Hardware Design for Decimal Multiplication

Mário P. Véstias
INESC-ID/ISEL/IPL, Portugal

Horácio C. Neto
INESC-ID, Instituto Superior Técnico, Universidade de Lisboa, Portugal

INTRODUCTION

The IEEE-754 2008 standard for floating point arithmetic has definitely dictated the importance of decimal arithmetic. Human-centric applications, like financial and commercial, depend on decimal arithmetic since the results must match exactly those obtained by human calculations. Decimal multiplication is a fundamental operation utilized in many algorithms, including algorithms for decimal division. Decimal multiplication is more complicated than binary multiplication due to the inherent difficulty to represent decimal numbers using a binary number system. Both bit and digit carries, as well as invalid results, must be considered in decimal multiplication in order to produce the correct result.

This article focuses on algorithms for hardware implementation of decimal multiplication. We describe decimal fixed-point and floating-point multiplication, including iterative and parallel solutions.

BACKGROUND

Usually, humans perform arithmetic operations by hand using decimal arithmetic. However, most computers do it with binary arithmetic. It means that performing decimal operations in a computer without support for decimal arithmetic is subject to errors from representing decimal numbers, converting them and rounding. In fact, it is easy find decimal numbers that cannot be represented exactly in binary format (e.g., 0.1). Several examples exist where errors due to binary calculation of decimal numbers are obtained. A clarifying example came from the Vancouver Stock Exchange (Quinn, 1983), where due to rounding errors an initial index value of 1000.000 dropped to 574.081 instead of the correct result of 1098.892.

In fact, the business and commercial markets were one of the triggers for the importance of decimal computer arithmetic since many commercial databases have more than 50% of the numerical data represented in decimal (Tsang & Olschanowsky, 1991). In these cases, to avoid errors with undesirable consequences it is important to have a complete system to support decimal arithmetic.

At the era of electronic computers, both binary and decimal arithmetic functions were considered. We had computer systems, like the ENIAC (Goldstine & Goldstine, 1996) and IBM 650 (Knuth, 1986) implementing arithmetic functions in decimal, and others like EDSAC (Wilkes, 1997) and EDVAC (Williams, 1993) that adopted binary based arithmetic implementations. Both arithmetic systems were still considered after the advent of transistorized computers with decimal numbers represented with four bits following different representations, like in Binary-Coded Decimal (BCD) format. However, soon binary arithmetic was adopted by most computer systems since at that time scientific computing, whose operations could be more efficiently implemented in binary, were more in demand than financial computing that requires decimal arithmetic to avoid costs from representation errors. So, binary became very popular, while decimal was supported only by some computers in the 1960s and 1970s.

Precise decimal arithmetic operations with binary based computing systems are done with software. In some cases, these binary-based computing systems include some specific hardware instructions that are hardware supported and so software algorithms can take advantage of them to speed-up execution. Several languages include primate decimal datatypes, including Ada, COBOL, and SQL. Several other languages support the GDAS (General Decimal Arithmetic Specification) (Cowlishaw, 2008), including the IBM C DecNumber Library (Cowlishaw), the Java BigDeci-

DOI: 10.4018/978-1-4666-5888-2.ch538

mal (Sun Microsystems), Eiffel Decimal Arithmetic (Crismer), Python Decimal (Batista), among others. Decimal floating point extensions conforming to the IEEE 754-2008 standard were proposed for C (JTC 1, 2007) and C++ (JTC 1, 2008) languages. These extensions were supported by GNU C compiler 4.2 release. Intel has also developed a decimal floating-point math library (Intel) that implements decimal floating-point arithmetic specified in IEEE 754-2008.

Hardware support for decimal arithmetic is needed if the percentage of time spent executing decimal functions from these software libraries is relevant. Two different perspectives have emerged in the end of the last decade. Wang (Wang, L.-K., et al., 2007), examined several financial benchmarks and concluded that the time spent on executing decimal operations ranged from 33.9% to 93.1%. On the contrary, a research from Intel (Cornea & Crawford, 2007) concluded that most commercial applications spend less than 5% executing decimal operations. Therefore, hardware for decimal arithmetic is not a priority in the design of Intel's processors. In fact, Intel x86 processors offer only a set of eight fixed-point decimal arithmetic instructions, and Motorola 68K reduces this set to just five instructions. On the other side, several IBM's processors include a considerable support for decimal arithmetic. The S/390 processor (ESA/390, 2001) includes a dedicated decimal adder to execute decimal fixed point addition, subtraction, multiplication and division. The last two, are executed iteratively using additions and subtractions. The IBM System z9 (Duale, et al., 2007) and System z10 (Schwarz, Kapernick, & Cowlishaw, 2009) already include a decimal floating-point arithmetic unit in conformance with IEEE 754-2008 standard. The GNU C Compiler (GCC) 4.3 Release and several compilers from IBM (e.g., IBM XL C/C++ (IBM)) were extended and developed to utilize the dedicated instructions and hardware units present in these IBM's processors.

DECIMAL MULTIPLICATION

Hardware implementations for decimal multiplication can be classified according to the type of operands to be multiplied as fixed or floating-point, whether the operands are fixed or floating-point, respectively.

Fixed-point multiplication follows generically the typical hand process that starts by generating partial products, followed by reduction of the partial products using decimal addition. The process can be either based on iterative or parallel algorithms. In the iterative approach partial products are generated and accumulated step-by-step in an iterative process, while in the parallel case partial products are generated in parallel and added with an adder tree.

Decimal floating-point multiplication typically uses a fixed-point decimal multiplier to multiply the trailing significant fields, together with exponent addition, rounding and sign calculation, similar to a binary multiplier.

In the following, the basic algorithms and architectures of each type of decimal multiplier are introduced together with state-of-the-art proposals based on each of these types.

DECIMAL FIXED-POINT MULTIPLICATION

Whether in the iterative or in the parallel method, partial product generation in a fixed-point multiplication can be either the result of digit by digit, digit by word or word by word multiplication. In a digit by digit multiplication process each digit of the multiplier is multiplied by a digit of the multiplicand. A faster approach is to multiply each digit of the multiplier by the whole multiplicand, that is, a multiple of the multiplicand is accumulated for each digit of the multiplier. Word by word multiplication is a method that seeks to reduce the number of partial products at the cost of more complex partial product generation.

Digit by Digit Multiplication Design

Different designs are obtained for digit by digit iterative multiplication depending on the how the digits are traversed. One approach consists on multiplying a digit of the multiplier by all digits of the multiplicand from the least significant digit to the most significant digit. Thus, the multiplier must run all digits for each digit of the multiplier (see Figure 1a).

At each step, a new digit by digit product must be added to the accumulated partial product using an n