The Microarchitecture of the Synergistic Processor for a Cell Processor

Brian Flachs, Shigehiro Asano, *Member, IEEE*, Sang H. Dhong, *Fellow, IEEE*, H. Peter Hofstee, *Member, IEEE*, Gilles Gervais, Roy Kim, Tien Le, Peichun Liu, Jens Leenstra, John Liberty, Brad Michael, Hwa-Joon Oh, Silvia Melitta Mueller, Osamu Takahashi, *Member, IEEE*, A. Hatakeyama, Yukio Watanabe, Naoka Yano, Daniel A. Brokenshire, Mohammad Peyravian, Vandung To, and Eiji Iwata

***Abstract*—This paper describes an 11 FO4 streaming data processor in the IBM 90-nm SOI-low-k process. The dual-issue, four-way SIMD processor emphasizes achievable performance per area and power. Software controls most aspects of data movement and instruction flow to improve memory system performance and core performance density. The design minimizes instruction latency while providing for fine grain clock control to reduce power.**

***Index Terms*—Cell, DSP, RISC, SIMD, SPE, SPU.**

I. Introduction

INCREASING thread level parallelism, data bandwidth, memory latency, and leakage current are important drivers for new processor designs, such as Cell. Today's media-rich application software is often characterized by multiple light weight threads and software pipelines. This trend in software design favors processors that utilize these threads to drive the improved data bandwidths over processors designed to accelerate a single thread of execution by taking advantage of instruction level parallelism. Memory latency is a key limiter to processor performance. Modern processors can lose up to 4000 instruction slots while they wait for data from main memory. Previous designs emphasize large caches and reorder buffers, first to reduce the average latency and second to maintain instruction throughput while waiting for data from cache misses. However, these hardware structures have difficulty scaling to the sizes required by the large data structures utilized by media rich software. Transistors oxides are now a few atomic levels thick and the channels are extremely narrow. These features are very good for improving transistor performance and increasing transistor density, but tend to increase leakage current. As processor performance becomes power limited, leakage current becomes an important performance issue. Since leakage is proportional to area, processor designs need to extract more performance per transistor.

Manuscript received April 15, 2005; revised August 31, 2005.

B. Flachs, S. H. Dhong, H. P. Hofstee, G. Gervais, R. Kim, T. Le, P. Liu, J. Liberty, B. Michael, H.-J. Oh, O. Takahashi, D. A. Brokenshire, and V. To are with the IBM Systems and Technology Group, Austin, TX 78758 USA (e-mail: flachs@us.ibm.com).

S. Asano and Y. Watanabe are with Toshiba America Electronic Components, Austin, TX 78717 USA.

N. Yano is with the Broadband System LSI Development Center, Semiconductor Company, Toshiba Corporation, Kawasaki, Japan.

J. Leenstra and S. M. Mueller are with the IBM Entwicklung GmbH, Boeblingen 71032, Germany.

A. Hatakeyama and E. Iwata are with Sony Computer Entertainment, Austin, TX 78717 USA.

M. Peyravian is with IBM Microelectronics, Research Triangle Park, NC 27709 USA.

Digital Object Identifier 10.1109/JSSC.2005.859332

II. Architecture

The Cell processor is a heterogeneous shared memory multiprocessor [2]. It features a multi-threaded 64 bit POWER processing element (PPE) and eight synergistic processing elements (SPE). Performance per transistor is the motivation for heterogeneity. Software can be divided into general purpose computing threads, operating system tasks, and streaming media threads and targeted to a processing core customized for those tasks. For example, PPE is responsible for running the operating system and coordinating the flow of the 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.

The synergistic processor element (SPE) is the first implementation of a new processor architecture designed to accelerate media and streaming workloads. The architecture aims to improve the effective memory bandwidth achievable by applications by improving the degree to which software can tolerate memory latency. SPE provides processing power needed by streaming and media workloads through four-way SIMD operations, dual issue and high frequency.

Area and power efficiency are important enablers for multicore designs that take advantage of parallelism in applications, where performance is power limited. Every design choice must trade off the performance a prospective feature would bring versus the prospect of omitting the feature and devoting the area and power toward higher clock frequency or more SPE cores per Cell processor chip. Power efficiency drives a desire to replace event and status polling performed by software during synchronization with synchronization mechanisms that allow for low power waiting.

