of 25% and a logic area savings of 50%, which translates into significant active power and leakage power savings. Thus, for streaming and vector workloads that can take advantage of the large register file to hide instruction latency, the 11-FO4 design reduces both area and power while achieving very high performance levels.

Table 2 Unit and instruction latency.

Unit	*Instructions*	*Execution pipe*	*Unit pipeline depth*	*Instruction latency*
Simple fixed	Word arithmetic, logicals, count leading zeros, selects, and compares	Even	2	2
Simple fixed	Word shifts and rotates	Even	3	4
Single precision	Multiply–accumulate	Even	6	6
Single precision	Integer multiply–accumulate	Even	7	7
Byte	Pop count, absolute sum of differences, byte average, and byte sum	Even	3	4
Permute	Quad-word shifts, rotates, gathers, shuffles, and reciprocal estimate	Odd	3	4
Local storage	Load and store	Odd	6	6
Channel	Channel read and write	Odd	5	6
Branch	Branches	Odd	3	4

Figure 4 shows the SPE organization and the key bandwidths (per cycle) between units. Instructions are fetched from the LS in 32 four-byte groups when the LS is idle. Fetch groups are aligned to 64-byte boundaries to improve the effective instruction fetch bandwidth. The fetched lines are sent in two cycles to the instruction line buffer (ILB), which stores 3.5 fetched lines [1]. A half-line holds instructions until they are sequenced into the issue logic, while another line holds the single-entry software-managed branch target buffer (SMBTB). Two lines are used for inline prefetching, and instructions are sent two at a time from the ILB to the issue control unit.

The SPE issues and completes all instructions in program order and does not reorder or rename its instructions. Although the SPE is not a very long instruction word processor (VLIW), it does feature a VLIW-like dual-issue feature and can issue up to two instructions per cycle (IPC) to nine execution units organized into two execution pipelines (**Table 1**). Instruction pairs can be issued if the first instruction (from an even address) is routed to an even pipe unit and the second instruction to an odd pipe unit. Execution units are assigned to pipelines to maximize dual-issue efficiency for a variety of workloads. SPE software does not require NOP (nothing-operation) padding when dual issue is not possible. Instruction issue and distribution require three cycles. The simple issue scheme provides for very high performance, saves at least one pipeline stage, simplifies resource and dependency checking, and contributes to the extremely low fraction of logic devoted to instruction sequencing and control.

Operands are fetched from either the register file or the forward network and sent to the execution pipelines. Each of the two pipelines can consume three 16-byte operands and produce a 16-byte result every cycle. The register file has six read ports, two write ports, and 128 entries of 128 bits each, and it is accessed in two cycles. Register file data is sent directly to the functional unit operand latches. Results produced by functional units are held in the forward macro (FM) until they are committed and available from the register file. These results are read from six FM read ports and distributed to the units in one cycle.

Loads and stores transfer 16 bytes of data between the register file and the LS, which is a six-cycle, fully pipelined, single-ported, 256-KB static random access memory (SRAM). The LS is shared between the SPE load and store unit, the SPE instruction fetch unit, and the DMA unit. Several workloads keep the LS busy between 80% and 90% of their cycles. To provide good performance while keeping the processor simple, a cycle-by-cycle arbitration scheme is used. DMA requests are scheduled in advance but are first in priority. DMA requests access 128 bytes in the LS in a single cycle, providing more than sufficient bandwidth with relatively little interference with the SPE loads and stores. Loads and stores are second in priority and wait in the issue stage for an available LS cycle. Instruction fetch accesses the LS when it is otherwise idle, again with a 128-byte access to minimize the chances of performance loss due to instruction run-out.

Table 2 shows the execution units. Simple fixed-point [6], FP [7], and load results are bypassed directly from the unit output to the input operands to reduce result latency. Other results are sent to the FM, from which they are distributed a cycle later. Result bypassing involves pushing result data from the outputs of the functional units over 128-bit-wide buses that cover about half of the data-flow height. This distribution requires

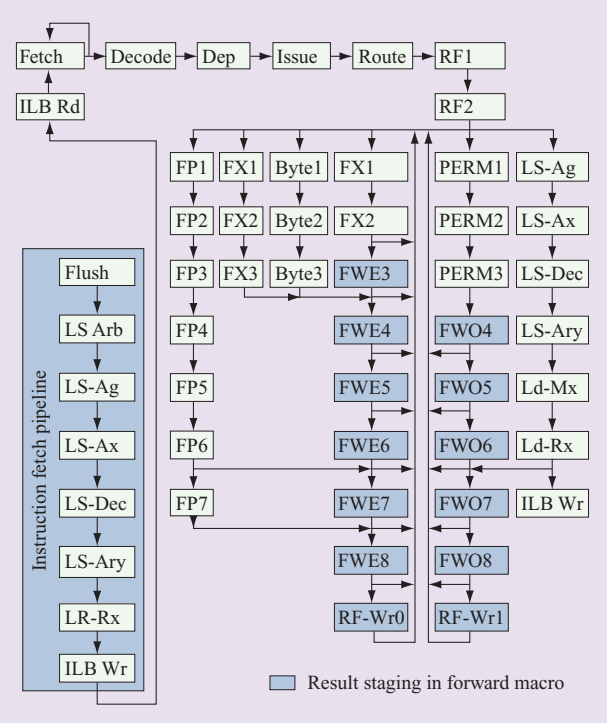


Figure 5

SPE pipeline. (©2006 IEEE. Reprinted, with permission, from Reference [1].)

about half of the 11-FO4 cycle time and forces the simple fixed-point operations to take two cycles rather than one.

Figure 5, the SPE pipeline, shows how flush and fetch are related to other instruction processing. Although frequency is an important element of SPE performance, pipeline depth is similar to that found in 20-FO4 processors. Circuit design, efficient layout, and logic simplification are the keys to supporting the 11-FO4 design frequency while constraining pipeline depth.

To save area and power, the SPE omits hardware branch prediction and branch history tables. However, mispredicted branches flush the pipelines and consume 18 cycles, so it is important that software employ techniques to avoid mispredict. Whenever possible, the common case for conditional branches should be arranged to be an entirely inline sequence of instructions. When the common case cannot be identified, it is often advantageous to compute both paths and use the select instruction to select the correct results at the end of the block. When a commonly taken branch is necessary, especially for the backward branch of a loop, software can utilize the SMBTB, which is loaded using the branch hint instructions (*hbr*, *hbra*, and *hbrr*). These instructions identify both the address of a branch and the probable address of the branch target. When a branch hint instruction is executed, instructions from the branch target are read from the LS and written to the SMBTB. Later, when the indexed branch instruction is sent to the issue logic, the SPE instruction sequencer can send the first instruction of the branch target in the next consecutive cycle. In this way, a correctly hinted branch can execute in a single cycle.

