Cell Broadband Engine Microprocessor Architecture

Jaseem V V (06CS12)

ABSTRACT

Cell Broadband Engine is a microprocessor architecture jointly developed by Sony Computer Entertainment, Toshiba, and IBM. While the Cell chip can have a number of different configurations, the basic configuration is a multi-core chip composed of one "Power Processor Element" ("PPE") (sometimes called "Processing Element", or "PE"), and multiple "Synergistic Processing Elements" ("SPE"). The PPE and SPEs are linked together by an internal high speed bus dubbed "Element Interconnect Bus" ("EIB"). It also has a memory flow controller (MFC), the internal interrupt controller (IIC), and the Bus Interface Controller (BIC).

This architecture has powerful short vector SIMD capabilities and a massive register file. Cache hierarchies would be replaced by small and fast local memories and powerful DMA engines. This design approach resulted in a 241 million transistors chip, which today delivers performance barely approachable by its billion transistor counterparts and is available to the broad computing community in a truly off-the-shelf manner via a $500 PS 3 gaming console.

INDEX

1. **PREAMBLE** 4

2. **ACRONYM AND DEFINITIONS** 5

3. **ARCHITECTURE** 6

3.1.The PowerPC Processor Element 7

3.1.1 PowerPC Registers 7

3.1.2 PPE instruction sets 9

3.1.3 Addressing modes 9

3.1.4 Instruction types 10

3.1.5 Vector/SIMD Multimedia Extension instructions 11

3.1.6 Addressing modes 11

3.1.7 Instruction types 11

3.1.8 Vector data types 12

3.2 Synergistic Processor Elements (SPEs) 12

3.2.1 SPE configuration 13

3.2.2 Synergistic Processor Unit 13

3.2.3 SPE registers 14

3.2.4 The local store (LS) 14

3.2.5 SPU instruction set 14

3.3 Memory flow controller 15

3.3.1 Channels 15

3.3.2 Mailboxes 16

3.3.3 Signal notification 16

3.4 Data layout in registers **16**

3.5 Element Interconnect Bus (EIB) **17**

4. SCOPE OF THE SEMINAR **18**

5. CONCLUSION & FUTURE WORK **19**

6. REFERENCE **20**

1. PREAMBLE

Cell BE is a microprocessor architecture jointly developed by Sony Computer Entertainment, Toshiba, and IBM, an alliance known as "STI". Cell combines a general-purpose Power Architecture core of modest performance with streamlined co-processing elements which greatly accelerate multimedia and vector processing applications, as well as many other forms of dedicated computation.
The first major commercial application of Cell was in Sony's PlayStation 3 game console.

The architecture emphasizes efficiency/watt, prioritizes bandwidth over latency, and favors peak computational throughput over simplicity of program code. Software adoption remains a key issue in whether Cell ultimately delivers on its performance potential.
With the help of the computing power of over half a million PlayStation 3 consoles, the distributed computing project Folding@Home has been recognized by Guinness World Records as the most powerful distributed network in the world. The first record was achieved on September 16, 2007, as the project surpassed one petaFLOPS, which had never been reached before by a distributed computing network. Additionally, the collective efforts enabled PS3 alone to reach the petaFLOPS mark on September 23, 2007.

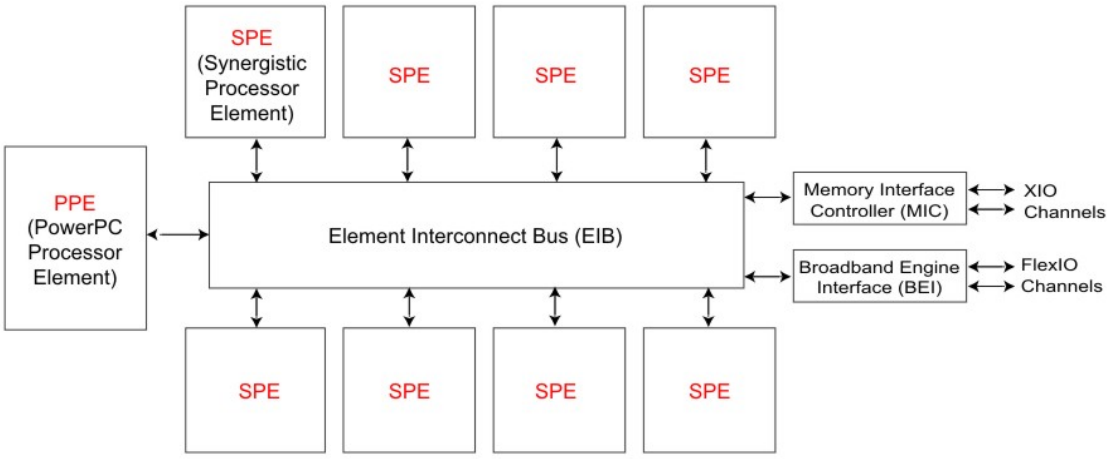
IBM's latest supercomputer, IBM Roadrunner, is a hybrid of General Purpose CISC Opteron as well as Cell processors. This system assumed the #1 spot on the June 2008 Top 500 list as the first supercomputer to run at petaFLOPS speeds, having gained a sustained 1.026 petaFLOPS speed.
In late 2008, a cluster of 200 PlayStation 3 consoles was used to generate a rogue SSL certificate, effectively cracking its encryption.