Fig. 1 is a diagram of the SPE architecture's major entities and their relationships. Local store is a private memory for SPE instructions and data. The synergistic processing unit (SPU) core is a processor than runs instructions from the local store and can read or write the local store with its load and store instructions. The direct memory access (DMA) unit transfers data between local store and system memory. The DMA unit is programmable by SPU software via the channel unit. The channel unit is a message passing interface between the SPU core and

the DMA unit and the rest of the Cell processing system. The channel unit is accessed by the SPE software through channel access instructions.

The SPU core is a SIMD RISC-style processor. All instructions are encoded in 32 bit fixed length instruction formats and there are no hard to pipeline instructions. SPU features 128 general purpose registers. These registers are used by both floating point and integer instructions. The shared register file allows the highest level of performance for various workloads with the smallest number of registers. 128 registers allow for loop unrolling which is necessary to fill functional unit pipelines with independent instructions. Most instructions operate on 127 bit wide data. For example, the floating point multiply add instruction operates on vectors of four 32 bit single precision floating point values. Some instructions, such as floating point multiply add, consume three register operands and produce a register result. SPE includes instructions that perform single precision floating point, integer arithmetic, logicals, loads, stores, compares and branches in addition to some instructions intended to help 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's architecture reduces area and power while facilitating improved performance by requiring software solve "hard" scheduling problems such as data fetch and branch prediction. Because SPE will not be running the operating system, SPE concentrates on 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 is serviced by the local store (LS). LS is a private memory, not a cache, and does not require tag arrays or backing store. Loads, stores and instruction fetch complete with fixed delay and without exception, greatly simplifying the core design and providing predictable real-time behavior. This design reduces the area and power of the core while allowing for higher frequency operation.

Data is transferred to and from the LS in 1024 bit lines by the SPE DMA engine. The SPE DMA engine allows SPE software to schedule data transfers in parallel with core execution. Fig. 2 is a time line that illustrates how software can be divided into coarse grained threads to overlap data transfer and core computation. As thread 1 finishes its computation it initiates 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 gets 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 from system memory the next data set. Thread 2 then branches back to thread 1. Techniques like double buffering and course grained multithreading allow software to overcome memory latency to achieve high memory bandwidth and improve 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

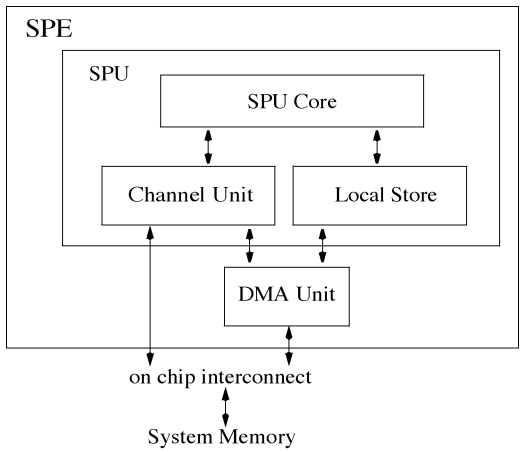


Fig. 1. SPE architecture diagram.

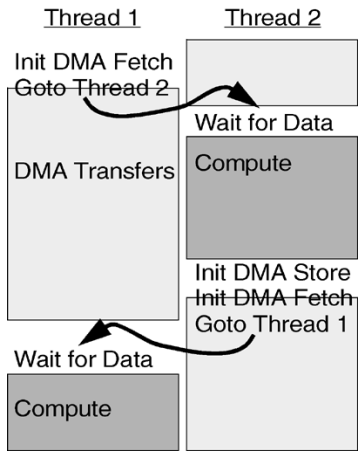


Fig. 2. Example time line of concurrent computation and memory access.

instructions. These commands can perform scatter-gather operations from system memory or setup a complex set of status reporting and notification mechanisms. Not only can software achieve much higher bandwidth through the DMA engine than it could with a hardware prefetch engine, a much higher fraction of the bandwidth is useful data than would occur with the prefetch engine design.

The channel unit is a message passing interface between the SPU core and the rest of the system. Each device is allocated one or more channels through which messages can be sent to or from the SPU core. SPU software sends and receives messages with the write and read channel instructions. Channels have capacity which allows for multiple messages to be queued. Capacity allows 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 will stall 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.

