Exploring the Cell with HPEC Challenge Benchmarks

Sharon M. Sacco, Glenn Schrader, and Jeremy Kepner
MIT Lincoln Laboratory
{ssacco, gschrad, kepner}@ll.mit.edu

Abstract

New architectures require thoughtful testing to explore the performance potential of the processor. Here the HPEC Challenge Benchmarks are used to explore the Cell processor to understand the realistic performance as well as how that performance can be attained. More specifically, for the time domain FIR benchmark, we find that a significant fraction (86%) of the peak performance can be achieved (as expected). However, this performance does come at significant coding cost.

Introduction to Cell

The Cell processor, a joint venture among IBM, Sony, and Toshiba, is an exciting, commercially available multi-processor on a single chip. A simple block diagram of its configuration is shown in Fig. 1.

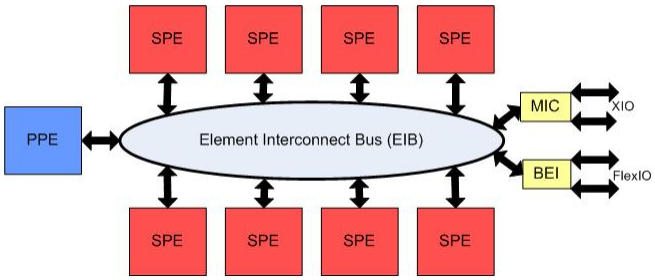

Fig. 1 Block Diagram for Cell Processor (adapted from IBM's "Cell Broadband Engine Hardware Initialization Guide")

Each Cell processor consists of a PowerPC Processing Element (PPE), eight Synergistic Processing elements, an Element Interconnect Bus (EIB), a Memory Interface Controller (MIC) with XIO, and a Broadband Engine Interface (BEI) with FlexIO.

The PPE has a 64-bit PowerPC with 32 KB L1 data and instruction caches and a 512 KB unified L2 cache. The L1 instruction cache is two-way set associative. The L1 data cache is four-way set associative and write through. The L2 cache can be configured to be eight-way set associative and write back. Cache lines in all caches are 128 bytes. The PPE can read 32-bytes per cycle and store 16-bytes per cycle.

The PPE also supports the Vector/SIMD Multimedia Extensions. There are some subtle differences in this implementation when compared to other implementations. This version does not generate floating point exception interrupts, even when JAVA mode is selected.

Each SPE consists of the Synergistic Processing Unit (SPU), 256KB of local store (LS), and a Memory Flow Controller (MFC) that provides an interface between the SPU and anything connected to the EIB. The MFC has built in DMA engines that can deliver information to or from main memory or other processors. For general processor to processor DMAs, memory must be aliased.

The SPU is a vector computational unit with 128 16-byte SIMD registers. While there are similarities to the vector unit on the PPE, they are not the same. Each has a unique instruction set with little exact overlap. While the PPE vector unit is stronger in permutation and summing over registers, the SPU has a rich set of instructions with immediate values. The SPU supports all data types found on the PPE's vector unit plus 64-bit floating point. However, there is very limited support for 8-bit integer.

The SIMD register set is the universal register set for most things on the SPU. As a result, the instructions support scalar operations such as addressing for loads and stores, loop control, and conditional results in the SIMD registers. Generally, these scalar quantities will occupy a register by themselves.

The EIB consists of four ring buses, two running clockwise and two running counter clockwise. Each ring is 16-bytes wide and can transfer 128 bytes at a time. The EIB's maximum bandwidth is 96 bytes per cycle. It supports up to 100 outstanding DMA transfers from the SPEs at any given time.

HPEC Challenge Benchmark Suite

The HPEC Challenge Benchmark Suite[5] is a publicly available collection of common signal processing algorithms to use to compare HPEC systems. It consists of eight kernel benchmarks (finite impulse response filtering (FIR), QR decomposition (QR), singular value decomposition (SVD), constant false-alarm rate detection (CFAR), pattern matching (PM), genetic algorithm (GA), data base (DB), corner turn (CT)) as well as a SAR System Benchmark. All of these benchmarks have their roots in DARPA's Polymorphous Computing Architecture (PCA) or High-Productivity Computing Systems (HPCS) programs.

Time Domain FIR on a Single SPU

Marketing brochures always quote a theoretical maximum speed for a processor based on the number of floating point operations that can be started in a single cycle times the frequency of the processor. Most processor designs cannot sustain these speeds other than for a few cycles at most.

This work is sponsored by AFRL under Air Force contract FA8721-050C-0002. Opinions, interpretations, conclusions and recommendations are those of the author and not necessarily endorse by the United States Government.