This design approach resulted in a 241 million transistors chip, which today delivers performance barely approachable by its billion transistor counterparts and is available to the broad computing community in a truly off-the-shelf manner via a $500 PS 3 gaming console.

2. ACRONYMS AND DEFINITIONS

BEI	Broadband Engine Interface
CBEA	Cell Broadband Engine Architecture
CR	Condition Register
CTR	Count Register
DMA	Direct Memory Access
EIB	Element Interconnect Bus
FPRs	Floating Point Registers
FXU	Fixed-point Unit
GPRs	General Purpose Registers
LR	Link Register
LS	Local Store
LSB	Least-Significant bit
MFC	Memory Flow Controller
MIC	Memory Interface Controller
MMIO	Memory-Mapped I/O
MSB	Most-Significant bit
PPE	Power Processor Element
PPSS	Power Processor Storage Subsystem
PPU	Power Processor Unit
RISC	Reduced Instruction Set Computer
SIMD	Single Instruction Multiple Data
SPE	Synergistic Processor Element
VRs	Vector Registers
VSCR	Vector Status and Control Register
VRSAVE	Vector Save Register
VXU	Vector/SIMD Multimedia Extension unit
XER	Fixed-Point Exception Register

3. ARCHITECTURE



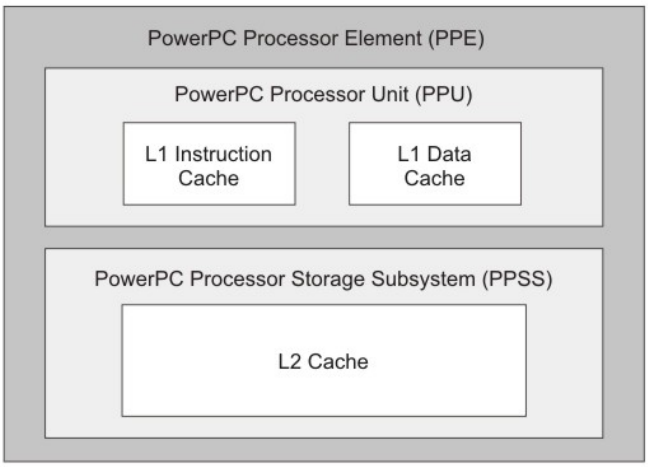
The Cell BE chip is first released with 95nm technology and at present its 65nm and 45nm chip are scheduled for the next release. This design approach resulted in a 241 million transistors chip, which today delivers performance barely approachable by its billion transistor counterparts and is available to the broad computing community in a truly off-the-shelf manner via a $500 PS 3 gaming console. Whereas the Intel Quad-core have more than 2 billion transistors and offers less performance and the consumes more power and dissipates more heat and costs more compared to the Cell BE.

The Cell Broadband Engine Architecture (CBEA) defines a processor structure directed toward distributed processing. The Cell Broadband Engine is a single-chip multiprocessor with nine processors operating on a shared, coherent memory. In this respect, it extends current trends in PC and server processors. Their function is specialized into two types:

- Power Processor Element (PPE)
- Synergistic Processor Element (SPE).

The CBE processor has one PPE and eight SPEs linked together by an internal high speed bus, the "Element Interconnect Bus" ("EIB").