The SPE has separate 8-byte-wide inbound and outbound data busses. The DMA engine supports transfers requested locally by the SPE through the SPE request queue and requested externally either via the external request queue or external bus requests through a window in the system address space. The

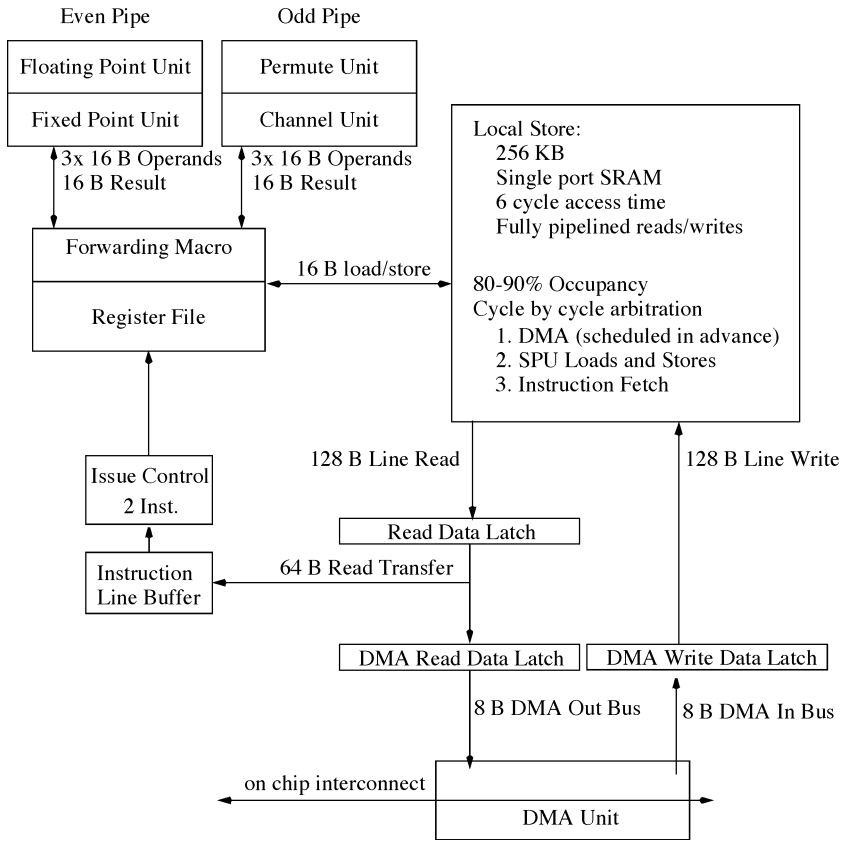


Fig. 3. SPE organization.

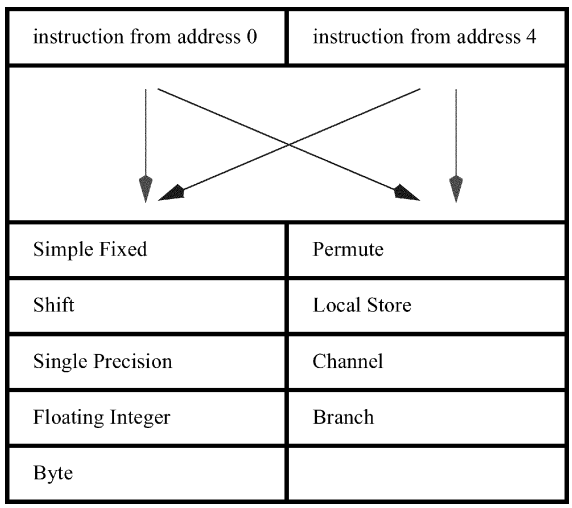
SPE request queue supports up to 16 outstanding transfer requests. Each request can transfer up to 16 KB of data to or from the local address space. DMA request addresses are translated by the MMU before the request is sent to the bus. Software can check or be notified when requests or groups of requests are completed.

The SPE programs the DMA engine through the Channel Interface. The channel interface is a message passing interface intended to overlap I/O with data processing and minimize power consumed by synchronization. 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 which can be configured as blocking or nonblocking.

III. Microarchitecture

Fig. 3 shows how the SPE is organized and the key bandwidths (per cycle) between units. Instructions are fetched from the LS in 32 4-byte groups when 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). 3.5 fetched lines are stored in the ILB [1]. A half line holds instructions while they are sequenced into the issue logic while another line holds the single-entry software-managed branch target buffer (SMBTB) and two lines are used for inline prefetching. Instructions are sent, two at a time, from the ILB to the issue control unit.