Table 3 shows performance on several workloads that are written in C using intrinsics for the SIMD types and DMA transfers. SP LINPACK, Advanced Encryption Standard (AES), and the transform-light algorithm (a critical stage commonly found in vertex-based graphics pipelines) achieve performance very close to the SPE peak of two IPC. The LINPACK data is especially impressive in that the DMA required to move the data into and out of the LS is entirely overlapped with execution. The other workloads execute out of the SPU LS. Transform light is a computationally intensive application that has been unrolled four times and its software pipelined to schedule out most instruction dependencies. The relatively short instruction latency is important. If the pipelines were deeper, this algorithm would require further unrolling to hide the extra latency; this unrolling would require more than 128 registers and would thus be impractical. These benchmarks show that the SPE is a very effective streaming-data processor, even when running software written in high-level languages. This is due in part to the simplicity of the instruction set architecture, the large register file, and the relatively short execution pipelines, making it easy for the compiler [8, 9] to schedule code for the SPE. These benchmarks also show that the DMA programming model can overcome memory latency and allow the SPU core to achieve peak performance rather than wait for needed data. Even with a very simple microarchitecture and very small area, the SPE can compete on a cycle-by-cycle basis with more complex cores on non–double-precision (DP) streaming workloads.

DP LINPACK is one of the applications from the class of DP HPC applications in which the weak DP performance of the SPE can be exposed. Architecturally, the SPU executes two-way SIMD DP multiply–add instructions. However, the current SPU implementation can execute only one of these instructions every seven cycles. This means that at 3.2 GHz, the SPU peaks at only 1.8 DP Gflops. The Cell/B.E. processor DP HPC performance was studied by Williams et al. [10]. They report that, in general, the current Cell/B.E. processor outperforms general-purpose processors by a factor of 3 to 18 and is competitive with the Cray X1E** processor on several DP algorithms. The algorithms studied overcame the weak DP performance by taking full advantage of the 25-GB/s memory bandwidth. The authors conclude that

Table 3 Cell/B.E. processor application performance.

Class	*Algorithm*	*General-purpose processor*[b]	*3.2-GHz Cell/B.E. processor*	*IPC/SPE*	*Relative performance*
Computing	Single-precision matrix multiplication	25.6 Gflops (with SIMD)	25.6 Gflops (8 SPEs)	1.96	8×
	Single-precision LINPACK 4K × 4K	25.6 Gflops (with SIMD)	156 Gflops (8 SPEs)	1.64	6×
	Double-precision LINPACK 1K × 1K	7.2 Gflops (3.6 GHz/SSE3)	9.67 Gflops	0.30	1.3×
	Single-precision FFT[a] 16M	1.2 Gflops (2 GHz)	46.8 Gflops		24×
Graphics	Terrain rendering engine	0.85 fps (2.7 GHz/VMX)	30 fps		35×
	Transform light	128 Mvtx/s (2.7 GHz/VMX)	8 × 217 Mvtx/s	1.50	13×
Security	AES ECB encrypt 128-bit key	1.03 Gb/s	8 × 2.06 Gb/s		16×
	AES ECB decrypt 128-bit key	1.04 Gb/s	8 × 1.5 Gb/s		11×
	TDES ECB encrypt	0.13 Gb/s	8 × 0.17 Gb/s	1.80	10×
	DES ECB encrypt	0.43 Gb/s	8 × 0.49 Gb/s		9×
	SHA-1	0.9 Gb/s	8 × 2.12 Gb/s		18×
Video	MPEG-2 decode (SDTV)	354 fps (with SIMD)	8 × 329 fps	0.80	7×

AES: Advanced Encryption Standard; DES: Data Encryption Standard; MPEG: Moving Picture Experts Group; Mvtx/s: millions of vertices per second; SDTV: standard-definition television; SHA: secure hash algorithm; TDES: Triple Data Encryption Standard; fps: frames per second.
[a]A programming example: Large fast Fourier transform (FFT) on the Cell Broadband Engine.
[b]3.2-GHz Intel Pentium** 4 processor, except where noted.

the DMA model offers advantages over traditional cache-based architectures but suggest that DP HPC could benefit, by up to a factor of 2, from improved DP FP performance. A new SPU design featuring a larger, fully pipelined, two-way SIMD DP FP unit (FPU) is planned to be a part of the Los Alamos National Laboratories petaflop supercomputer [11]. This HPC-oriented design is intended to improve the DP LINPACK performance by a factor of 6 over the original gaming-oriented design. The new level of SPE DP performance should allow the HPC-oriented Cell/B.E. processor to solidly outperform its general-purpose counterparts and lead the way into the next generation of gaming and HPC.

Implementation

About one-half of the 65-nm Cell/B.E. processor shown in Figure 1 is dedicated to eight SPEs. The large fraction of chip area and the high repeat count make the physical implementation of the SPE very important. A slightly larger SPE might have meant that the number of SPEs would have been reduced from eight to six, resulting in up to 25% performance loss.

Figure 6 shows the floorplans of both the SPU and the memory flow controller (SMF) that make up the original gaming-oriented SPE. This SPE has two primary stacks. The left-hand stack has four 64-KB SRAM arrays [6] that implement local storage, while the right-hand stack is the SIMD data flow. The SIMD data flow has a 128-bit, 128-entry unified register file (GPR) with six read ports and two write ports, four 32-bit fixed-point units (SFX) [7], and four SP FPUs (SFP) [12]. Between the data flow and the LS is the control bay stack. The top third of the control bay stack is filled with a DP FPU, while the bottom features all of the instruction fetch and sequencing logic. This floorplan illustrates the importance of heterogeneity. The instruction sequencing logic occupies only about 10% of the SPU area. This is a small fraction of the area required by the corresponding functions in an industry-standard general-purpose processor, which has large structures for instruction caching, branch prediction, renaming, and out-of-order completion. In general, comparisons with general-purpose processors show that about four SPEs fit in the area of a general-purpose processor and its L2 cache.

The HPC SPE design deletes the DP FPU from the control bay stack and adds two DP FPUs at the top of the data flow. Thus, the HPC SPE can execute the SIMD DP instructions in a fully pipelined manner, while the gaming SPE must double-pump its single DP unit. The SMF is designed to operate at half the frequency of the SPU.