3.1 The PowerPC Processor Element

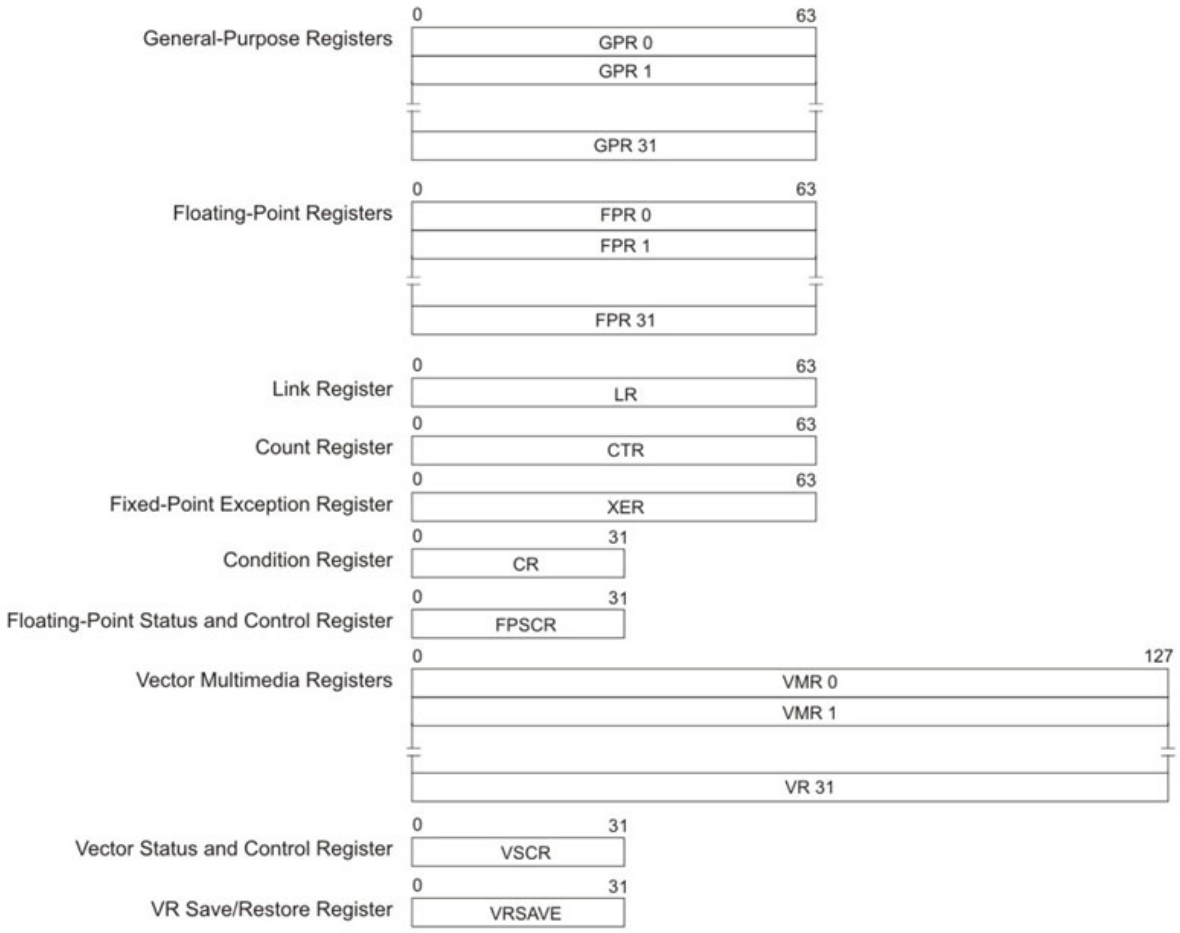


The PowerPC Processor Element (PPE) is a general-purpose, dual-threaded, 64-bit RISC processor that conforms to the PowerPC Architecture, with the Vector/SIMD Multimedia Extension.
The PPE consists of *two* main units: The Power Processor Unit (PPU) and the Power Processor Storage Subsystem (PPSS).The PPE is responsible for overall control of the system. It runs the operating systems for all applications running on the Cell Broadband Engine. The PPU supports two simultaneous threads of execution and can be viewed as a 2-way multiprocessor with shared dataflow. This appears to software as two independent processing units. The PPSS handles memory requests from the PPE and external requests to the PPE from other processors or I/O devices.

3.1.1 PowerPC Registers

All computational instructions operate only on registers – there are *no* computational instructions that modify storage. To use a storage operand in a computation and then modify the same or another storage location, the contents of the storage operand must be:
1. loaded into a register,
2. modified,
3. stored back to the target location.

The PPE registers include:
General-Purpose Registers (GPRs) – Fixed-point instructions operate on the full 64-bit width of the GPRs, of which there are 32. The instructions are mode-independent, except that in 32-bit mode, the processor uses only the low-order 32 bits for determination of a memory address and the carry, overflow, and record status bits.



Floating-Point Registers (FPRs) – The 32 FPRs are 64 bits wide. The internal format of floating-point data is the IEEE 754 double-precision format. Single-precision results are maintained internally in the double-precision format.

Link Register (LR) – The 64-bit LR can be used to hold the effective address of a branch target. Branch instructions with the link bit (LK) set to 1 (that is, subroutine-call instructions) copy the next instruction address into the LR. A Move To Special-Purpose Register instruction can copy the contents of a GPR into the LR.

Count Register (CTR) – The 64-bit CTR can be used to hold either a loop counter or the effective address of a branch target. Some conditional-branch instruction forms decrement the CTR and test it for a zero value. A Move To Special-Purpose Register instruction can copy the contents of a GPR into the CTR.