TABLE I
Dual Issue Unit Assignments



The SPE issues and completes all instructions in program order and does not reorder or rename its instructions. Although the SPE is not a VLIW processor, it does feature a VLIW like dual issue feature and can issue up to two instructions per cycle to nine execution units organized into two execution pipelines, as shown in Table I. Instruction pairs can be issued if the first instruction (from an even address) will be 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

TABLE II
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, byte sum	Even	3	4
Permute	Quadword shifts, rotates, gathers, shuffles as well as reciprocal estimate	Odd	3	4
Local Store	Load and store	Odd	6	6
Channel	Channel Read/Write	Odd	5	6
Branch	Branches	Odd	3	4

NOP 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 either from the register file or 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, 128 entries of 128 bits each and 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 until they are committed and available from the register file. These results are read from 6 forward-macro read-ports and distributed to the units in one cycle.

Loads and stores transfer 16 bytes of data between the register file and the local store. The LS is a six-cycle, fully pipelined, single-ported, 256 KB SRAM [3]. The LS is shared between the SPE load/store unit, the SPE instruction fetch unit and the DMA unit. Several workloads exhibit 80%–90% LS occupancy. In order 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 the local store 128 bytes in a single cycle, providing lots of bandwidth with relatively little interference to the SPE loads and stores. Load and stores are second in priority and wait in the issue stage for an available LS cycle. Instruction fetch accesses the local store when it is otherwise idle, again with 128 byte accesses to minimize the chances of performance loss due to instruction run out.

Table II details the eight execution units. Simple fixed point [4], floating point [5] and load results are bypassed directly from the unit output to input operands to reduce result latency. Other results are sent to the forward macro from where they are distributed a cycle later. Fig. 4 is a pipeline diagram for the SPE

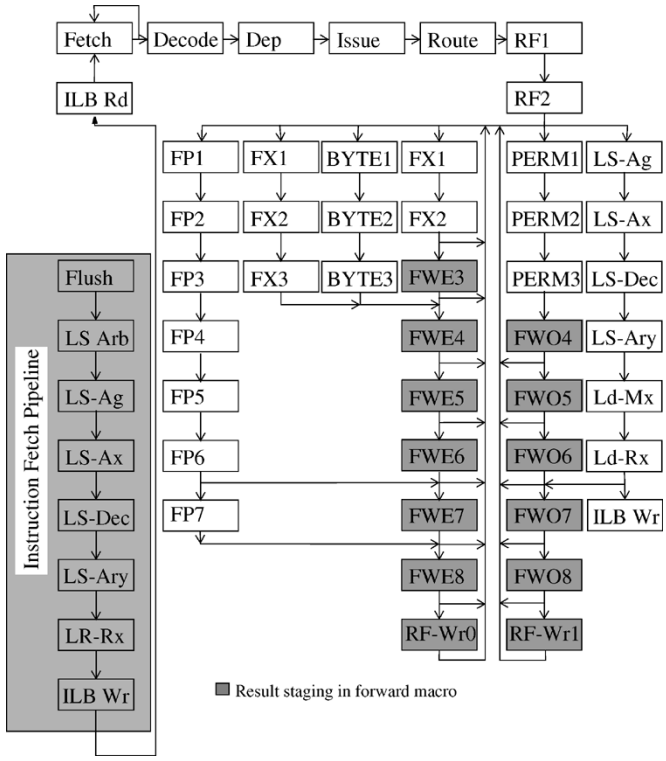


Fig. 4. SPE pipeline diagram.

that 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 those 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.

In order to save area and power, SPE omits hardware branch prediction and branch history tables. However, mispredicted branches flush the pipelines and cost 18 cycles so it is important that software employ mispredict avoidance techniques.

TABLE III
SPE APPLICATION PERFORMANCE

Benchmark	Performance/ 1B Cycles	IPC
AES CBC decrypt 128 bit key	470 Mbytes	1.85
RSA private key Encrypt/Decrypt (1024b)	245 encrypt-decrypt	1.25
HDTV MPEG decode (MP@HL-18Mb/s)	22 Frames	0.64
Triangle Lighting (3D transform with perspective division, one infinite light source, color conversion and packing)	80 M Triangles	1.51
Single Precision LINPACK (8k x 8k)	7.3 G Floating Point Operations	1.72

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. The SMBTB holds a single branch target, and is loaded by software with a load branch target buffer instruction. When a branch is identified and its target is loaded into the SMBTB, the SPE can execute the taken branch in a single cycle.