The SPE design has roughly 20.9 million transistors, and the chip area including the SMF is 14.8 mm^2 (2.54 mm × 5.81 mm) fabricated with a 90-nm silicon-on-insulator (SOI) technology. The 65-nm version of the design is 10.5 mm^2. Full CMOS static gates and transmission gates are used to implement the majority of the logic. Dynamic circuits are used for the timing-critical areas, including forwarding, control signal generation,

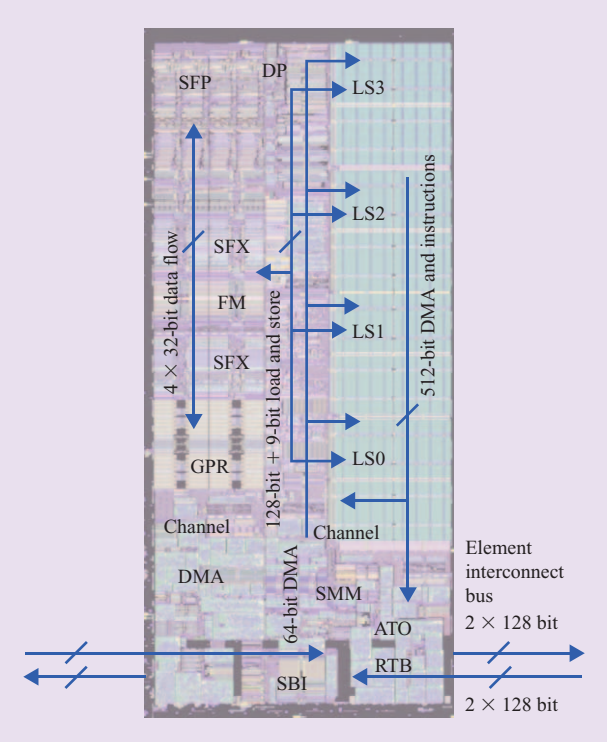


Figure 6

The 90-nm gaming SPE with engineered buses. (ATO: atomic unit for synchronization; FM: forward macro; GPR: unified register files; RTB: self-test unit; SBI: SPE bus interface; SFP: single-precision FPUs; SFX: 32-bit fixed-point units; SMM: SPE memory management unit.) (©2006 IEEE. Reprinted, with permission, from Reference [2].)

GPR, load and store, and dynamic programmable logic arrays (DPLAs), which occupy 19% of the non-SRAM area in the SPE. Extensive clock gating is implemented, ensuring that only necessary logic is activated. The SPE has been tested at various temperatures, supply voltages, and operating frequencies. Correct operation of a 90-nm SPE has been observed at up to 5.6 GHz at 1.4-V supply and 56°C, while the 65-nm SPE has run at up to 7.2 GHz at 1.35-V supply.

Latch use

Four kinds of latches [3] are used in the SPE. Most of these are transmission gate latches with either a single port or a two-way multiplexer (mux) configuration. Where a wider mux is required and timing is critical, dynamic scannable mux latches are used with various configurations from a three-way to a nine-way mux. Where area and timing are critical, pulsed clock latches are used. Finally, DPLA latches are used where wide ANDing and ORing are required and timing is critical.

Table 4 Simulated active power of DCM (mW).

Data input switching factor (%)	*Clock activity* (%)				
	0	*25*	*50*	*75*	*100*
0	2.0	8.1	14.1	20.2	26.3
5	2.3	10.9	19.4	28.0	36.5
10	2.7	13.7	24.7	35.7	46.7
20	3.4	19.3	35.2	51.2	67.1
30	4.1	24.9	45.8	66.7	87.5
38	4.6	29.4	54.2	79.0	103.8
40	4.8	30.6	56.4	82.1	107.9
50	5.5	36.2	66.9	97.6	128.3

GPR file and data flow

Three cycles are required for GPR operation (**Figure 7**). The first cycle is for address predecoding and decoded signal distribution; the second cycle is for final decoding and array access; and the third cycle is for data-flow distribution. A read operation is performed first, followed by a write operation within an 11-FO4 cycle. Up to eight operations, six reads and two writes (6r2w), are performed independently every cycle. Collisions are avoided by the SPE control logic. A dynamic two-stage domino read scheme is used for a read operation, and a static write scheme is used for a write operation. The data from each of three read ports is taken from the true side of a register cell, and the data from the other three read ports is taken from the complement side of a register cell. This leads to three true read ports and three complement read ports at the GPR macro boundary. The selection of ports with true or complement data is carefully made so that the number of inverters in the data flow distributing the GPR data is minimized to reduce path delay. The wire level, width, and spacing for all major signal distributions in the data flow are predetermined by Spice (simulation program with integrated circuit emphasis) simulations. Actual wires are then implemented using a specially developed routing tool to ensure optimized signal quality and distribution delays. This tool is used for all data-flow signals, load and store paths, DMA access paths, and instruction fetch paths (Figure 6).

Dependency checking and data forwarding

Dependency checking and data forwarding are performed by several macros: the dependency check macro (DCM), the FM, and several DPLAs. DCM compares the newly issued instruction with those in the execution pipes. The subsequent DPLAs determine the final dependencies,