Fixed-Point Exception Register (XER) – The 64-bit XER contains the carry and overflow bits and the byte count for the move-assist instructions. Most arithmetic operations have instruction forms for setting the carry and overflow bit.

Condition Register (CR) – Conditional comparisons are performed by first setting a condition code in the 32-bit CR with a compare instruction or with a recording instruction.

The condition code is then available as a value or can be tested by a branch instruction to control program flow. The CR consists of eight independent 4-bit fields grouped together for convenient save or restore during a context switch. Each field can hold status information from a comparison, arithmetic, or logical operation. The compiler can schedule CR fields to avoid data hazards in the same way that it schedules the use of GPRs. Writes to the CR occur only for instructions that explicitly request them; most operations have recording and non-recording instruction forms.
Floating-Point Status and Control Register (FPSCR*)* – The processor updates the 32-bit FPSCR after every floating-point operation to record information about the result and any associated exceptions. The status information required by IEEE 754 is included, plus some additional information for exception handling.

Vector Registers (VRs) – There are 32 128-bit-wide VRs. They serve as source and destination registers for all vector instructions.

Vector Status and Control Register (VSCR) – The 32-bit VSCR is read and written in a manner similar to the FPSCR. It has 2 defined bits, a non-Java™ mode bit and a saturation bit; the remaining bits are reserved. Special instructions are provided to move the VSCR to a VR register.

Vector Save Register (VRSAVE) – The 32-bit VRSAVE register assists user and privileged software in saving and restoring the architectural state across context switches.

3.1.2 PPE instruction sets

The PowerPC Processor Element (PPE) supports *two* instruction sets: the PowerPC instruction set and the Vector/SIMD Multimedia Extension instruction set.
The PowerPC instruction set uses instructions that are 4 bytes long. It supports byte, halfword, word, and doubleword operand accesses between storage and its 32 general-purpose registers (GPRs). The instruction set also supports word and doubleword operand accesses between storage and a set of 32 floating-point registers (FPRs). Signed integers are represented in twos-complement form. The Vector/SIMD Multimedia Extension instruction set uses instructions that, like PowerPC instructions, are 4 bytes long. Most of the Vector/ SIMD Multimedia Extension operands are vectors, including single-precision floating-point, integer, scalar, and fixed-point of vector-element sizes of 8,16, and 32 bits, PowerPC instructions. It follows Big-endian data ordering.

3.1.3 Addressing modes

All instructions, except branches, generate addresses by incrementing a program counter. All load and store instructions specify a base register. The effective address in memory for a data value is calculated relative to the base register in one of three ways:

Register + Displacement – The displacement forms of the load and store instructions calculate an address that is the sum of a displacement specified by the sign-extended 16-bit immediate field of the instruction plus the contents of the base register.

Register + Register – The indexed forms of the load and store instructions calculate an address that is the sum of the contents of the index register, which is a GPR, plus the contents of the base register.

Register – The Load String Immediate and Store String Immediate instructions use the unmodified contents of the base register to calculate an address. Loads and stores can specify an update form that reloads the base register with the computed address, unless the base register is the target register of the load. Branches are the only instructions that explicitly specify the address of the next instruction.

A branch instruction specifies the effective address of the branch target in one of the following ways:

Branch Not Taken – The byte address of the next instruction is the byte address of the current instruction, plus 4.

Absolute – The word address of the next instruction is given in an immediate field of the branch instruction.

Relative – The word address of the next instruction is given by the sum of the immediate field of the branch instruction and the word address of the branch instruction itself.

Link Register or Count Register – The byte address of the next instruction is the effective byte address of the branch target specified in the Link Register or Count Register, respectively.

3.1.4 Instruction types

The PowerPC Processor Element (PPE)'s PowerPC instructions can have up to three operands. Most computational instructions specify two source operands and one destination operand. The PPE's PowerPC instructions include the following types:

Integer Instructions – These include arithmetic, compare, logical, and rotate/shift instructions. They operate on byte, halfword, word, and doubleword operands.

Floating-Point Instructions – These include floating-point arithmetic, multiply-add, compare, and move instructions, as well as instructions that affect the Floating-Point Status and Control Register (FPSCR). Floating-point instructions operate on single-precision and double-precision floating-point operands.

Load and Store Instructions – These include integer and floating-point load and store instructions, with byte-reverse, multiple, and string options for the integer loads and stores.

Memory Synchronization Instructions – These instructions control the order in which memory operations are completed with respect to asynchronous events, and the order in which memory operations are seen by other processors or memory-access mechanisms. The

instruction types include load and store with reservation, synchronization, and enforce in-order execution of I/O. They are especially useful for multiprocessing.

Flow Control Instructions – These include branch, Condition-Register logical, trap, and other instructions that affect the instruction flow.

