

# MultiCore Framework

## Harnessing the Performance of the Cell BE™ Processor

- Simplifies the development of Cell-based high-performance applications
- Runs tasks on the SPE without Linux® overhead
- Overlaps data movement and computation automatically
- Maximizes the resources and performance available to your application
- Leverages proven Mercury technologies

MultiCore Framework (MCF) software from Mercury Computer Systems is an application programming interface (API), which facilitates the development of applications that execute on a heterogeneous multicore processor such as the Cell Broadband Engine™ (BE) processor. MCF maximizes resources and application performance by taking full advantage of the multicore processor's computation model. The MCF library of functions manages concurrent processes running on the heterogeneous cores that make up the multicore processor and efficiently distributes data.

### Multicore Architecture

The Cell BE processor is a breakthrough technology capable of massive floating-point processing for compute-intensive applications. The chip's multicore architecture consists of a general-purpose processor called the Power™ processor element (PPE), and an array of eight math co-processors called synergistic processor elements (SPEs) (see Figure 1). The SPEs have their own individual local stores (LSs) with 256 KB of memory each. This architecture naturally lends itself to an environment in which the general-purpose processor manages the other processors, which efficiently perform computations and move data in parallel.

The unprecedented processing density of this multicore architecture requires a unique programming approach for developers to deploy

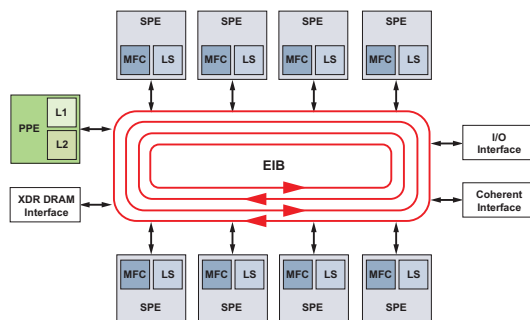


Figure 1. Cell BE processor functional block diagram

solutions quickly and effectively. To harness the full performance potential of the Cell BE processor, developers need the help of a software framework that supports its computation model and heterogeneous, distributed memory architecture.

### Function Offload Engine Model

MCF is an API for programming multicore processors using the function offload engine (FOE) model. In this model, the manager (PPE) is the coordinator responsible for running the control plane code, while the workers (SPEs) are the math/function offload engines responsible for running the processing plane code. The primary goal of the MCF library is to provide an efficient way to use the “strip-mining” data movement technique, which moves data in successive chunks from main store (XDR) into the LSs of the SPEs, where partial results are produced. Mercury's implementation of this technique allows for concurrent I/O and processing while minimizing the amount of system-level code that must reside on each SPE.

The FOE model gives the application developer the flexibility to use MCF to implement a wide variety of applications. The developer writes an application-specific program for the manager (PPE) and one or more complementary programs for the tasks that the SPE workers must perform.

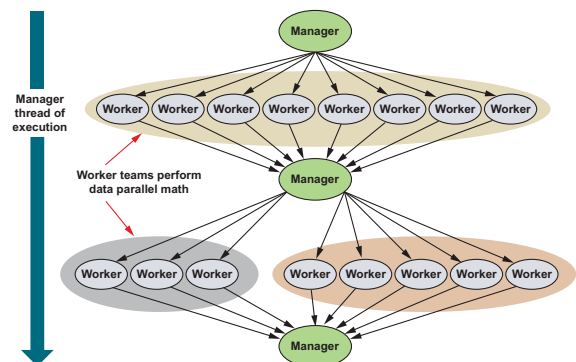


Figure 2. Function offload engine (FOE) example

## Optimizing Application Performance

MCF supports programmers who need precise control of data distribution and assignment of processing resources on a multicore processor, but relieves them of the hardware-specific details involved. The keys to achieving high multicore processor performance are:

- Minimizing latency
- Efficient resource utilization
- Overlapping computation and communication
- Appropriate computational granularity
- Limiting SPE code, ideally, to just DMAs and math

## Using MCF

The application developer uses MCF to establish a network consisting of a manager and a set of workers, to define teams of workers, and to assign and queue tasks to each team of workers. These queued tasks are run without Linux® overhead. Once the teams are established, MCF can be used to allocate DMA-friendly aligned memory regions, find named remote allocation locations via a name server, and perform multi-buffered strip-mining of n-dimensional data sets between a large memory (XDR memory in the Cell architecture) and the small worker memories (local stores).