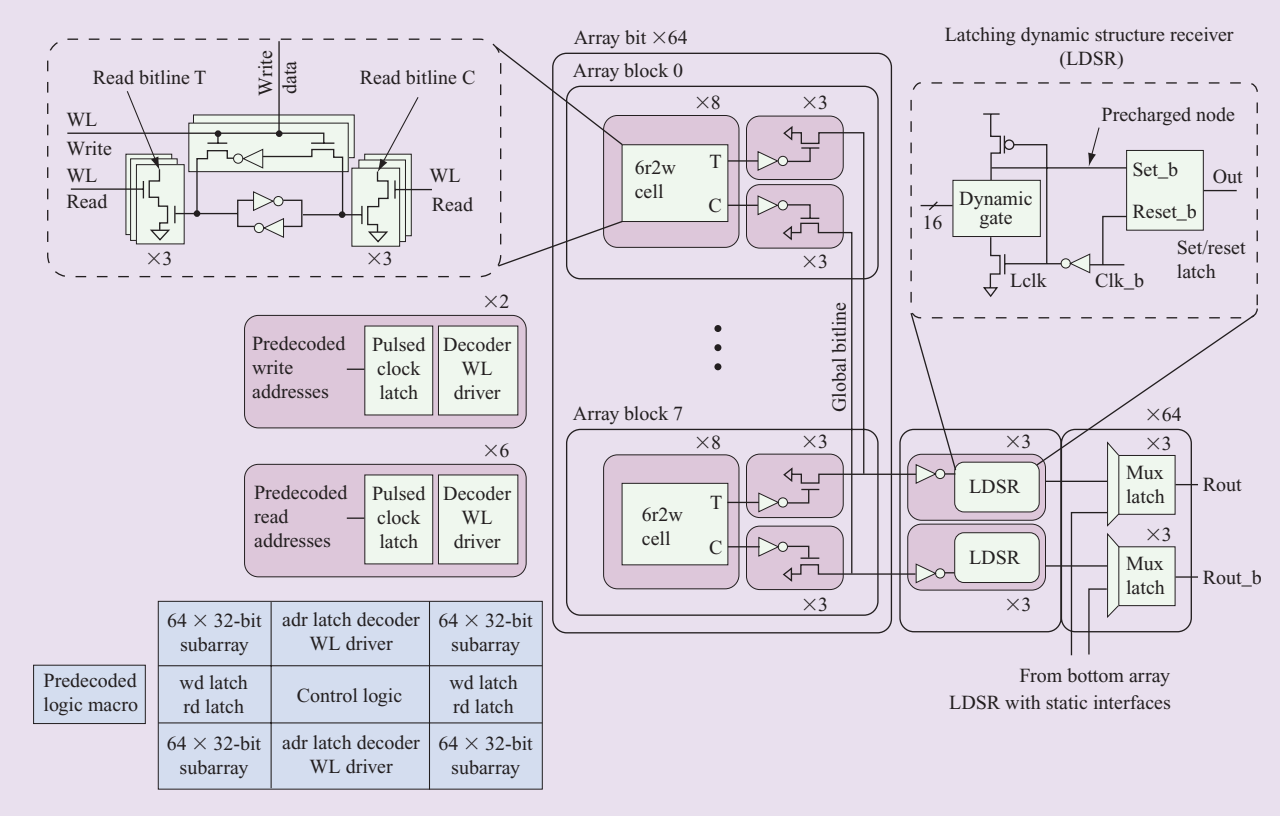


Figure 7

SPE GPR file circuit diagram and organization. (adr: address; C: complement; rd: read data; T: true; wd: write data; WL: wordline.) (©2006 IEEE. Reprinted, with permission, from Reference [2].)

including exceptions, as well as the dependencies among the committed instructions. The data in the correct stage in FM is selected and forwarded to the receiving operand latches (dynamic mux latches) according to the results of the DCM and DPLAs. The results from DPLAs are used to select the final data at the mux in the receiving operand latches. DCM is implemented with the combination of full static CMOS and transmission gates [**Figure 8(a)**]. Its circuit design, layout, placement, and wiring to and from the DCM are highly customized in order to satisfy the 11-FO4 cycle time. The simulated active power numbers [13] of the DCM are tabulated with respect to clock activity and the fraction of input data signals that switch in a cycle in **Table 4**.

One slice of FM consists of sixteen 32-bit registers and six 16-way muxes [**Figures 8(b)** and **8(c)**]. Each register contains either a two-way or a three-way mux, depending on the stage. When the FM is operated in shift mode, the register data in a stage is shifted to the next stage. When the data in a stage is forwarded to the destination operand latches, the 16-way mux path is selected. The 16-way mux is implemented with a dynamic 8-way NOR gate followed by a cross-coupled NAND (CCNAND) gate and finally a 2-way static NAND to complete the 16-way multiplexing. Special attention has been paid to the physical implementation of the bitlines in the 16-way mux to avoid any crosstalk among both the macro internal and the chip-level data-flow wiring.

DPLAs are generated with a specially developed program. The program receives an espresso file as an input and generates both schematics and layouts. ANDing is implemented with a dynamic footed NOR followed by a strobe circuit. Then ORing is done using a dynamic footless NOR followed by a scannable CCNAND [**Figure 8(d)**]. Twenty-seven DPLAs with 18 configurations are used in the 90-nm SPE. DPLA AND and OR planes can accept engineering changes by modifying metal 1, metal 2, and the contacts between and below metal 1. DPLAs are selectively used only when an equivalent static implementation cannot make the required timing. Some DPLAs have been replaced with static logic in the 65-nm SPE, reducing the number of DPLA configurations from 18 to 12. These reductions typically save area and power

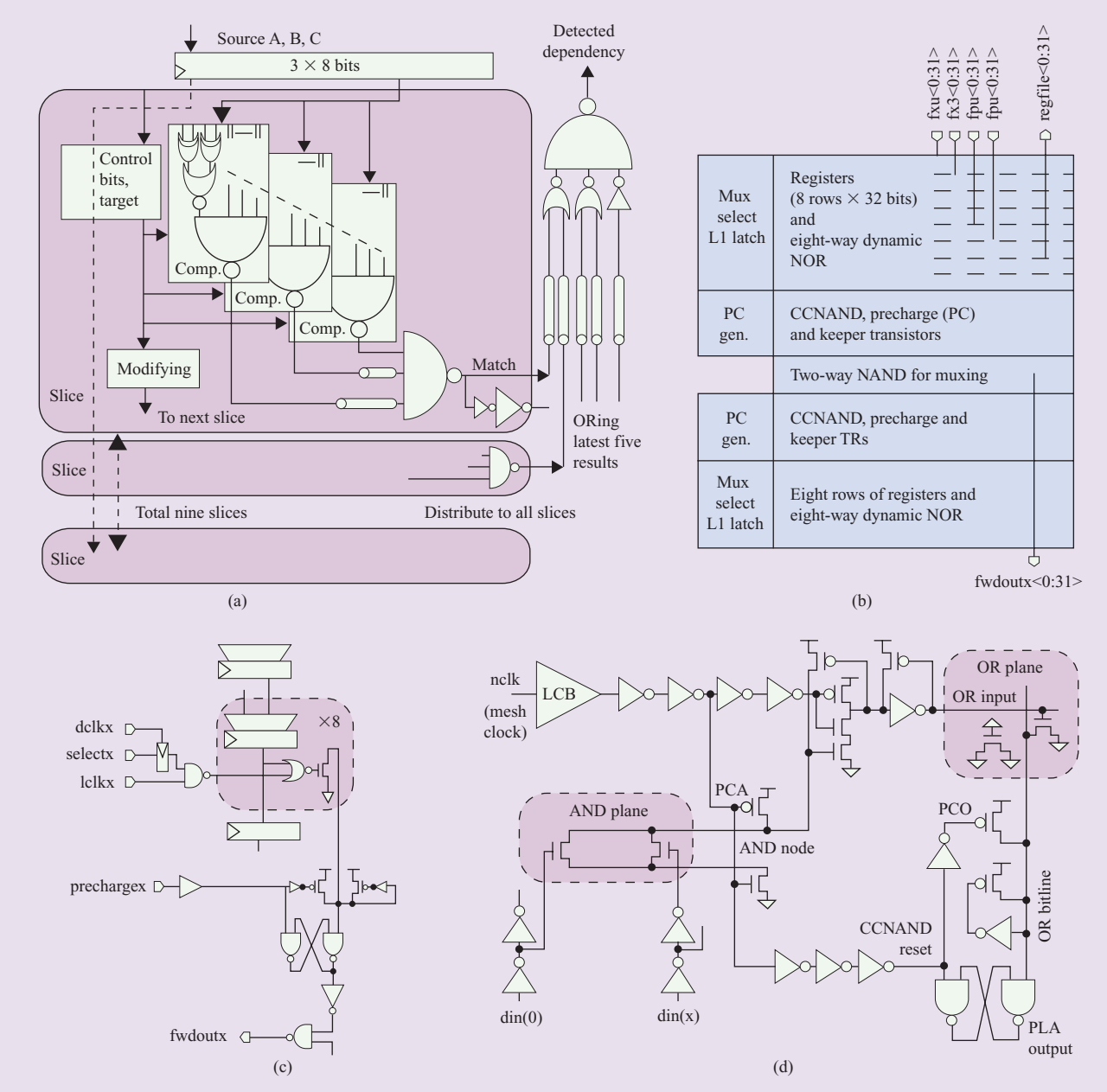


Figure 8

Dependency checking and data forwarding: (a) DCM circuit and organization; (b) forward macro organization; (c) forward macro circuit diagram; (d) dynamic programmable logic array (DPLA). (LCB: local clock buffer; PCA: precharge AND; PCO: precharge OR; Comp.: complement.) (©2006 IEEE. Reprinted, with permission, from Reference [2].)

and are accomplished mostly by taking advantage of logic simplification and retiming.