Processor Control Instructions – These instructions are used for synchronizing memory accesses and managing caches, Translation Lookaside Buffers (TLBs), segment registers, and other privileged processor states. They include move-to/from special-purpose register instructions.

Memory and Cache Control Instructions – These instructions control caches, TLBs, and segment registers.

External Control Instructions – These instructions allow a user-level program to communicate with a special-purpose device.

3.1.5 Vector/SIMD Multimedia Extension instructions

The 128-bit Vector/SIMD Multimedia Extension unit (VXU) operates concurrently with the PPU's fixed-point integer unit (FXU) and floating-point execution unit (FPU). Like PowerPC instructions, the Vector/SIMD Multimedia Extension instructions are 4 bytes long. The Vector/SIMD Multimedia Extension instructions support simultaneous execution on multiple elements that make up the 128-bit vector operands. These vector elements may be byte, halfword, or word.
All Vector/SIMD Multimedia Extension instructions are designed to be easily "pipelined". Parallel execution with the PPE's integer and floating-point instructions is simplified by the fact that Vector/SIMD Multimedia Extension instructions:
do *not* generate exceptions (other than data-storage interrupt exceptions on loads and stores),
do *not* support unaligned memory accesses or complex functions, and
share few resources or communication paths with the other PPE execution units.

3.1.6 Addressing modes

All Vector/SIMD Multimedia Extension load and store operations use the register + register indexed addressing mode, which forms the sum of the contents of an index GPR plus the contents of a base-address GPR. This addressing mode is very useful for accessing arrays. In addition to the load and store operations, the Vector/SIMD Multimedia Extension instruction set provides a powerful set of element-manipulation instructions – for example, shuffle, permute (similar to the SPEs' shuffle), rotate, and shift – to manipulate vector elements into the desired alignment and arrangement after the vectors have been loaded into vector registers.

3.1.7 Instruction types

Most Vector/SIMD Multimedia Extension instructions have three or four 128-bit vector operands – two or three source operands and one result. Also, most instructions are SIMD in

nature. The instructions have been chosen for their utility in digital signal processing (DSP) algorithms, including 3D graphics. The Vector/SIMD Multimedia Extension instructions include the following types:

Vector Integer Instructions – These include vector arithmetic, compare, logical, rotate, and shift instructions. They operate on byte, halfword, and word vector elements. The instructions use saturation-clamping.

Vector Floating-Point Instructions – These include floating-point arithmetic, multiply/add, rounding and conversion, compare, and estimate instructions. They operate on single-precision floating-point vector elements.

Vector Load and Store Instructions – These include only basic integer and floating-point load and store instructions. No update forms of the load and store instruction are provided. They operate on 128-bit vectors.

Vector Permutation and Formatting Instructions – These include vector pack, unpack, merge, splat, permute, select, and shift instructions.

Processor Control Instructions – These include instructions that read and write the vector status and control register (VSCR).

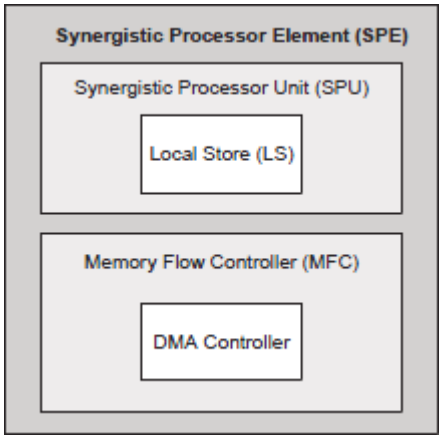
Memory Control Instructions – These include instructions for managing caches (user-level and supervisor-level). These instructions are "no-ops".

3.1.8 Vector data types

The Vector/SIMD Multimedia Extension model adds a set of fundamental data types, called vector types. The represented values are in decimal (base-10) notation. The vector registers are 128 bits and can contain:

Sixteen 8-bit values, signed or unsigned
Eight 16-bit values, signed or unsigned
Four 32-bit values, signed or unsigned
Four single-precision IEEE-754 floating-point values

3.2 Synergistic Processor Elements (SPEs)



The eight identical Synergistic Processor Elements (SPEs) are optimized for compute-intensive applications in which a program's data and instruction needs can be anticipated and transferred into the local store (LS) by DMA while the SPE computes using previously

transferred data and instructions. The streaming data sets in 3D graphics, media, and broadband communications are examples of applications that run well on SPEs. However, the SPEs are not optimized for running programs that have significant branching, such as an operating system. Each SPE supports only a single program context at any one time. Typically, the operating system runs on the PPE, and user-mode threads are execute on the SPEs. The SPEs achieve high performance, in part, by eliminating the overhead of load and store address translation, hardware-managed caches, out-of-order instruction issue, and branch prediction. Instead, the SPEs capitalize on the high computational efficiencies that can be obtained for streaming-data applications by providing a large (128-entry by 128-bit) unified register file, dual-instruction issue, and high DMA bandwidth between the LS and main storage. Each SPE supports the single-instruction, multiple-data (SIMD) instruction architecture.

