



Intel[®] Technology Journal

Designing Technology with People in Mind

IEEE 754R Decimal Floating-Point Arithmetic: Reliable and Efficient Implementation for Intel[®] Architecture Platforms

IEEE 754R Decimal Floating-Point Arithmetic: Reliable and Efficient Implementation for Intel[®] Architecture Platforms

Marius Cornea, Software & Solutions Group, Intel Corporation
John Crawford, Digital Enterprise Group, Intel Corporation

The IEEE Standard 754-1985 for Binary Floating-Point Arithmetic [1] is being revised. An important addition in the current draft [2] is the definition of decimal floating-point arithmetic. The primary motivation is that decimal arithmetic makes numerical calculations more human-friendly. Results will be as people expect them, identical to what would be obtained using pencil and paper. Decimal arithmetic also provides a robust, reliable framework for financial applications that are often subject to legal requirements concerning rounding and precision of the results in the areas of banking, telephone billing, tax calculation, currency conversion, insurance, or accounting in general. The binary floating-point arithmetic that computers use does not always satisfy the existing accuracy requirements. For example, $(7.00 / 10000.0) * 10000.0$ calculated in single precision is 6.9999997504, and not 7.00. The underlying cause is that most decimal fractions, such as 0.1, cannot be represented exactly in binary floating-point format. The IEEE 754R standard proposal attempts to resolve such issues by defining all the rules for decimal floating-point arithmetic in a way that can be adopted and implemented on all computing systems in software, hardware, or a combination of the two.

In the past, the lack of a good standard for decimal computations has led to the existence of numerous proprietary software packages for this purpose, each with its own characteristics and capabilities. In planning to improve on this and make Intel one of the early adopters of the proposed Standard 754R for decimal floating-point arithmetic, we have developed new algorithms along with correctness proofs and we have used them to implement in software a decimal floating-point library [3, 4, 5]. Standardization committees for high-level languages such as C and C++ are already developing plans to add decimal floating-point support to these languages. Once this happens, our new software package for decimal floating-point arithmetic will be available to be used as a computation engine by compilers, and it should constitute an attractive option for various financial computations. A logical question is whether a software implementation of the decimal arithmetic will be fast enough for financial applications, or whether a hardware implementation is needed. A close look at decimal-intensive applications today reveals that most spend between 1% and 5% of their execution time performing decimal operations while the rest is spent in integer, binary floating-point, memory, I/O, or other operations. A software implementation of the IEEE 754R decimal floating-point arithmetic should therefore be sufficient for the foreseeable future, compliance with the standard being far more important than raw performance in this case.

A decimal floating-point number n defined by the IEEE 754R standard proposal can be represented as $n = \pm C \cdot 10^e$ where C is a positive significand with at most p decimal digits, and e is an integer exponent. The basic decimal floating-point formats with their precision and exponent range are shown in the following table:

Table 1: Decimal floating-point formats with their precision and exponent range

	decimal32	decimal64	decimal128
Precision p	7	16	34
Exponent range $[e_{\min}, e_{\max}]$	[-95, +96]	[-383, +384]	[-6143, +6144]

Two equivalent encoding methods are described in the IEEE 754R proposal: the decimal encoding method [2] and the binary encoding method [2, 6]. The main characteristic of the decimal encoding is that it uses the Densely Packed Decimal (DPD) method, which compresses three consecutive decimal digits in 10 bits. The binary method encodes the entire decimal significand as a binary integer, and hence it is also referred to as Binary Integer Decimal, or BID.

The new algorithms for decimal floating-point arithmetic implemented in the Intel IEEE 754R Decimal Floating-Point Library are equally applicable to both encodings, although they are more efficient with the binary (BID) encoding.

Selected performance results expressed in numbers of clock cycles for the Intel® BID Decimal Floating-Point Library are shown in the figure below, where average and median values are given after subtracting the call overhead. Clock cycle counts are shown for decimal64 and decimal128 format addition, multiplication, division, square root, conversion to string, conversion from string, and quantize operations. An Intel® Xeon® processor 5100[‡] series 3.0 GHz was used for these measurements. The library is currently being tested in Windows*, Linux*, OS X*, and HP-UX* on IA-32, Intel® 64[†], and IPF platforms.