Instruction line buffer

The instruction line buffer (ILB) uses an eight-transistor latch cell with single-end bitline for a read operation and dual datalines for a write operation (**Figure 9**). The path from the SPE main memories (four copies of 64 KB) to the ILB is one of the most critical paths in the SPE. The ILB write operation is carefully designed to accept the late arrival of data from the main memories.

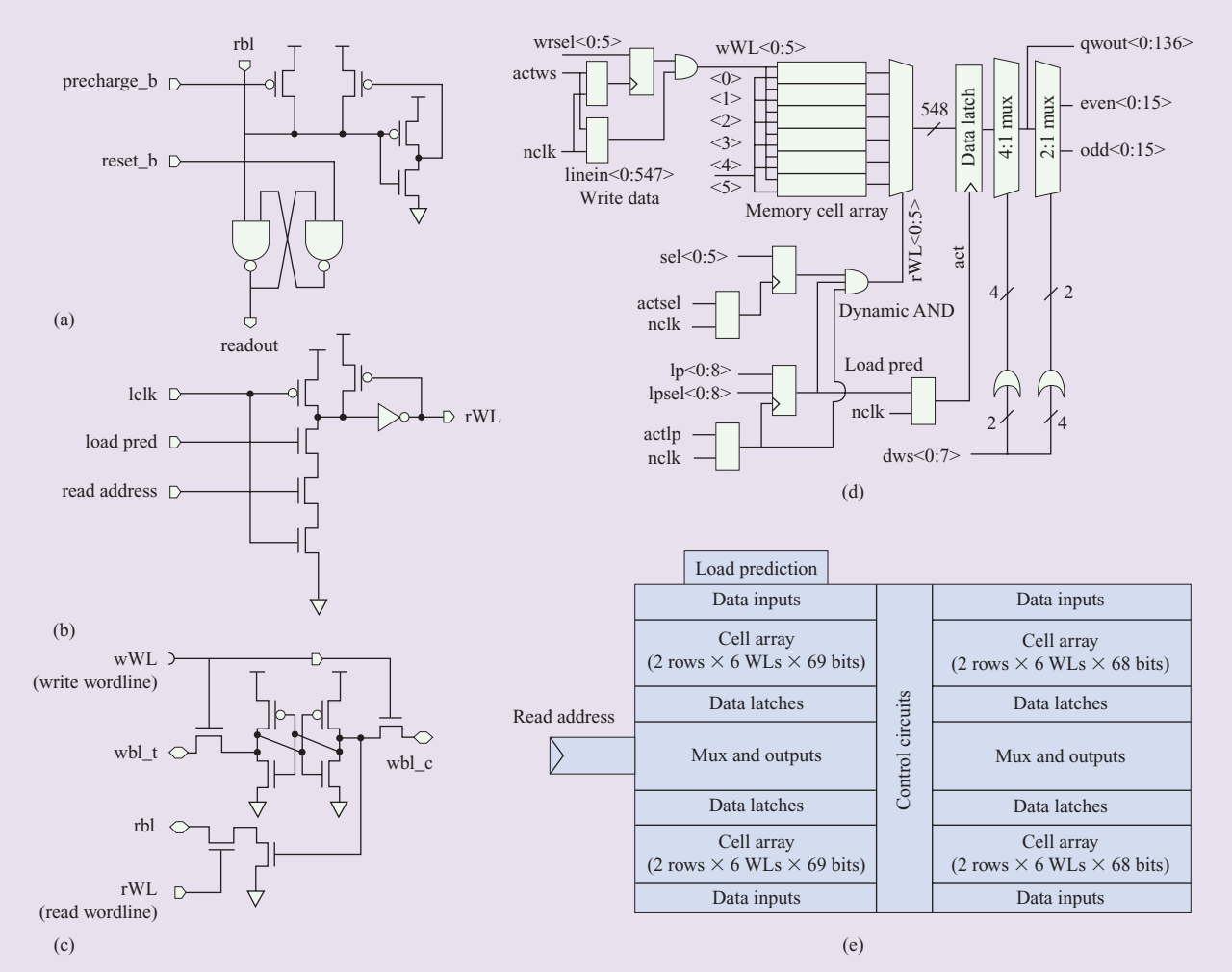


Figure 9

Instruction line buffer: (a) read circuits; (b) wordline driver; (c) memory cell; (d) block diagram; (e) floorplan. (©2006 IEEE. Reprinted, with permission, from Reference [2].)

SPE hardware measurements

The SPE has been rigorously tested and the correct operation for complicated workloads, such as three-dimensional picture rendering, has been observed. The fastest operation of a 90-nm SPE at 5.6 GHz has been observed at 1.4 V and 56°C, while the 65-nm SPE has been observed to operate at 7.3 GHz. The SPE voltage–frequency shmoo plot for a 90-nm and a 65-nm part is shown in **Figure 10**.

Conclusion

The SPE represents the middle ground between graphics processors and general-purpose processors. It is more flexible and programmable than graphics processors but has more focus on streaming workloads than general-purpose processors. The SPE competes favorably with general-purpose processors on a cycle-by-cycle basis with substantially less area and power while running many streaming and HPC algorithms. The efficiency in area and power encourages the construction of a system on a chip using multiple SPEs and offering performance many times that of competitive general-purpose processors. It is possible to address the memory latency bottleneck and improve application performance through the DMA programming model. This model provides concurrency between data access and computation while making efficient use of the available memory bandwidth. Full custom design techniques can address a challenging frequency, area, and power design point. These techniques

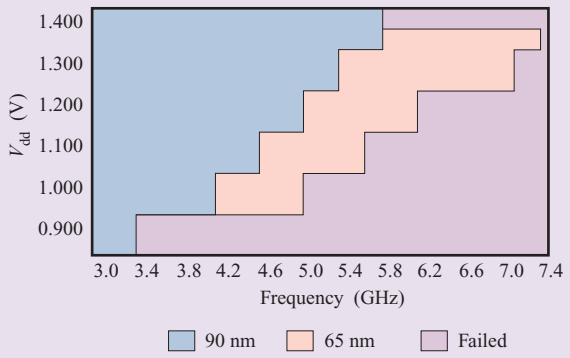


Figure 10

Voltage–frequency shmoo plot for 90-nm and 65-nm SPEs.

allow the SPE to have a shorter pipeline, occupy less area, and as a total package, dissipate less power.