3.2.1 SPE configuration

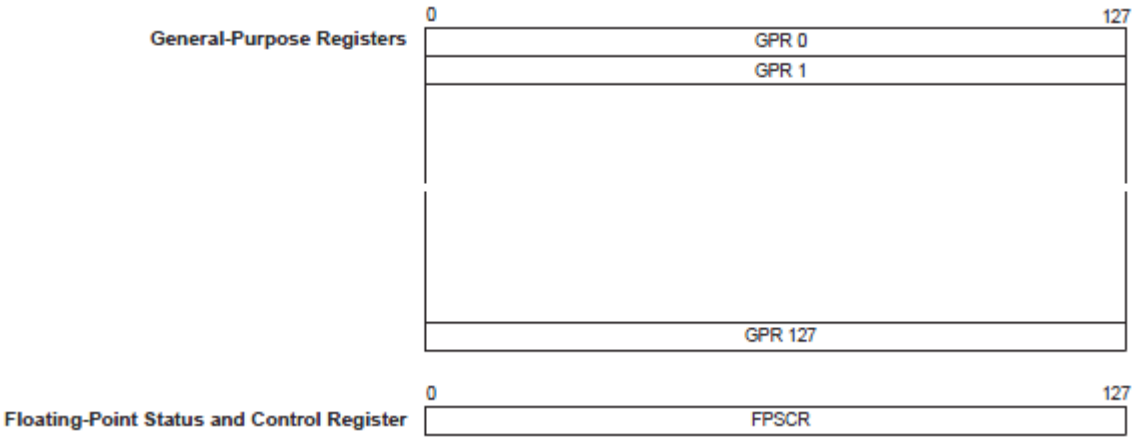
This section describes the main components of a Synergistic Processor Element (SPE). Their functions include: Synergistic Processor Unit (SPU) — The SPU executes SPU instructions fetched from its 256-KB LS. The SPU fills its LS with instructions and data using DMA transfers initiated from SPU or PPE software. Memory Flow Controller (MFC) — The MFC provides the interface, by means of the Element Interconnect bus (EIB), between the SPU and main storage. The MFC performs DMA transfers between the SPU's LS and main storage, and it supports mailbox and signal-notification messaging between the SPE and the PPE and other devices. The SPU communicates with its MFC through SPU channels. The PPE and other devices (including other SPEs) communicate with an MFC through memory-mapped I/O (MMIO) registers associated with the SPU's channels.

3.2.2 Synergistic Processor Unit

Each of the eight SPEs is an independent processor with its own program counter, register file, and 256-KB LS. An SPE operates directly on instructions and data in its LS. It fills its LS by requesting DMA transfers from its MFC, which manages the DMA transfers. The SPU has specialized units for executing load and store, fixed-point, floating-point unit (single-precision and double-precision), and channel-interface instructions. The large 128-entry, 128-bit wide register file, and its flat architecture (all operand types stored in a single register file), allows for instruction-latency hiding without speculation. The register file is unified—meaning that all data types (integer, single-precision and double-precision floating-point, scalars, vectors, logicals, bytes, and others) use the same register file. The register file also stores return addresses, results of comparisons, and so forth. As a consequence of the large, unified register file, expensive hardware techniques such as out-of-order processing or deep speculation are not needed to achieve high performance. LS addresses can be aliased by PPE privileged software onto the main-storage (effective-address) space. DMA transfers between the LS and main storage are coherent in the system. A pointer to a data structure created on the PPE can be passed to an SPU, and the SPU can use this pointer to issue a DMA command to bring the data structure into its LS. PPE software can use locking instructions and mailboxes for synchronization and mutual exclusion. The SPU architecture has the following restrictions: v No direct (SPU-program addressable) access to main storage. The SPU accesses main storage only by using the

MFC's DMA transfers. v No direct access to system control, such as page-table entries. PPE privileged software provides the SPU with the address-translation information that its MFC needs. v With respect to accesses by its SPU, the LS is unprotected and un-translated storage.

3.2.3 SPE registers



This section describes the Synergistic Processor Element (SPE) user registers. The complete set of SPE user registers is shown in Figure. All computational instructions operate only on registers—there are no computational instructions that modify storage.
The SPE registers include:
General-Purpose Registers (GPRs) — All data types can be stored in the 128-bit GPRs, of which there are 128.
Floating-Point Status and Control Register (FPSCR) — The processor updates the 128-bit FPSCR after every floating-point operation to record information about the result and any associated exceptions.