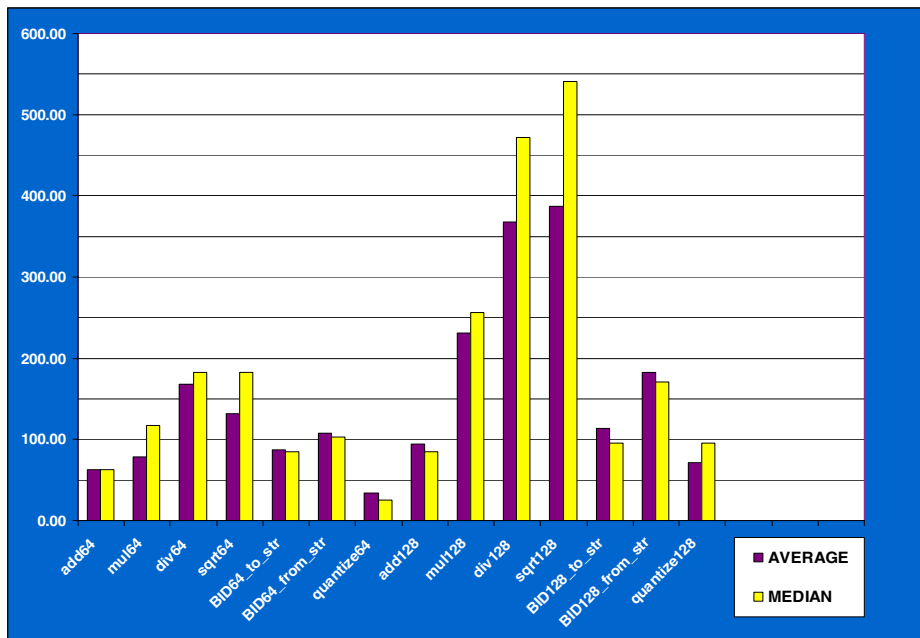


Figure 1: Performance Results: Clock cycle counts for a subset of functions from the Intel® IEEE 754R BID Decimal Floating-Point Library, measured on an Intel® Xeon® processor 5100 series 3.0 GHz

We expect that properties and algorithms used in the new library for decimal floating-point arithmetic can be applied as well for a hardware implementation, with re-use of existing circuitry for binary operations. We believe that our library work represents an important step toward reliable and efficient implementations of the IEEE 754R decimal floating-point arithmetic on Intel® Architecture platforms.

REFERENCES

- [1] "IEEE Standard 754-1985 for Binary Floating-Point Arithmetic," *IEEE*, 1985.
- [2] "DRAFT Standard for Floating-Point Arithmetic P754," <http://754r.ucbtest.org/drafts/754r.pdf>*
- [3] "Software Implementation of the IEEE 754R Decimal Floating-Point Arithmetic," Marius Cornea and Cristina Anderson, in *Proceedings, Real Numbers and Computers Conference*, 2006.
- [4] "Software Implementation of the IEEE 754R Decimal Floating-Point Arithmetic," Marius Cornea, Cristina Anderson, Charles Tsien, in *Proceedings, ICSoft Conference*, 2006.
- [5] Reference Software Implementation of the IEEE 754R Decimal Floating-Point Arithmetic, <http://www3.intel.com/cd/software/products/asm-na/eng/219861.htm>
- [6] Intel and Floating-Point, <http://www.intel.com/standards/floatingpoint.pdf>

AUTHORS' BIOGRAPHIES

John H. Crawford is an Intel Fellow at the Intel Corporation, Santa Clara, California, where he investigates emerging technology directions and issues for server products. Mr. Crawford was the Chief Architect of both the Intel386™ and Intel486™ microprocessors, and co-project manager for the Pentium® processor. He managed the joint Intel/HP team that defined the Itanium® Processor Family instruction set architecture, and directed aspects of Itanium processor product development. Mr. Crawford was awarded the ACM/IEEE Eckert-Mauchly Award in 1995 and the IEEE Ernst Weber Engineering Leadership Recognition in 1997. He was elected to the National Academy of Engineering in 2002. Mr. Crawford received an Sc.B. degree in Computer Science from Brown, and a M.S. degree in Computer Science from the University of North Carolina, Chapel Hill. His e-mail is john.h.crawford at intel.com.

Marius Cornea is a principal engineer in Intel's Software & Solutions Group. He received a master's degree in Electrical Engineering from the Polytechnic Institute of Cluj, Romania and a Ph.D. degree in Computer Science from Purdue University in West Lafayette, IN. Since joining Intel in 1994, his work is related to scientific computation, design and development of numerical algorithms, floating-point emulation, exception handling, and new floating-point instruction definition and analysis. His e-mail is marius.cornea at intel.com.

‡ Intel processor numbers are not a measure of performance. Processor numbers differentiate features within each processor family, not across different processor families. See http://www.intel.com/products/processor_number for details.

† 64-bit computing on Intel architecture requires a computer system with a processor, chipset, BIOS, operating system, device drivers and applications enabled for Intel® 64 architecture. Processors will not operate (including 32-bit operation) without an Intel® 64 architecture-enabled BIOS. Performance will vary depending on your hardware and software configurations. Consult with your system vendor for more information.

Copyright © Intel Corporation 2007. All rights reserved. Intel, Xeon, Pentium, Intel386, Intel486, and Itanium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

* Other names and brands may be claimed as the property of others.

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL® PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER, AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

Intel may make changes to specifications and product descriptions at any time, without notice.

This publication was downloaded from <http://www.intel.com>.

Legal notices at <http://www.intel.com/sites/corporate/tradmarx.htm>.

THIS PAGE INTENTIONALLY LEFT BLANK

For further information visit:

developer.intel.com/technology/itj/index.htm