Tile channels facilitate the communication and data movement between the manager and the workers for lower latency and efficient resource utilization. The manager calculates the DMA operations and

descriptor information that the workers must perform. After the data organization has been defined by the manager, the worker task connects to the tile channel and thereby obtains the prescription for DMA transfers that move data into and out of the local store (see Figure 3). When a worker gets a tile buffer from a tile channel, a tile descriptor is provided. This tile descriptor provides information that facilitates accessing the tile data in local store. This information includes the location of the tile data in local store (its virtual address), as well as the dimension, size, and other tile-specific parameters.

To support data reorganization (such as global matrix transpose), MCF contains reorg channels, which are similar to tile channels. The application can optimally perform multi-buffered, many-to-many, n-dimensional data reorganizations among sets of worker teams via the reorg channels. While the tile and reorg channels are useful for managing data movement, MCF message queues can be used to transmit control information among members of an MCF network

In addition to the synchronization and data movement provided by the channels to manage multi-buffered data movement, barriers, semaphores and low level DMA functions are provided to all members of an MCF network.

## MCF Library

The MCF library includes functions for the manager (PPE) and the worker (SPE). A developer writes the manager side of an application using the manager API and the worker side using the worker API. The manager automatically loads the worker kernel when the MCF network is initialized. A 12-KB worker kernel is loaded once into each worker in an MCF network. The worker kernel allows the workers to participate in the MCF network, enabling synchronization, communication, shared allocations, and quick loading of worker tasks.

## MCF System Requirements

**Processor** Cell BE processor

### Operating system

Yellow Dog Linux® 2.6.14-1.ydl.2cell from Terra Soft Solutions

Patches and downloads from Barcelona Supercomputing (see [www.bsc.org.es](http://www.bsc.org.es)):

SPUFS

libspe (version 1.0 or higher)

Note: All software dependencies are pre-installed on Mercury-supplied Cell-based systems, so that developers can begin immediately to focus on application development.

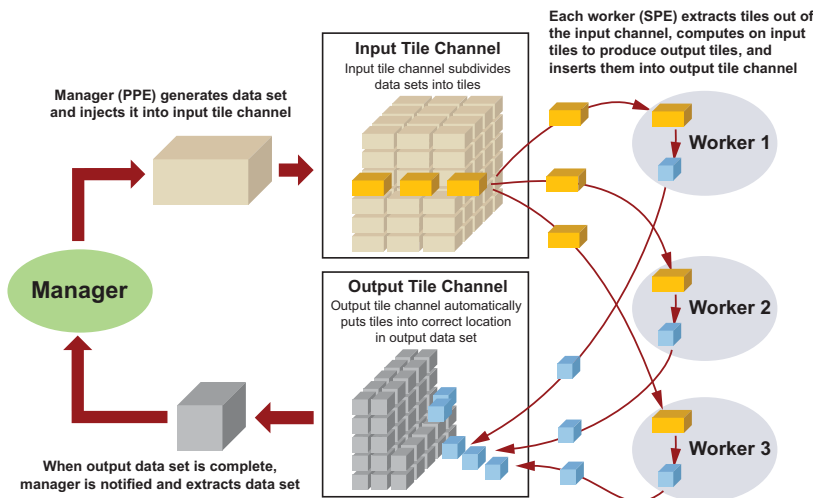


Figure 3. MCF tile channel

Cell Broadband Engine and Cell BE are trademarks of Sony Computer Entertainment, Inc. MultiCore Plus and Challenges Drive Innovation are trademarks of Mercury Computer Systems, Inc. All other products mentioned may be trademarks or registered trademarks of their respective holders. Mercury Computer Systems, Inc. believes this information is accurate as of its publication date and is not responsible for any inadvertent errors. The information contained herein is subject to change without notice.

Copyright © 2006 Mercury Computer Systems, Inc.

469.01E-0506-DS-MCFramework

Computer Systems, Inc.  
**MERCURY**  
Challenges Drive Innovation™

### Corporate Headquarters

199 Riverneck Road  
Chelmsford, MA 01824-2820 USA  
+1 (978) 967-1401 • +1 (866) 627-6951  
Fax +1 (978) 256-3599  
[www.mc.com](http://www.mc.com)

### Worldwide Locations

Mercury Computer Systems has R&D, support and sales locations in France, Germany, Japan, the United Kingdom and the United States.

For office locations and contact information, please call the corporate headquarters or visit our Web site at [www.mc.com](http://www.mc.com).