3.2.4 The local store (LS)

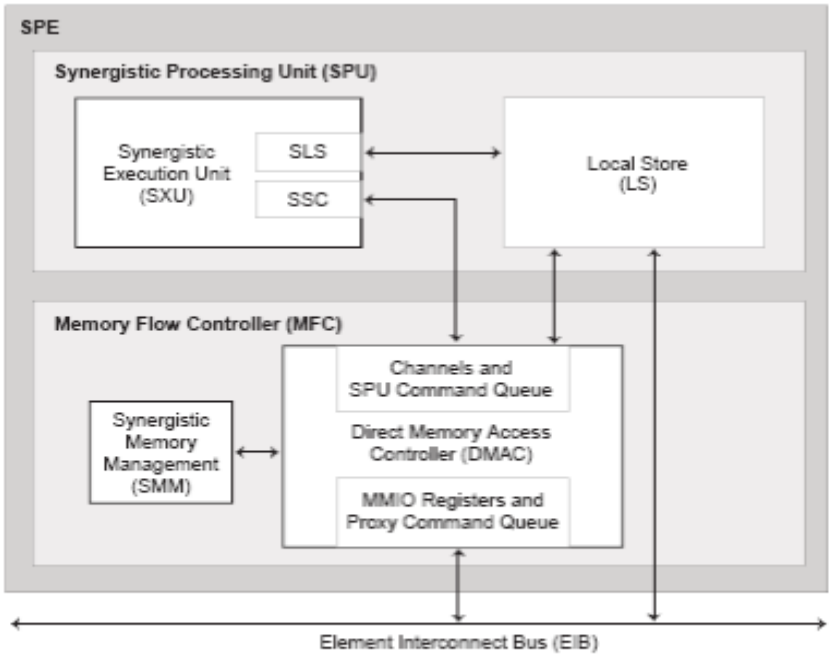
The local store (LS) can be regarded as a software-controlled cache that is filled and emptied by DMA transfers.
Key features of the LS include:
Holds instructions and data
16-bytes-per-cycle load and store bandwidth,
128-bytes-per-cycle DMA-transfer bandwidth
128-byte instruction prefetch per cycle
The SPU arbitrates access to the LS according the following priorities (with the highest priority first):
1. DMA reads and writes by the PPE or an I/O device.
2. SPU loads and stores.
3. Instruction prefetch.

3.2.5 SPU instruction set

The SPU ISA operates primarily on SIMD vector operands, both fixed-point and floating-point, with support for some scalar operands. The PPE and the SPE both execute SIMD

instructions, but the two processors execute different instruction sets, and programs for the PPE and SPEs must be compiled by different compilers.

3.3 Memory flow controller



MMIO Memory-Mapped I/O Registers
SLS SPU Load and Store Unit
SSC SPU Channel and DMA Unit

The primary functions of the Memory Flow Controller (MFC) are to connect the SPU to the EIB and support DMA transfers between main storage and the LS. The MFC maintains and processes queues of DMA commands from its SPU or from the PPE or other devices. The MFC's DMA controller (DMAC) executes the DMA commands. This allows the SPU to continue execution in parallel with the MFC's DMA transfers. The MFC supports channels and associated MMIO registers for the purposes of enqueueing and monitoring DMA commands, monitoring SPU events, performing interprocessor-communication via mailboxes and signal-notification, accessing auxiliary resources such as the decrementer (timer), and other functions.

3.3.1 Channels

Channels are unidirectional message-passing interfaces that support 32-bit messages and commands. Many of the channels provide communications between the SPE and its MFC, which in turn, mediates communication with the PPE and other devices. Each channel has a corresponding count that indicates the remaining capacity (the maximum number of outstanding transfers) in that channel.

Key features of the SPE channel operations include:

All transactions on the channel interface are unidirectional.

Each channel transaction is independent of any other transaction.

Sequential read and write transactions are supported.

External access to control MMIO registers has higher priority than channel operations.

Channel operations are done in program order.
Channel read operations to reserved channels return zeros.
Channel write operations to reserved channels have no effect.
Reading of channel counts on reserved channels returns zero.