IV. IMPLEMENTATION

Performance on several important workloads is shown in Table III. These workloads are written in C using intrinsics for the SIMD data types and DMA transfers. Linpack, AES and Triangle Lighting achieve performance very close to the SPE peak of two instructions per cycle (IPC). The Linpack data is especially impressive in that the DMA required to move the data into and out of the local store is entirely overlapped with execution. The other workloads execute out of SPU local store. Triangle Lighting is a computationally intensive application, that has been unrolled four times and 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 thus be impractical. These benchmarks show that 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 ISA, the large register file and the relatively short execution pipelines that the compiler can easily schedule code for the SPE. These benchmarks also show that the DMA programming model can over come memory latency and allows the SPU core to achieve peak performance rather than wait for needed data. Even with a very simple microarchitecture and very small area, SPE can compete on a cycle-by-cycle basis with more complex cores on streaming workloads.

Fig. 5 is a photo of the 2.54×5.81 mm^2 SPE, in 90-nm SOI. SPE has 21 M transistors. Roughly 14 M of these transistors are in arrays while the remaining 7 M transistors are in logic. The bus interface unit (BIU) is under the on chip interconnect that runs horizontally at the bottom of the SPE. Just above the BIU is the DMA unit and its memory management unit (MMU). SPU core occupies the upper three quarters of the floor plan. The LS is on the right hand side, built from four 64 KB arrays. The data path is on the left side, starting at the bottom with the register file. The odd pipeline permute unit is between the register file and the forward macro. The even pipeline is at the top of the data flow. The four way SIMD nature of the SPE is clearly visible in the data flow. Instruction sequencing and control accounts for less than 10% of the total area and is generally located between the data flow and the local store. The 64 B LS read bus runs down over the LS ending in the channel unit under the LS stack and in the ILB directly to the left of the bottom most LS array. Instruction processing then flows back toward the middle of the dataflow as instructions and operand are distributed to the execution units.

Table IV is a voltage versus frequency shmoo that shows SPE active power and die temperature while running a single precision intensive lighting and transformation workload that averages 1.4 IPC. Limiting pipeline depth also helps minimize power. The shmoo shows SPE dissipates 1 W at 2 GHz, 2 W at 3 GHz and 4 W of active power at 4 GHz. Active power is the amount of power consumed by switching logic and array circuits. For reference, this SPE dissipated about 1.7 W through leakage current and 1.3 W in its clock grid at the 1.2 V, 2 GHz operating condition. Although the shmoo shows function up to 5.2 GHz, separate experiments show that at 1.4 V and 56 °C, the SPE can achieve up to 5.6 GHz.

The shmoo clearly shows the SPE to be a high frequency processor. In terms of the time required for an inverter to drive four copies of itself the SPE is an 11 FO4 design. This allows between four and eight stages of logic per cycle. Recall that SPE pipelines are similar in depth to those of 20 FO4 processors. There are many ingredients required to achieve these results. The architecture eliminates many hard instructions and functions. The microarchitecture reduces control complexity with reduced latency and early instruction commit. Careful floor planning minimizes the substantial effect of wires on cycle time. Wire lengths are shortened by high density layout. Selected dynamic circuits are used in the instruction sequencing logic and forward macro to maintain short instruction issue pipelines. The SPE implementation makes use of a rich latch

TABLE IV
VOLTAGE/FREQUENCY SCHMOO

Vdd (Volts) \ Freq (GHz)	2	2.2	2.4	2.6	2.8	3	3.2	3.4	3.6	3.8	4	4.2	4.4	4.6	4.8	5	5.2
1.3	48C 4W	49C 4W	50C 5W	50C 6W	51C 6W	52C 7W	53C 7W	54C 7W	55C 8W	56C 8W	57C 9W	58C 9W	59C 10W	60C 10W	61C 10W	63C 11W	61C
1.2	39C 2W	39C 3W	40C 3W	41C 4W	42C 4W	42C 4W	43C 5W	44C 5W	45C 5W	45C 5W	46C 6W	47C 6W	47C 7W	48C	49C		
1.1	32C 2W	33C 2W	33C 3W	35C 3W	35C 3W	36C 3W	36C 4W	37C 4W	37C 4W	38C 4W	38C 4W	39C	39C				
1	28C 2W	28C 2W	29C 2W	29C 2W	30C 2W	30C 3W	30C 3W	31C 3W	31C 3W	31C 3W	32C						
0.9	25C 1W	26C 1W	26C 1W	26C 2W	27C 2W	27C 2W	27C	Failed									

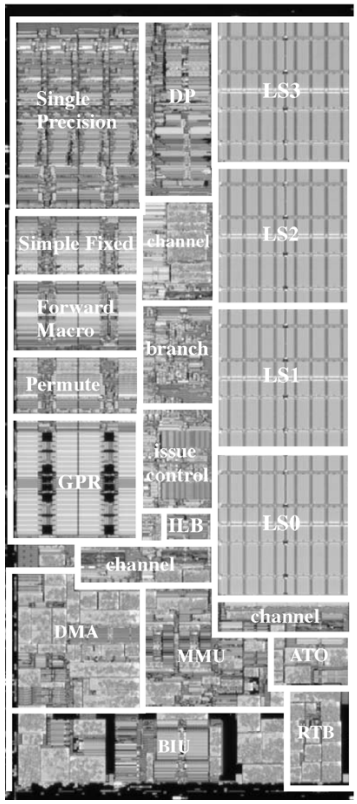


Fig. 5. SPE die photo.

library that has options for low insertion delay and embedded functions to reduce latch overhead.