Acknowledgments

The authors thank the IBM Burlington Product Engineering team, especially L. Maurice, K. O'Buckley, and M. Maurice; the software team, especially B. Minor and G. Fossum; the system test team, especially R. Berry and N. Criscolo; the convergence team; the verification team, especially S. Gupta and B. Hoang; the management team, especially L. Van Grinsven and R. Putney; and all of the SPE design team members.

*Trademark, service mark, or registered trademark of International Business Machines Corporation in the United States, other countries, or both.

**Trademark, service mark, or registered trademark of Rambus, Inc., Intel Corporation, or Cray, Inc., in the United States, other countries, or both.

†Cell Broadband Engine and PLAYSTATION are trademarks of Sony Computer Entertainment, Inc., in the United States, other countries, or both.

References

1. B. Flachs, S. Asano, S. H. Dhong, H. P. Hofstee, G. Gervais, R. Kim, T. Le, et al., "The Microarchitecture of the Synergistic Processor for a Cell Processor," *IEEE J. Solid State Circuits* **41**, No. 1, 63–70 (2006).
2. O. Takahashi, R. Cook, S. Cottier, S. H. Dhong, B. Flachs, K. Hirairi, A. Kawasumi, et al., "The Circuits and Physical Design of the Synergistic Processor Element of a Cell Processor," *Proceedings of the Symposium on VLSI Circuits*, Kyoto, Japan, 2005, pp. 20–23.
3. J. A. Kahle, M. N. Day, H. P. Hofstee, C. R. Johns, T. R. Maeurer, and D. Shippy, "Introduction to the Cell Multiprocessor," *IBM J. Res. & Dev.* **49**, No. 4/5, 589–604 (July/September 2005).
4. Synergistic Processor Unit Instruction Set Architecture, Version 1.2, IBM Corporation, Sony Computer Entertainment Corporation, and Toshiba Corporation; see *http://www.ibm.com/chips/techlib/techlib.nsf/techdocs/76CA6C7304210F3987257060006F2C44/$file/SPU_ISA_v1.2_27Jan2007_pub.pdf*.
5. V. Srinivasan, D. Brooks, M. Gschwind, P. Bose, V. Zyuban, P. N. Strenski, and P. G. Emma, "Optimizing Pipelines for Power and Performance," *Proceedings of the 35th Annual ACM/IEEE International Symposium on Microarchitecture*, Istanbul, Turkey, 2002, pp. 333–344.
6. S. H. Dhong, O. Takahashi, M. White, T. Asano, T. Nakazato, J. Silberman, A. Kawasumi, and H. Yoshihara, "A 4.8GHz Fully Pipelined Embedded SRAM in the Streaming Processor of a Cell Processor," *Proceedings of the IEEE International Solid-State Circuits Conference*, San Francisco, CA, 2005, pp. 486–487.
7. N. Mäding, J. Leenstra, J. Pille, R. Sautter, S. Büttner, S. Ehrenreich, and W. Haller, "The Vector Fixed Point Unit of the Synergistic Processor Element of the Cell Architecture Processor," *Proceedings of the Conference on Design, Automation and Test in Europe: Designers' Forum*, Munich, Germany, 2006, pp. 244–248.
8. A. E. Eichenberger, J. K. O'Brien, K. M. O'Brien, P. Wu, T. Chen, P. H. Oden, D. A. Prener, et al., "Using Advanced Compiler Technology to Exploit the Performance of the Cell Broadband Engine™ Architecture," *IBM Syst. J.* **45**, No. 1, 59–84 (2006).
9. A. E. Eichenberger, K. O'Brien, K. O'Brien, P. Wu, T. Chen, P. H. Oden, D. A. Prener, et al., "Optimizing Compiler for a Cell Processor," *Proceedings of the 14th International Conference on Parallel Architectures and Compilation Techniques (PACT)*, Saint Louis, MO, 2005, pp. 161–172.
10. S. Williams, J. Shalf, L. Oliker, S. Kamil, P. Husbands, and K. Yelick, "The Potential of the Cell Processor for Scientific Computing," *Proceedings of the 3rd Conference on Computing Frontiers*, Ischia, Italy, 2006, pp. 9–20.
11. IBM Corporation (September 2, 2002). *IBM to Build World's First Cell Broadband Engine Based Supercomputer*. Press release; see *http://www-03.ibm.com/press/us/en/pressrelease/20210.wss*.
12. H.-J. Oh, S. M. Mueller, C. Jacobi, K. D. Tran, S. R. Cottier, B. W. Michael, H. Nishikawa, et al., "A Fully Pipelined Single-Precision Floating-Point Unit in the Synergistic Processor Element of a Cell Processor," *IEEE J. Solid-State Circuits* **41**, No. 4, 759–771 (2006).
13. R. Chaudhry, D. Stasiak, S. Posluszny, and S. Dhong, "A Cycle Accurate Power Estimation Tool," *Proceedings of the Asia and South Pacific Conference on Design Automation*, Yokohama, Japan, 2006, pp. 867–870.

Received January 9, 2007; accepted for publication February 6, 2007; Internet publication August 8, 2007

Brian Flachs *IBM Systems and Technology Group, 11400 Burnet Road, Austin, Texas 78758 (flachs@us.ibm.com).* Dr. Flachs served as architect, microarchitect and unit logic leader for the SPU team. He is interested in low-latency–high-frequency processors. Dr. Flachs received a B.S.E.E. degree from New Mexico State University and M.S. and Ph.D. degrees from Stanford University.

Shigehiro Asano *Toshiba Corporation, 1, Komukai-Toshiba-Cho, Saiwai-Ku, Kawasaki 212-8582, Japan.* Mr. Asano is a Senior Research Scientist. He has an M.S. degree in information engineering from the Tokyo Institute of Technology, Japan. His research interests include parallel processing, media processors, and reconfigurable processors. Mr. Asano is a member of the IEEE and the IEEE Computer Society.

Sang H. Dhong *IBM Systems and Technology Group, 11400 Burnet Road, Austin, Texas 78758 (dhong@us.ibm.com).* Dr. Dhong received a B.S.E.E. degree from Korea University and M.S. and Ph.D. degrees in electrical engineering from the University of California at Berkeley. He joined the IBM Research Division in 1983 as a Research Staff Member and became the chief technologist of the Austin Research Laboratory in 1999. In 2000, he joined the Sony–Toshiba–IBM (STI) Design Center as one of the key leaders, primarily concentrating primarily on an 11-FO4 coprocessor design. He is an IBM Distinguished Engineer, a member of the IBM Academy of Technology, and a Fellow of IEEE. He holds more than 125 U.S. patents. Dr. Dhong has received four IBM Outstanding Innovation and Technical Achievement Awards.