3.3.2 Mailboxes

Mailboxes are queues that support exchanges of 32-bit messages between an SPE and other devices. Each mailbox queue has an SPE channel assignment as well as a corresponding MMIO register assignment. Two 1-entry mailbox queues are provided for sending messages from the SPE:

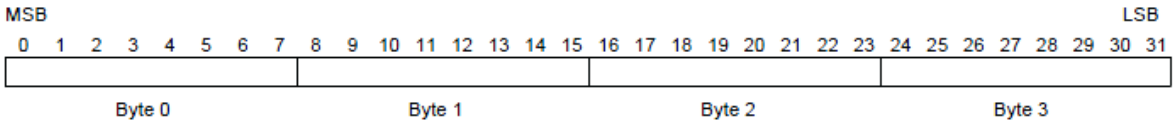
- SPU Write Outbound Mailbox
- SPU Write Outbound Interrupt Mailbox

Although the mailboxes are primarily intended for communication between the PPE and the SPEs, they can also be used for communication between an SPE and other SPEs, processors, or devices. For this to happen, however, privileged software needs to allow one SPE to access the mailbox register in another SPE. If software does not allow this, then only system memory communications are available for SPE-to-SPE communications.

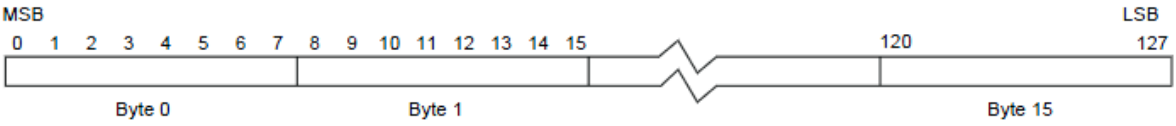
3.3.3 Signal notification

Signal-notification channels, or signals, are inbound (to an SPE) registers. They can be used by other SPEs, the PPE, or other devices to send information, such as a buffer-completion synchronization flag, to an SPE. Each SPE has two 32-bit signal-notification registers, each of which has a corresponding memory-mapped I/O (MMIO) register into which the signal-notification data is written by the sending processor.

3.4 Data layout in registers



Bit and Byte Order for a 32-bit Word



Bit and Byte Order for a 128-bit Register

The SPE supports big-endian data ordering, an ordering in which the lowest-address byte and lowest-numbered bit are the most-significant (high) byte and bit, respectively. Bits in registers are numbered in ascending order from left to right, with bit 0 representing the most-significant bit (MSb) and bit 127 the least-significant bit (LSb). The SPU hardware defines the following data types:

Byte, halfword, word, doubleword, quadword

3.5 Element Interconnect Bus (EIB)

The PPE and SPEs communicate coherently with each other and with main storage and I/O through the EIB. The EIB is a 4-ring structure (two clockwise and two counterclockwise) for data, and a tree structure for commands. The EIB's internal bandwidth is 96 bytes per cycle, and it can support more than 100 outstanding DMA memory requests between main storage and the SPEs.

The memory-coherent EIB has *two* external interfaces:

The **Memory Interface Controller (MIC)** provides the interface between the EIB and main storage. It supports two Rambus Extreme Data Rate (XDR) I/O (XIO) memory channels and memory accesses on each channel of 1-8, 16, 32, 64, or 128 bytes.

The **Cell Broadband Engine Interface (BEI)** manages data transfers between the EIB and I/O devices. It provides address translation, command processing, an internal interrupt controller, and bus interfacing. It supports two Rambus FlexIO external I/O channels. One channel supports only non-coherent I/O devices. The other channel can be configured to support either non-coherent transfers or coherent transfers that extend the logical EIB to another compatible external device, such as another Cell Broadband Engine.

4. SCOPE OF THE SEMINAR

In the world of computers, the most powerful of them are the ones termed as supercomputers.
And of the supercomputers, the most powerful ones are made with the combination of the Cell Broadband Engine and the AMD Opteron at present.
More than that the same microprocessor is used to power the best and the most powerful gaming console in the world, the PlayStation 3.
So this is look into how all these work with a tiny chip, the Cell BE microprocessor, and its architecture for the curious.

5. CONCLUSION & FUTURE WORK

This microprocessor provides robust power and efficiency with its 9 cores. It's used for complex graphics operations, simulations, research, gaming and much more.

There are versions of GNU/Linux of which – Yellow Dog Linux that runs on this architecture and brings the power of to the desktop users.

However the programmers must use the programming intrinsic and constructs provided by the Cell Broadband Engine SDK to target the PPE and SPEs and to make the program utilize the power of the microprocessor the best.

In future a 45 nm Cell BE microprocessor will be rolling out that will power the world's best super computers as well as the Play Station gaming consoles equally likely.

6. REFERENCE

[1] Cell Broadband Engine Programming Handbook *IBM*
[2] Software Development Kit for Multicore Acceleration Version 3.0 Programming Tutorial
IBM
[3] Cell Broadband Engine Programming Tutorial v 2.0 *IBM*
[4] Programming the Cell Broadband Engine™ Architecture Examples and Best Practices
IBM
[5] C/C++ Language Extensions for Cell Broadband Engine Architecture Version 2.5 *IBM*
[6] PS3 Programming Lectures *MIT*
[7] SPU C/C++ Language Extensions Version 2.1 *IBM*