

---

# Cell Broadband Engine Architecture from 20,000 feet

A brief view of CBEA's tripartite organization of storage and its programming implications

Skill Level: Introductory

[Dr. H. Peter Hofstee \(hofstee@us.ibm.com\)](mailto:hofstee@us.ibm.com)

Architect  
IBM

24 Aug 2005

The Cell Broadband Engine Architecture (CBEA, or, informally, "Cell") defines a new processor structure based upon the 64-bit Power Architecture™ technology, but with unique features directed toward distributed processing and media-rich applications. The Cell architecture defines a single-chip multiprocessor consisting of one or more Power Processor Elements (PPEs) and multiple high-performance SIMD Synergistic Processor Elements (SPEs). While each SPE is an independent processor running its own application programs, a shared, coherent memory and a rich set of DMA commands provide for seamless and efficient communications between all Cell processing elements. This article provides a concise view inside the Cell's architecture.

The first generation Cell Broadband Engine is the first incarnation of the new family of microprocessors conforming to the CBEA. The CBEA is a new architecture that extends the 64-bit Power Architecture technology. The CBEA and the Cell Broadband Engine are the result of a collaboration between Sony, Toshiba, and IBM known as STI, formally started in early 2001.

Although the Cell Broadband Engine is initially intended for application in game consoles and media-rich consumer-electronics devices, the architecture and the Cell Broadband Engine implementation have been designed to overcome some of the fundamental limitations to processor performance. A much broader use of the architecture is envisioned.

## The Cell structure

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.

The most distinguishing feature of the Cell Broadband Engine is that although all processors share main storage (the effective-address space that includes main memory), their function is specialized into two types -- the Power Processor Element (PPE) and the Synergistic Processor Element (SPE). The Cell Broadband Engine has one PPE and eight SPEs.

- The first type of processor, the PPE, is a 64-bit Power Architecture core. It is fully compliant with the 64-bit Power Architecture specification and can run 32-bit and 64-bit operating systems and applications.
- The second type of processor, the SPE, is optimized for running compute-intensive applications, and it is not optimized for running an operating system.

The SPEs are independent processors, each running its own individual application programs. Each SPE has full access to coherent shared memory, including the memory-mapped I/O space. The designation *synergistic* for this processor was chosen carefully -- there is a mutual dependence between the PPE and the SPEs. The combination of the two working in harmony produces a greater effect than each working alone. The SPEs depend on the PPE to run the operating system and, in many cases, the top-level control thread of an application. The PPE depends on the SPEs to provide the bulk of the application performance.

## To the programmer

The SPEs are designed to be programmed in high-level languages and support a rich instruction set that includes extensive single-instruction, multiple-data functionality (SIMD). However, just like conventional processors with SIMD extensions, use of SIMD data types is preferred but is not mandatory. For programming convenience, the PPE also supports the Power Architecture Vector/SIMD Multimedia Extension.

To an application programmer, the Cell Broadband Engine looks like a nine-way coherent multiprocessor. The PPE is more adept at control-intensive tasks and quicker at task switching. The SPEs are more adept at compute-intensive tasks and slower at task switching. However, either processor is capable of both types of

functions. This specialization has allowed increased efficiency in the implementation of both the PPE and the SPE (especially the SPE), and is a significant factor in the substantial performance improvement in applications which take advantage of the CBEA.

The more significant difference between the SPE and PPE is in how they access memory. The PPE accesses main storage (the effective-address space that includes main memory) with load and store instructions that go between a private register file and main storage (which may be cached), whereas the SPEs access main storage with DMA commands that go between main storage and a private local memory used to store both instructions and data. SPE instruction- fetches and load and store instructions access this private local store rather than shared main storage.

This three-level organization of storage (register file, local store, main storage) -- with asynchronous DMA transfers between local store and main storage -- is a radical break with conventional architecture and programming models because it explicitly parallelizes computation and the transfers of data and instructions.