Power efficiency starts in the architecture which requires translation only for DMA commands and provides a low power alternative to polling. Microarchitectural decisions were made in the context of logic driven power estimation tools. Limiting pipeline depths and the elimination of tag array access from the execution of speculative instructions both reduce power. High density layout reduces wire cap and the required driver sizes. SPE features fine grain clock control and mostly static circuits in the execution units. These design choices result in active power for an idle SPU to be 1/5 of the active power required for a heavy work load and allow for impressive performance per watt.

V. CONCLUSION

The SPE represents a 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. SPE competes favorably with general purpose processors on a cycle by cycle basis with substantially less area and power. The efficiency in area and power encourages the construction of a system on a chip including multiple SPEs and many times the performance of competitive general purpose processors. It is possible to address the memory latency wall 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.

ACKNOWLEDGMENT

The authors thank S. Gupta, B. Hoang, N. Criscolo, M. Pham, D. Terry, M. King, E. Rushing, B. Minor, L. Van Grinsven, R. Putney, and the rest of the SPE design team.

REFERENCES

[1] O. Takahashi *et al.*, "The circuits and physical design of the streaming processor of a CELL processor," presented at the Symp. VLSI Circuits, Kyoto, Japan, 2005.

[2] D. Pham *et al.*, "The design and implementation of a first-generation CELL processor," in *IEEE ISSCC Dig. Tech. Papers*, 2005, pp. 184–186.

[3] S. H. Dhong *et al.*, "A fully pipelined embedded SRAM in the streaming processing unit of a CELL processor," in *IEEE ISSCC Dig. Tech. Papers*, 2005, pp. 486–488.

[4] J. Leenstra *et al.*, "The vector fixed point unit of the streaming processor of a CELL processor," presented at the Symp. VLSI Circuits, Kyoto, Japan, 2005.

[5] H. Oh *et al.*, "A fully-pipelined single-precision floating point unit in the streaming processing unit of a CELL processor," presented at the Symp. VLSI Circuits, Kyoto, Japan, 2005.

Brian Flachs received the B.S.E.E. degree in 1988 from New Mexico State University, and the M.S. and Ph.D. degrees in 1994 from Stanford University.

He served as architect, microarchitect, and unit logic lead for the SPU Team and is interested in low latency high frequency processors. Previously, serving as microarchitecture for IBM Austin Research Laboratories' 1 GHz PowerPC Project. His research interests include computer architecture, image processing, and machine learning.

Shigehiro Asano (M'96) received the M.S. degree in information engineering from Waseda University, Tokyo, Japan.

He is a Senior Research Scientist at Toshiba Corporation, Austin, TX. His research interests include parallel processors, media processors, and reconfigurable processors. He is a member of the IEEE Computer Society and the Information Processing Society of Japan (IPSJ).

Sang H. Dhong (M'76–SM'99–F'01) received the B.S.E.E. degree from Korea University, Seoul, Korea, and the M.S. and Ph. D. degrees in electrical engineering from the University of California at Berkeley, CA.

He joined IBM's Research Division in Yorktown Heights, NY, in 1983 as a research staff member where he was involved in the research and development of silicon processing technology, mainly bipolar devices and reactive-ion etching (RIE). From 1985 to 1992, he was engaged in research and development of DRAM cell structures, architectures, and designs, spanning over five generations of IBM DRAMs, from 1 Mb DRAMs to 256 Mb DRAMs. The key contributions in this area are high-speed DRAMs, low-power DRAMs, and NMOS-access transistor trench DRAM cells.

