Synthetic Aperture Radar Backprojection on Sony PlayStation 3 Cell Broadband Engine and Intel Quad-core Xeon

Mark Backues[a], Uttam Majumder[b], Daniel York[c], Michael Minardi[b]
[a]SET Corporation, 1005 N. Glebe Rd. Arlington, VA 22201
[b]The Air Force Research Laboratory, Sensors Directorate, 2241 Avionics Circle, Wright-Patterson Air Force Base, OH 45433
[c]SOCHE, Miami Valley Research Park, 3155 Research Blvd.,Suite 204 Dayton, OH 45420-4015

ABSTRACT

A single-precision floating-point SAR back-projection algorithm has been implemented and optimized on a PlayStation 3 Cell Broadband Engine. The same algorithm was also ported to a server with two Intel Quad-Core Xeons. Speed performance was very similar on the two systems, with performance on the two 1.6GHz Quad-Core Xeons slightly faster than on the 3.2GHz Cell.
.

Keywords: Synthetic Aperture Radar (SAR), Backprojection, Multicore Processor, Cell Broadband Engine

1. INTRODUCTION

Cell Broadband Engine has generated interest as a possible cost effective real-time signal processing solution [1]. As part of a larger evaluation of its potential, AFRL (Air Force Research Laboratory) has funded an evaluation of its performance on a selected SAR (Synthetic Aperture Radar) imaging algorithm. Of interest are the processor's computational performance, I/O ability, power consumption, cost, and ease of programming. Only computational performance and ease of programming will be discussed here. The primary goal has been to perform an independent and balanced comparison with a comparably optimized code on another current microprocessor. This has required a fairly high degree of optimization, in order for the comparison to be meaningful. However, the absolute performance of the implementation, and its comparability with other studied implementations, has been less of a focus than the quality of the comparison between processors.

Back-projection is of interest as an alternative to the polar formatting imaging algorithm for SAR [2], [3]. Back-projection is advantageous in that it can naturally handle high wave-front curvature in wide angle applications, and can easily accommodate non-planar image surfaces. Its significant disadvantage is that in its simplest form it is order n^3, where n is the image diameter in pixels. This contrasts to order $n^2 \log n$ for polar formatting. There are more complicated 'fast' back-projection variations that can yield better performance than order n^3, with some tradeoff in image quality. For the sake of this evaluation however, a simple single-precision 'full' back-projection algorithm was chosen, since other more sophisticated variations are similar at the core.

Correspondence POC: Uttam K. Majumder, Uttam. Majumder@wpafb.af.mil

PlayStation 3 was chosen as a platform for testing the Cell Processor because of its low cost and ready availability. Cell blade servers are also available, but are considerably more expensive. E5310 “Clovertown” Xeons were chosen for comparison with the Cell Processor, because they are relatively high performance, are available in a variety of practical platforms at a relatively low cost, and an ASUS DSBV-D server board with two of them was already on hand. The Xeon implementation was also ported to a dual-core Opteron cluster, where it performed fairly well. The Opteron performance was not explored in as much depth however, and will not be discussed here, because the hardware on hand was not as current, and extensive performance comparisons between Intel and AMD processors are already available elsewhere.

2. PERFORMANCE

Figure 1 shows a speed comparison between the current PlayStation 3 Cell Processor and Xeon implementations. The execution time on dual, Quad-Core Xeons (ASUS DSBV-D) is on average 84% of the execution time on the PS3 Cell Processor, for the image sizes tested. The Cell Processor clock speed is 3.2GHz, twice the 1.6GHz Xeon speed on the ASUS board. A system with higher Xeon clock speeds would perform proportionally better.