H. Peter Hofstee *IBM Systems and Technology Group, STI Design Center, 11400 Burnet Road, Austin, Texas 78758 (hofstee@us.ibm.com).* Dr. Hofstee received his doctorandus degree in theoretical physics from the Rijks Universiteit Groningen, The Netherlands, in 1988, and his M.S. and Ph.D. degrees in computer science from the California Institute of Technology in 1991 and 1994, respectively. After two years on the faculty at Caltech, in 1996 he joined the IBM Austin Research Laboratory, where he participated in the design of two 1-GHz PowerPC prototypes, focusing on microarchitecture, logic design, and chip integration. In 2000 he helped start the Sony–Toshiba–IBM Design Center to design a next generation of processors for the broadband era, the Cell/B.E. processor. Dr. Hofstee is a member of the Cell/B.E. Architecture team, and he is the Chief Architect of the synergistic processor in the Cell/B.E. processor. He was elected to the IBM Academy of Technology in 2004.

Gilles Gervais *IBM Systems and Technology Group, 11400 Burnet Road, Austin, Texas 78758.* Mr. Gervais received a B.S. degree in electrical engineering from Drexel University. He joined the IBM Federal Systems Division in 1982, moving to the IBM Austin Research Laboratory in 1994. He was the design leader for the Altivec engine of the Apple G5 processor. Mr. Gervais is currently the manager of the SPE verification and logic design teams for the Cell/B.E. processor.

Roy Kim *IBM Systems and Technology Group, 11400 Burnet Road, Austin, Texas 78758 (roykim@us.ibm.com).* Mr. Kim received a B.S. degree in electrical engineering from the State University of New York at Stony Brook and an M.S. degree in computer engineering from Syracuse University. He has worked on many hardware development projects in chip, card, and systems design. Mr. Kim most recently worked on the Cell/B.E. processor design at the STI Design Center.

Tien Le *IBM Systems and Technology Group, 11400 Burnet Road, Austin, Texas 78758.* Mr. Le served as the verification leader for the SMF unit. He received a B.S. degree in electrical and electronics engineering, mathematics, and computer science and an M.S. degree in mathematics, both from California State Polytechnic University. Mr. Le joined the STI team in 2002 and is working on the follow-on project.

Peichun Liu *IBM Systems and Technology Group, 11400 Burnet Road, Austin, Texas 78758.* Mr. Liu received a B.S.E.E. degree from the National Taiwan Ocean University and an M.S.E.E. degree from the University of Texas at Arlington. He joined IBM in 1991 and worked on POWER3*, POWER4*, and Cell/B.E. microprocessor design projects.

Jens Leenstra *IBM Systems and Technology Group, IBM Deutschland Entwicklung GmbH, Schoenaicherstrasse 220, D-71032 Boeblingen, Germany (leenstra@de.ibm.com).* Dr. Leenstra received an M.S. degree from the University of Twente, The Netherlands, and a Ph.D. degree from the University of Eindhoven, The Netherlands, joining IBM at Boeblingen in 1994. He has worked in several areas of the Server Group and the Systems and Technology Group development organization, including logic design and verification of I/O chips, multiprocessor system verification of the G2 and G3 mainframe computers, the Cell/B.E. processor SPEs, and the POWER6* VMX unit. He is currently working on next-generation IBM microprocessors. Dr. Leenstra's current interests focus on computer architecture, high-frequency design, low power, and design for testability.

John S. Liberty *IBM Systems and Technology Group, 11501 Burnet Road, Austin, Texas 78758.* Mr. Liberty received B.S. and M.S. degrees in electrical engineering from North Carolina State University. He is an advisory engineer in the Sony–Toshiba–IBM (STI) Design Center, responsible for helping architect and develop the Cell Broadband Engine. His main focus within the Cell/B.E. processor is the SPU SIMD processor. He was a leader in designing the SPU channel interface and the SPU inherent security architecture. Before working on the Cell/B.E. processor, he was a designer working on graphic processing units. Mr. Liberty holds four patents, with 12 patents pending, he has co-authored three technical articles.

Brad Michael *IBM Systems and Technology Group, 11400 Burnet Road, Austin, Texas 78758.* Mr. Michael received a B.S. degree in computer engineering from Iowa State University in 1989. He joined IBM that same year and is now involved in microprocessor logic design in the STI Design Center.

Hwa-Joon Oh *IBM Systems and Technology Group, 11400 Burnet Road, Austin, Texas 78758 (hjoh@us.ibm.com).* Dr. Oh received B.S and M.S. degrees from Yonsei University, Korea, and a Ph.D. degree from Michigan State University. In 1998, he joined the IBM Austin Research Laboratory, where he worked on POWER4 microprocessor design. He is currently working with the STI Design Center, where he is involved in

architecture, logic, circuits, and physical implementations of Cell/B.E. microprocessor. His main research area is artificial neural network, SRAM and DRAM design, and broadband microprocessor design. Dr. Oh has authored or co-authored several journal papers and patents.

Silvia Melitta Mueller *IBM Systems and Technology Group, IBM Deutschland Entwicklung GmbH, Schoenaicherstrasse 220, D-71032 Boeblingen, Germany (smm@de.ibm.com)*. Dr. Mueller is a Senior Technical Staff Member in the high-performance processor design team. She received B.S. degrees in mathematics and computer science, an M.S. degree in mathematics, and a Ph.D. degree in computer science from the University of Saarland, Germany. Dr. Mueller held a postdoctoral position at Berkeley and spent two short sabbaticals at the computer science department of the Massachusetts Institute of Technology. In 1998, she became a Privatdozent at the computer science department of the University of Saarland and still holds a teaching assignment there. Dr. Mueller joined IBM at Boeblingen in late 1999. From 2001 to 2003 she was on international assignment to IBM Austin, joining the STI Design Center developing the Cell/B.E. processor. She led the development of the floating-point units for the Cell/B.E. processor and for the POWER6 VMX.

Osamu Takahashi *IBM Systems and Technology Group, 11400 Burnet Road, Austin, Texas 78758 (osamu@us.ibm.com)*. Dr. Takahashi is a Senior Technical Staff Member and manager of the circuit design team for the SPE developed at the STI Design Center. He received a B.S. degree in engineering physics and an M.S. degree in electrical engineering from the University of California at Berkeley, as well as a Ph.D. degree in computer and mathematical sciences from Tohoku University, Japan.

Koji Hirairi *Sony Corporation, Atsugi Technology Center, 4-14-1 Asahi-cho, Atsugi-shi, Kanagawa 243-0014 Japan.* Mr. Hirairi was in charge of research and development of the custom datapath macro at the STI Design Center. He received B.S. and M.S. degrees in electronics engineering from Chuo University, Japan and is currently involved in DRAM and I/O circuit design.