After spending three years in development of one of IBM's PowerPC microprocessors as a Branch/Icache circuit team leader of 15 designers, he worked on a simple but fast processor core based on the PowerPC architecture and on high-speed embedded DRAM (eDRAM) in the Austin Research Lab of the IBM research division, leading, managing, and growing the high-performance VLSI design group to a peak of 25 people from 1995 to 1999. The work resulted in setting a major milestone for the microprocessor industry by prototyping1-GHz PowerPC processors. Also, the work on the high-speed eDRAM provided the justification for the logic-based eDRAM foundry/ASIC technology offering by IBM as well as the design basis for eDRAM macros of DRAM-like density with SRAM-like high speed.

Since becoming the chief technologist of the Austin Research Lab in 1999, he worked on three areas: fast low-power embedded PowerPC, super-pipelined multi-gigahertz PowerPC servers, and high-speed eDRAM. In 2000, he joined the Sony-Toshiba-IBM (STI) design center as one of the key leaders, primarily concentrating on a 11-FO4 coprocessor design, streaming process (SPE). As the partition leader of SPE team, he defined, executed, and delivered the technology, circuits and latch styles, floor plan, and basic lower-power micro-architecture and led technically a multi-discipline team of 70 or more engineers. Currently, he is the chief hardware engineer/SPE partition lead, being responsible for productization of BE chip from the STI center side. In this role, his major focus is on power-frequency yield tradeoff, interacting and directing manufacturing, PE, and design teams in a matrix organization of more than 100 engineers.

He is an IBM Distinguished Engineer, a member of IBM Academy of Technology, and a Fellow of IEEE. He holds more than 125 U.S. patents as well as many technical publications and has received four Outstanding Innovation/Technical Achievement Awards from IBM.

H. Peter Hofstee (M'96) received the "Doctorandus" degree in theoretical physics from the University of Groningen, The Netherlands, in 1989, and the M.S. and Ph.D. degrees in computer science from the California Institute of Technology (Caltech), Pasadena, in 1991 and 1995, respectively.

After being on the faculty at Caltech for two years, he joined the IBM Austin Research Laboratory in 1996, where he worked on high-frequency processors. In 2000, he helped develop the concept for the Cell Broadband Engine processor. He is currently the Chief Scientist for the Cell Broadband Engine processor and Chief Architect of the synergistic processor element in Cell. His focus is on future processor and system architectures and the broader use of the Cell.

Gilles Gervais received the B.S. degree in electrical engineering from Drexel University, Philadelphia, PA, in 1982.

He joined IBM Federal Systems Division in Owego, NY, in 1982 working on test equipment. In 1984, he transfered to IBM Endicott, NY, to work on IO subsystem design for IBM mainframes. He joined the Somerset design center in Austin, TX, in 1994 working on the design of the 604 family of PowerPC products. He worked as Design Lead of the Altivec engine of the Apple G5 processor in 1999 and he is currently the Manager of the SPE verification and logic design teams on the Cell processor developed at the Sony/Toshiba/IBM Design Center, Austin.

Roy Kim received the M.S.C.E. degree from Syracuse University, Syracuse, NY, in 1985 and the B.S.E.E. degree from the State University of New York at Stony Brook in 1979.

He joined IBM in 1979 at the IBM Development Laboratory in Owego, NY. Over the years, he has worked on hardware development and project management in chip design, card design and system design of PowerPC and S390 architecture. Most recently, he worked on Cell processor design at the STI Design Center, Austin, TX.

Tien Le received the B.S. degrees in electrical and electronics engineering, mathematics, and computer science, and the M.S. degree in mathematics from California State Polytechnic University, Pomona.

He served as the verification lead for the SMF unit. In IMB, he worked in printer development, robotic automation of assembly line, RS System 6000 from first generation until 2002, and artificial intelligence for microprocessor verification. He joined the STI team in 2002 and currently is working on the follow-on project.

Peichun Liu received the B.S.E.E. degree from National Taiwan Ocean University in 1979 and the M.S.E.E. degree from University of Texas at Arlington in 1985.

He joined IBM in 1991 and has worked in the Power3, Power4, and Cell microprocessor design projects.

Jens Leenstra received the M.S. degree from the University of Twente, The Netherlands, in 1986 and the Ph.D. degree from the University of Eindhoven, The Netherlands, in 1993.

He joined IBM in 1994 at the IBM Development Laboratory in Boeblingen, Germany. He has worked in several areas of the Server Group development organization, including logic design and verification of I/O chips, multiprocessor system verification, and several PowerPC processor designs, including the Cell Processor. He is currently working on the next generation IBM microprocessors. His current interests focus on computer architecture, high frequency design, low power, and design for testability.

John Liberty received the B.S. degree in electrical engineering from North Carolina State University, Raleigh, in 1987, and the M.S. degree in electrical engineering in 1989.