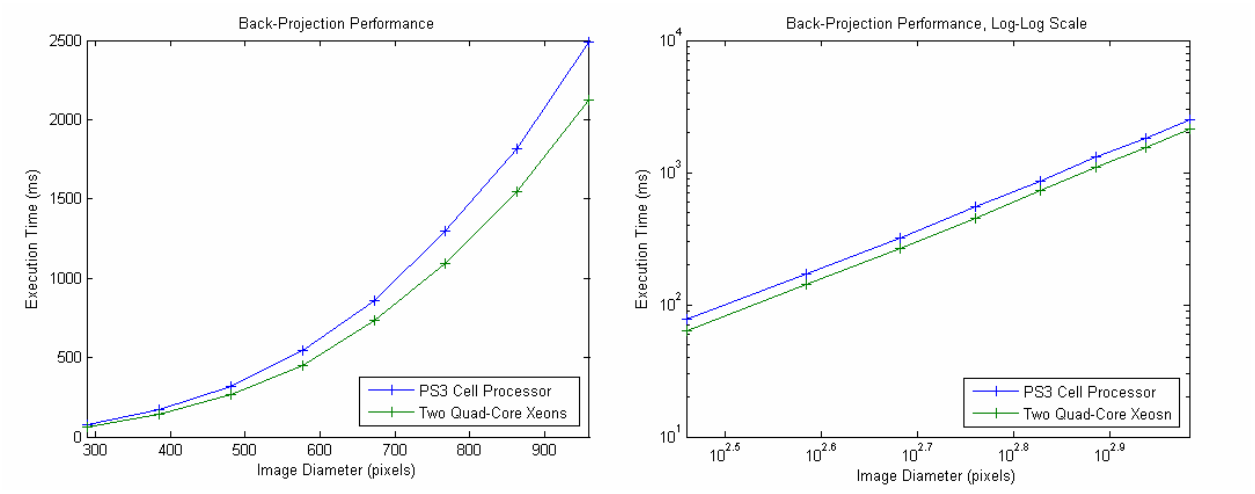

Figure 1: Execution time on dual Quad-Core Xeons is about 84% what it is on the PS3 Cell Processor. The log-log plot illustrates the order n^3 *nature of both implementations.*

For the sake of these comparisons, the number of input pulses was equal to the image diameter in pixels, and the number of range bins was 4 times that. Output correctness was verified with both real and synthetic data.

The data transfers between the SPU and the PPU’s were not double buffered for the back-projection algorithm, but both calculations and tests show this inefficiency to be negligible (less than 1ms for hundreds of ms of computation time). Double buffering would however be important for other applications that are lower complexity than back-projection.

The peak performance spec for two 1.6GHz Quad-Core Xeons is 51.2 GFLOPS. That is the clock speed times the number of cores, times 4 floating point operations per cycle. The spec for a

PlayStation 3 Cell Processor is 204.8 GFLOPS. That is the clock speed times 8 (for the PPU and 7 function SPUs), times 4 floating point operations per cycle.

Why is the performance for the back-projection algorithm so similar, when Cell has 4 times the peak theoretical FLOPS? There are a variety reasons for this having to do with architectural differences. One is the availability of a square root instruction on Xeon. Another reason is the significance of data indexing, moving, and shifting. The relative performance of the two processors depends in large part on how efficiently they perform these tasks, rather than on how many arithmetic operations can be issued per cycle, which is what the theoretical peak numbers depend on. And for the back-projection implementation on the Cell, indexing into the range data and the trigonometric table required considerably more instructions than on the Xeon.

A third reason is the dependence of the Cell peak performance number on efficient use of the multiply-add instruction, which is what would enable two operations to be performed on four floats per clock cycle on each SPU. (Each PPU has two pipelines, but only the 'even' pipeline handles arithmetic operations, so the presence of two pipelines does not again double the theoretical peak GFLOPS.) Many algorithms can not be decomposed into a series of multiply-adds, in which case only half the ideal performance is attainable. A similar situation exists for Xeon, where instructions take two cycles to execute, but instructions which use different execution units can be interleaved so that one is performed per cycle. In cases where instructions must use the same execution unit, only half the performance is attainable. The situation for Xeon is more flexible than for Cell however. For Cell, attaining the maximum number of operations per cycle requires operations to be available in the same instruction, such as multiply-add or negative-multiply-subtract. For Xeon, it only requires that they don't share the same execution unit, which allows the pairing of a much wider variety of operations.