The question becomes how do we measure practical speeds that are the best?

Time domain FIR filters are the most efficient algorithms on floating point processors given that the numbers of additions and multiplies are nearly the same. This is the heart of the time domain version of the HPEC Challenge benchmark FIR. The code for the Cell SPU was written for split complex format in three versions, simple scalar C, a simple version using the SIMD C extensions, and a highly optimized hand coded assembly version. This format was chosen since it does not require extra data movement to separate real and imaginary parts. Since the purpose of this test was to measure the practical maximum performance of the SPU, only the FIR was timed. The number of FIR filters was chosen not to exceed the SPU local memory.

Coding productivity is a requirement for any processor. If coding is too difficult, the processor will require serious support from professional algorithm groups. Users who do code in these circumstances are less productive than for processors well designed for compilers. Fig. 2 describes the ease of coding for the Cell SPU. The lines of code for each style of time-domain FIR do not include comments or blank lines. Note that the SIMD C and assembly programs have size restrictions. As another measure of coding productivity, the units of time to describe how long it takes to do a task are given for design, coding and debugging for each implementation.

The efficiency of the code (the factor of peak performance) is also included. This was measured on a Mercury "Cell Technology Evaluation System" running at 2.4 GHz. The clock used for timing was the decrementer register on the SPU. This clock was chosen for its proximity to the calculation which reduces the overhead in reading it. Due to the small size of the LS, the number of filters specified in the time domain FIR benchmark could not be supported. The efficiency was measured for a case with kernel size 12, input vector size 1024, and two filters. This was repeated 10 times to form the timing.

	Lines of Code	Design Time	Coding Time	Debug Time	Efficiency of Code
C	33	Minute	Minute	Minute	.014
SIMD C	110	Hour	Hour	Minute	.27
Hand coding	371	Hour	Day	Day	.86

Fig. 2 Ease of Coding and Performance Metrics for single SPU Time Domain FIR

Note that the efficiency of the hand coding is 14% lower than the theoretical maximum limit. Most of this is due to an inherent inefficiency in the architecture of the Cell. In the Cell, at most only a single computation can be started on any given cycle. The consequence of this design choice is that address updates and loop counters cannot be covered by floating point operations. Loop unrolling will minimize this effect, but it will always be visible.

Parallelizing the Time Domain FIR

The time domain FIR will be parallelized across multiple SPEs. This decomposition is known to be "embarrassingly parallel". However, despite the simplicity of the decomposition, how that is implemented and the resulting performance based on those choices will be explored. How many processors for a given size, how does the DMA size affect the performance, and can the DMAs be covered by computation are all important questions that demand exploration.

Other Benchmarks

Other HPEC Challenge Benchmark kernels will also be examined. These will include the frequency domain FIR, CFAR and PM. These benchmarks should verify the lessons from time domain FIR and perhaps expand understanding of the Cell.

Cell Lessons Learned

In performing these explorations, much can be learned about the processor. The most noticeable fact is that although the SPU can process normal "vanilla" C code, it doesn't do it well. Vectorization techniques will improve performance noticeably, even if limited to simple C extension programs. However, approaching the theoretical limit of performance is limited by the architecture.

Maximizing the use of the SPUs will also require parallel techniques. Given the small size of the local stores and the number of available SPUs, an application of any reasonable size will benefit from optimizing DMA techniques as well as spreading the application over many processors, if possible.

Timers are available on both the PPU and the SPUs. Which clocks are chosen for timing will depend on the locality of the code that is timed, the time length, and the existence of other applications such as profilers that access a particular clock.

References

[1] International Business Machines Corp, Sony Computer Entertainment Corp., and Toshiba Corp., "Cell Broadband Engine Architecture", v1.0, © 2005.

[2] International Business Machines Corp, Sony Computer Entertainment Corp., and Toshiba Corp., "Cell Broadband Engine Hardware Initialization Guide", v. 1.3, © 2006.

[3] International Business Machines Corp, Sony Computer Entertainment Corp., and Toshiba Corp., "Cell Broadband Engine Programming Handbook", v. 1.0, © 2006.

[4] International Business Machines Corp, Sony Computer Entertainment Corp., and Toshiba Corp., "Synergistic Processor Unit Instruction Set Architecture", v. 1.1, © 2006.

[5] Ryan Haney, Theresa Meuse, Jeremy Kepner and James Lebak, "The HPEC Challenge Benchmark Suite", High Performance Embedded Computing (HPEC) Workshop, Lexington, MA. 20 – 22 September 2005.