Atsushi Kawasumi *Toshiba Corporation, 580-1, Horikawa-Cho, Saiwai-Ku, Kawasaki, 212-8520, Japan.* Mr. Kawasumi received B.S. and M.S. degrees in pure and applied sciences from the University of Tokyo. He joined the Toshiba Semiconductor Device Engineering Laboratory in 1991, where he engaged in research and development of ultrahigh-speed SRAM. He moved to Toshiba America Electronic Components, Inc., in 2002 to join the STI Cell/B.E. processor project. Mr. Kawasumi is currently with the Center for Semiconductor Research and Development, Semiconductor Company, Toshiba Corporation.

Hiroaki Murakami *Toshiba Corporation, 580-1, Horikawa-Cho, Saiwai-Ku, Kawasaki 212-8520, Japan.* Mr. Murakami received a B.S. degree in mathematical engineering from the University of Osaka Prefecture, Japan. He joined the Toshiba Corporation in 1982 and worked on developing RISC and Cell/B.E. processors. He is currently with the Broadband System LSI Development Center, Semiconductor Company, Toshiba Corporation.

Hiromi Noro *Toshiba America Electronic Components, 9696 N. Mopac Highway, Austin, Texas 78759.* Mr. Noro is a design engineering manager. He received a B.S. degree in electronics communication engineering and an M.S. degree in electrical engineering, both from Tokai University, Japan. Mr. Noro's research interests include the design of various types of high-speed custom circuits for microprocessors and of high-speed memory (SRAM, register file, and content-addressable memory) for microprocessors.

Shoji Onishi *IBM Engineering and Technology Service, IBM Japan, 800 Ichimiyake, Yasu-shi, Shiga 520-2392, Japan.* In 1990, Mr. Onishi joined the IBM Yasu Technology Application Laboratory, where he was involved in the design of hard disk controllers, SRAM, and DRAM. He was assigned to the IBM Austin Research Laboratory in 1996 to design the 1-GHz eDRAM macro. Mr. Onishi has been involved in the research and development of high-speed custom array macros of the SPE in the STI Design Center since 2001.

Juergen Pille *IBM Systems and Technology Group, IBM Deutschland Entwicklung GmbH, Schoenaicherstrasse 220, D-71032 Boeblingen, Germany (pillej@de.ibm.com)*. Mr. Pille is a Senior Technical Staff Member. He received an M.S. degree in microelectronics from Hanover University, Germany. Since joining IBM in 1990, he has worked on various microprocessor designs, focusing on arrays and circuits, most recently a first-generation Cell/B.E. processor. Mr. Pille is currently working on Cell/B.E. processor core array designs.

Joel Silberman *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598 (lber@us.ibm.com)*. Dr. Silberman is a Research Staff Member. He received a Ph.D. in electrical engineering from Stanford University in 1986 and joined the IBM Research Division that same year. He has worked on numerous research and product-oriented microprocessor design projects focusing on high-speed and low-power designs. Dr. Silberman's current work is concerned with tradeoffs in microarchitecture and circuit performance of advanced cache memory and instruction issue logic.

Suksoon Yong *IBM Systems and Technology Group, 11400 Burnet Road, Austin, Texas 78758.* Mr. Yong received a B.S. degree in electronics engineering from Korea University and an M.S. degree from the University of Michigan. Since joining IBM in 1990, he has been working on ASIC and microprocessor logic designs and system-on-chip ASIC integration and logic design. Mr. Yong is currently the SPE timing leader in the STI Design Center.

Akiyuki Hatakeyama *Sony Computer Entertainment, Inc., Tokyo, Japan.* Mr. Hatakeyama received B.S. and M.S. degrees in computer science from the Kanagawa Institute of Technology, Japan. He is currently involved in developing the Sony PLAYSTATION† system.

Yukio Watanabe *Toshiba Corporation, 580-1, Horikawa-Cho, Saiwai-Ku, Kawasaki 212-8520, Japan.* Mr. Watanabe received a B.E. degree in electronic engineering from Tohoku University, Japan, in 1993, and an M.E. degree in information science from the

Graduate School of Information Sciences, Tohoku University, Japan. He joined the Toshiba Corporation in 1995, where he has been engaged in the design and development of high-performance microprocessors. From 2001 to 2005 he was assigned to the STI Design Center where he was engaged in the development of the Cell/B.E. processor.

Naoka Yano *Toshiba Corporation, 580-1, Horikawa-Cho, Saiwai-Ku, Kawasaki 212-8520, Japan.* Ms. Yano received a B.S. degree in pure and applied sciences from the University of Tokyo. She joined the Semiconductor Device Engineering Laboratory, Toshiba Corporation, in 1991. She was engaged in the research and development of high-performance microprocessors. In 2001, she moved to Toshiba America Electronic Components, Inc., and was involved in the development of the Cell/B.E. processor. Ms. Yano is currently with the Broadband System LSI Development Center, Semiconductor Company, Toshiba Corporation.

Daniel A. Brokenshire *IBM Systems and Technology Group, 11400 Burnet Road, Austin, Texas 78758 (brokensh@us.ibm.com).* Mr. Brokenshire is a Senior Technical Staff Member with six years of experience in the Cell/B.E. Processor Design Center. He currently serves as a senior member of the IBM Cell Processor Systems Enablement team working on the Cell/B.E. Software Development Kit. His responsibilities include the development of programming standards, language extensions, reusable software libraries, and software documentation. Mr. Brokenshire received a B.S. degree in computer science and B.S. and M.S. degrees in electrical engineering, all from Oregon State University. Prior to his work on the Cell/B.E. processor, he enjoyed a productive career developing 3D graphics products for Tektronix, Inc., and IBM.

Mohammad Peyravian *IBM Systems and Technology Group, 3039 Cornwallis Road, P.O. Box 12195, Research Triangle Park, North Carolina 27709 (peyravn@us.ibm.com).* Dr. Peyravian received a Ph.D. degree in electrical engineering from the Georgia Institute of Technology. He is a network processor architect whose interests include networking, network processors, cryptography, and security. Dr. Peyravian has published more than 40 journal and conference papers; he holds more than 30 patents in networking, cryptography, and security.

VanDung To *IBM Systems and Technology Group, 11400 Burnet Road, Austin, Texas 78758.* Ms. To received a B.A. degree in computer science from Rice University. She is currently working on an M.B.A. degree at the University of Texas at Austin. Ms. To joined IBM in 2001 and is now working on Cell/B.E. processor software development.

Eiji Iwata *Sony Computer Entertainment, Inc., Tokyo, Japan (Eiji.Iwata@jp.sony.com).* Mr. Iwata received an M.E. degree in information systems engineering from Kyushu University, Japan. He joined Sony Corporation, Tokyo, in 1991. In 1996, he was a visiting researcher at the Hydra project in the Computer Science Laboratory at Stanford University. Returning to Sony, he worked on the development of a chip multiprocessor for media applications. He joined the STI Design Center in 2001, and he currently works on a project promoting the Cell/B.E. processor.