In 1989, he joined IBM in the Engineering Graphics Organization. In that organization he worked in such areas of the graphic subsystems as board design, power supplies, EMC standards compliance and ASIC logic design. In 2000, he joined the STI Design Center, Austin, TX, to work on the Cell Broadband Engine processor Synergistic Processor Element as a Custom Logic Designer.

Brad Michael received the B.S degree in computer engineering from Iowa State University, Ames, in 1989.

He joined IBM in that year, and is now involved in microprocessor logic design in the STI design center in Austin, TX.

Hwa-Joon Oh received the B.S and M.S. degrees from Yonsei University, Seoul, Korea, in 1987 and 1989, and received the Ph. D. degree from Michigan State University, East Lansing, in 1996.

From 1996 to 1998, he was with Hyundai Electronics America, San Jose, CA, where he was involved in research and development of DRAM designs. In 1998, he joined IBM Austin Research Laboratory (ARL), Austin, TX, where he worked on POWER4 microprocessor design. From 2000 to 2001, he briefly worked on short random cycle embedded memory design for IBM microelectronics. Currently, he is working at IBM System and Technology Group with Sony-Toshiba-IBM Design Center, Austin, TX, where he is involved in architecture, logic, circuits, and physical implementations of Cell microprocessor. He has authored or co-authored several journal papers and patents. His main research areas are artificial neural network, SRAM and DRAM design, and broadband microprocessor design.

Silvia Melitta Mueller received the M.S. degree in mathematics and the Ph.D. degree in computer science from the University of Saarbruecken, Germany, in 1989 and 1991, respectively, and became a Privatdozent for CS.

After eight years of research and teaching, she joined IBM in 1999 at the IBM Development Laboratory in Boeblingen, Germany. She is a IBM Senior Technical Staff Member; she leads two logic teams developing high-performance power-effienct FPUs for the next generation IBM processors. In this role, her major focus is on power/performance tradeoffs, area reduction, and a more efficient way to handle the design complexity.

Osamu Takahashi (M'97) received the B.S. degree in engineering physics and the M.S. degree in electrical engineering from the University of California, Berkeley, in 1993 and 1995, respectively, and the Ph.D. degree in computer and mathematical sciences from Tohoku University, Sendai, Japan, in 2001.

He is an IBM Senior Technical Staff Member and the Manager of the circuit design team for the synergistic processor element developed at the STI Design Center, Austin, TX.

A. Hatakeyama, photograph and biography not available at the time of publication.

Yukio Watanabe, photograph and biography not available at the time of publication.

Naoka Yano received the Bachelor of Arts and Science degree in pure and applied sciences from the University of Tokyo, Tokyo, Japan, in 1991.

She joined the Semiconductor Device Engineering Laboratory, Toshiba Corporation, Kawasaki, Japan, in 1991. She was engaged in the research and development of high performance microprocessors, including the Emotion Engine. In 2001, she moved to Toshiba America Electronic Components, Inc., Austin, TX, where she was involved in the development of the Cell processor. Now she is with the Broadband System LSI Development Center, Semiconductor Company, Toshiba Corporation, Kawasaki, Japan.

Daniel A. Brokenshire received the B.S. degree in computer science and the B.S. and M.S. degrees in electrical engineering, all from Oregon State University, Corvallis.

He is a Senior Technical Staff Member with the STI Design Center, Austin, TX. His responsibilities include the development of programming standards, language extensions, and reusable software libraries for the Cell Broadband Engine.

Mohammad Peyravian received the Ph.D. degree in electrical engineering from the Georgia Institute of Technology, Atlanta, in 1992.

He is a Network Processor Architect at IBM Microelectronics, Research Triangle Park, North Carolina. His interests include networking, network processors, cryptography, and security. He has published over 40 journal and conference papers, and has over 30 patents in networking and cryptography/security.

Vandung To received the B.A degree in computer science from Rice University, Houston, TX, in 2001. She is currently pursuing the M.B.A. degree at the University of Texas in Austin.

She joined IBM in 2001 and is now working on CELL software development.

Eiji Iwata received the M.E. degree in information systems engineering from Kyushu University, Fukuoka, Japan.

He joined Sony Corporation, Tokyo, Japan, in 1991. In 1996, he was a Visiting Researcher at the Hydra Project in CSL, Stanford University, Stanford, CA. Since 1997, he developed a CMP for media applications in Sony and joined the STI Design Center, Austin, TX, in 2001. Currently, he works on a Cell-related project with Sony Computer Entertainment, Austin.