A fourth reason is that only six of the PS3 Cell's 7 functioning SPUs are user accessible. So total SPU processing power would be 25% higher on a Cell with 8 accessible SPUs. And finally, unlike the SPUs, the PPU is not designed primarily for pipelined vector computations, though the theoretical peak figure is the same as for a SPU. For the back-projection implementation, performance on the PPU is less than a quarter of the performance on a single SPU. One reason for this is relatively poor performance of the sqrtf4() function on the PPU, though the SPU performed significantly better on all major parts of the back-projection loop. (A cycle level performance analysis of the PPU back-projection implementation has not been undertaken, because optimization relevant documentation for the PPU is more limited, and the timing tool used for the SPU is not applicable to the PPU.)

Worth noting is that Cell Processor would likely have fared significantly worse had double precision range and phase calculations been used for better image quality. Double precision performance would of course be 2*x* worse on both Cell on Xeon on account of the reduced data parallelism when using SIMD instructions. But beyond that, the Cell Processor double precision floating point instructions have 13 cycle latencies, including unavoidable six-cycle stalls. So this would have, for example, resulted in an additional 7*x* degradation in performance for multiply-add instructions. On Xeon, in contrast, instruction latency and throughput is unchanged between single and double precision for simple operations like multiply and add. Latency and throughput do degrade for complex Xeon double precision operations. For example, the double precision SIMD square root requires 62 cycles (both latency and throughput) in contrast to 32 cycles for single precision. But this is only an additional 2*x* degradation.

3. RESEARCH SUMMARY

Below is a summary of significant development and performance similarities and differences between Cell Processor and multi-core Xeon. Many of the same statements would also be valid when comparing other processors like AMD Opteron with Cell.

Similarities:

1. Both the Cell Processor and multi-core Xeon require multi-threaded parallelization.
2. Currently, both benefit substantially from manual SIMD vectorization and other manual optimizations.
3. Blade servers are available for both alternatives. (The 1GB Ethernet connection between PlayStations would be a serious issue for networking large numbers for most real-time processing applications.)
4. Speed performance is similar for both alternatives (1 3.2GHz Cell vs two 1.6GHz Quad-Core Xeons), at least for the back-projection application.

Differences:

1. Cell Processor requires explicit DMAs between the PPU and SPU, with run-time errors occurring if rules on data sizes and byte alignment are not followed. For problems where DMA time is significant, double-buffering is required for optimal performance.
2. More optimization has been required for efficient execution on Cell Processor, at least for this back-projection application. In particular, efficiently moving data into registers was much more difficult on Cell compared to Xeon.

Cell Processor would likely fare worse than it has here in a similar comparison where double precision operations are needed.

4. FUTURE RESERACH

It would be informative to repeat the back-projection comparison using double precision floats to represent ranges. This would yield more information on how well Cell performs relative to Xeon where double precision computations are involved, and would as a byproduct result in a more useful code.

It would also be worthwhile to perform a similar analysis on some other algorithms. The back-projection algorithm is exceptional in that relatively high order n^3 complexity makes data I/O issues relatively unimportant. Other useful lower order radar processing algorithms, such as digital spotlighting, would therefore also be of interest as test cases. Algorithms in which simple arithmetic instructions like multiply-add play a larger role, and where data parallel memory access is less of a problem, might also be interesting to look at. As before, the focus would be on making a balanced comparison between two equally optimized implementations on current processing technology, rather than on improving upon existing algorithm implementations.

REFERENCES

[1] "Cell Broadband Engine Technology", *http://www-03.ibm.com/technology/Cell/pdf/CellBrBandEngineWhitePaper.pdf*, 2007.

[2] Desai, Jenkins "Convolution Backprojection Image Reconstruction for Spotlight Mode Synthetic Aperture Radar", *IEEE Transactions on Image Processing*, October 1992.

[3] Jakowatz, Thompson, Doren, "Performance of the Polar Formatting Algorithm for SAR Image Formation on Wide Aperture Collections", *http://spib.rice.edu/DSP2000/submission/DSP/papers/paper089/paper089.pdf*, May 2001.

[4] "Software Development Kit for Multicore Acceleration, Version 3.0, Programming Tutorial", *http://www01.ibm.com/chips/techlib/techlib.nsf/techdocs/FC857AE550F7EB83872571A80061F788*, October 2007.

[5] "Intel® 64 and IA-32 Architectures Software Developer's Manual", *http://www.intel.com/products/processor/manuals/index.htm*, November 2007.