Extending the VSIPL standard to other precisions: Gaining the full performance of current processors

Albert Garrett
Verari Systems Software, Inc.
110 12th Street North Suite D103
Birmingham, AL 35901
(205) 397-3159
agarrett@verarisoft.com

Anthony Skjellum
Verari Systems Software, Inc.
110 12th Street North Suite D103
Birmingham, AL 35901
(205) 397-3141
tony@verarisoft.com

Abstract

Given that execution time is a key value in high-performance embedded computing, and that the adoption-rate of COTS middleware continues to increase in response to schedule constraints, the authors demonstrate that by extending the offering of the VSIPL standard to other precisions, significant performance improvements can be made in execution time without significant investments in new development or abandoning the standard API. Further, this paper offers design constraints for understanding when such steps can be taken and how best to utilize the full capabilities of the processors already in the field. Standard VSIPL API plus "hints" will complement design guidelines for when to use the accelerated but limited precision alternatives of key operations.

Introduction

The authors first establish a baseline of performance for the single-precision, FFT algorithm on Pentium 4 and PPC74xx processors as a function of problem size. This baseline is well-established by both free and commercial implementations of the FFT in the VSIPL framework. Next, the potential advantages of computing the FFT with integer precisions are discussed. The potential pitfalls of fixed-point operations on these processors is also discussed, including dynamic-range growth and quantization loss. Finally, the integer-based performance of the FFT is presented. Logical extensions to the standard are suggested in terms of API to directly include integer precisions for signal processing, particularly emphasizing 16-bit precisions, but with equivalent impact on other integer precisions.

Next, the paper describes a special, low-precision version of the single-precision FFT that works with integer precision underlying it, and exploits the existing VSIPL single-precision API plus a hint. Auxiliary information needed to assure limited distortion is provided as guides to users. Given the enhanced performance afforded by lower-precision computation, primarily owing to greater concurrency in the vector units and lower memory bandwidth requirements for smaller operands, users are offered a convenient alternative for problems that are less demanding. Opportunities to grow the applicability of these methods by additional scaling and windowing are mentioned.

Previously, one of the authors and others explored lower precision VSIPL for a Java-enabled, ARM-based PDA[1]. These processors were incapable of floating point, but fixed-point libraries (still relatively slow) could be used validly to emulate floating-point in VSIPL and use VSIPL from Java as a side effect. This effort complements the previous one by intentionally using integer operations for specific classes of problems stored in floating point format, but subject to limited dynamic range.

Other areas of potential enhancement include the acceleration of double-precision operations, using single-precision plus iterative refinement approaches for either matrix-matrix operations, or some factorizations. Previous classical work on this [3], plus recent work by others on SIMD processors [4], with applicability to VSIPL is offered.

Disclaimers and footnotes.

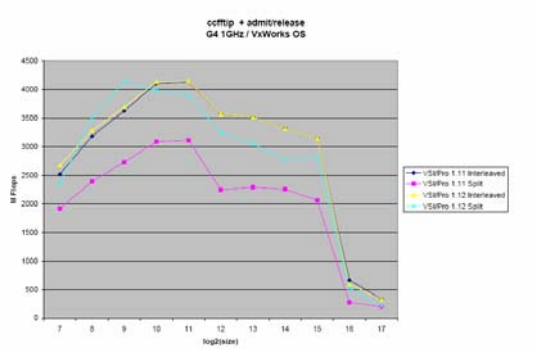

Figure 1: Typical FFT Performance plot, MFlops v log(N).

Other applications of fixed-point, and block-floating-point [2] in VSIPL are suggested, including opportunities to extend the standard. Issues of SIMD formulation for these techniques are mentioned.

Issues of precision and portability with specialized precisions are considered, as these were previous stumbling blocks to standardizing special precisions in the standard.

References

[1] Alford, Torey; Shah, Vijay; Skjellum, Anthony; Younan, Nicolas; Taylor, Claybourne. Interfacing Java and VSIPL Applications. Concurrancy – Practice and Experience. Vol 17 number 8, 2005. pp 919-940.

[2] Mitra, Abhijit. On Roundoff errors in Block-floating-point Arithmetic. IETE Journal of Research, vol 50 no5, Sept 2005.

[3] Oppenheim and Weinstien. "Effects of Finite Register Length in Digital Filtering and Fast Fourier Transform," Proc. IEEE, August 1972, pp. 957-76.

[4] Buttari, Alfredo; Dongerra, Jack; Kurzak, Jakub; Langua, Julie; Luszczek, Piotr. "Revising Iterative Refinement for Linear Systems." http://www.cs.utk.edu/~julie/iter-ref/