The reason for this radical change is that memory latency, measured in processor cycles, has gone up several hundredfold in the last 20 years. The result is that application performance is often limited by memory latency rather than peak compute capability or peak bandwidth. When a sequential program on a conventional architecture performs a load instruction that misses in the caches, program execution now comes to a halt for several hundred cycles. Compared with this penalty, the few cycles it takes to set up a DMA transfer for an SPE is quite small. Even with deep and costly speculation, conventional processors manage to get at best a handful of independent memory accesses in flight. The result can be compared to a bucket brigade in which a hundred people are required to cover the distance to the water needed to put the fire out, but only a few buckets are available.

In contrast, the explicit DMA model allows each SPE to have many concurrent memory accesses in flight without the need for speculation.

The most productive SPE memory-access model appears to be the one in which a list (such as a scatter-gather list) of DMA transfers is constructed in an SPE's local store so that the SPE's DMA controller can process the list asynchronously while the SPE operates on previously transferred data. In several cases, this new approach to accessing memory has led to application performance exceeding that of conventional processors by almost two orders of magnitude, significantly more than anyone would expect from the peak performance ratio (about 10x) between the Cell Broadband Engine and conventional PC processors.

## In summary

This article provided a brief introduction to the Cell Broadband Processor Architecture and gave a rationale for the major design decisions in the architecture.

# Resources

## Learn

- developerWorks just posted a bunch of [Cell documentation for download](#), including:
  - Cell Broadband Engine Architecture V1.0
  - Synergistic Processor Unit (SPU): Instruction Set Architecture V1.0
  - SPU Application Binary Interface: Specification V1.3
  - SPU Assembly Language: Specification V1.2
  - SPU C/C++ Language: Extensions V2.0
- Direct questions regarding availability, licensing, sales, and similar Cell-related topics to [IBM Engineering & Technical Services \(E&TS\)](#).
- The new Cell specification documents will be most useful to readers who are already familiar with the Power Architecture specification, as defined in the following resources:
  - [PowerPC User Instruction Set Architecture, Book I](#)
  - [PowerPC Virtual Environment Architecture, Book II](#)
  - [PowerPC Operating Environment Architecture, Book III](#)
  - [PowerPC Microprocessor Family: The Programming Environments for 32-Bit Microprocessors \(G522-0290-01\)](#)
  - [The Programming Environments Manual for 64-bit Microprocessors](#)
- Find [a dedicated Cell section](#) in the IBM Semiconductor Solutions Technical library and [elsewhere on the Semiconductor Solutions site](#).
- Learn about [Cell microprocessor services offerings from IBM E&TS](#).
- [Spufs: The Cell Synergistic Processing Unit as a virtual file system](#) discusses the Linux port to Cell, as does [Meet the experts: Arnd Bergmann on Cell](#) (developerWorks, June 2005).
- Read about the GCC port to Cell in [Porting the GNU Tool Chain to the Cell Architecture](#) from the proceedings of the GCC Developers' Summit in Ottawa, Ontario, Canada, June 2005 (PDF; scroll down to page 185 for the Cell-specific information).

## Discuss

- Technical discussion of these documents is going on now at the [Cell Architecture forum](#); comments and errata are welcome there also, or in the [comments box](#) at the bottom of this page. Errata on the specifications are welcome via e-mail to [CBE\\_Documentation@us.ibm.com](mailto:CBE_Documentation@us.ibm.com).

## About the author

Dr. H. Peter Hofstee

Dr. H. Peter Hofstee is the chief architect of the Cell Synergistic Processor, and Cell chief scientist. He received his PhD in computer science from the California Institute of Technology (Caltech) in 1995, and joined the Caltech faculty in 1995 and 1996 to teach computer science and VLSI. In 1996 he joined the IBM Austin research laboratory where he helped create the first GHz CMOS processor. Between 1997 and 2000 he worked on a number of other high-frequency server processor designs. In 2000 he helped create the concept for Cell and became one of the founding members of the STI (Sony -Toshiba - IBM) design center in the spring of 2001. His current interest focuses on application of the Cell processor beyond the gaming space and on future Cell designs.