Special Issue for International Conference of Advanced Materials Engineering and Technology (ICAMET 2013), 28-29 November 2013, Bandung Indonesia

Advances in Environmental Biology, 7(12) October Special Issue 2013, Pages: 3607-3610

Improved Addition and Subtraction in Logarithmic Number System: Technique and Analysis

R.C. Ismail, N. Norzahiyah, A.B. Jambek

School of Microelectronic Engineering, Universiti Malaysia Perlis, 02600 Arau, Perlis.

ARTICLE INFO

Article history:
Received 11 September 2013
Received in revised form 21 November 2013
Accepted 25 November 2013
Available online 5 December 2013

Key words:
Logarithmic Number System (LNS)
Fixed Point (FXP)

ABSTRACT

Logarithmic Number System (LNS) is an alternative beside Floating Point (FLP) when an application requires a large dynamic range in the number processed. It allows simple implementation of multiplication and division using a Fixed Point (FXP) method without rounding error. In contrast, LNS addition and subtraction become more complex procedure. Therefore over the year, difference ways of improving the addition and subtraction function have been purposed. In this paper several techniques have been discussed and analysed such as direct lookup table, interpolation, table partitioning and co-transformation for approximating LNS addition and subtraction.

INTRODUCTION

Usually computer arithmetic unit include four major basic operations such as addition, subtraction, multiplication and division. Commonly, the researcher always chooses FLP methods if application needs a large dynamic range compare to LNS. It's because LNS are not popular as FLP. LNS still in research and also handle exceptional cases involving zeroes, infinities, underflows or overflows [1].

Direct lookup table first suggested by Earl Swartzlander [2] but it seems impractical for long word length numbers, especially 32-bit. Another technique has been introduced known as the linear interpolation method, but it mainly provides for only addition algorithms which require a multiply unit in the hardware system [3]. The other technique is table partitioning. Table partitioning usually join with the interpolation technique with the aim of reducing the size of lookup tables [4-7]. More recently, a technique that can offer considerable memory savings compared to others is a combination of co-transformation and interpolation procedures.

The objective of this paper is to analyze various technique for approximating LNS addition and subtraction. The paper is organized into seven major parts. In Section 2, direct lookup table was reviewed while in section 3, interpolation technique was analyzed. Section 4 and 5 were discussed about table partitioning technique and co-transformation technique. In section 6, the discussion for each technique was performed. Finally in section 7, the results were concluded.

Direct lookup table:

In computer science, a lookup table is an array that replaces runtime computation with a simpler array indexing operation. The savings in terms of processing time can be significant, since retrieving a value from memory is often faster than undergoing an expensive computation or input/output operation [11]. The first LNS architecture for addition and subtraction was introduced by Earl Swartlander in 1975 [2]. The concept was a direct equation (1.0) and (2.0) using a lookup table or us calls read only memory (ROM) based on hardware that cover all the probability value of *sb(r)* and *db(r).*

$$L_1 = x + y$$
$$log_2| L_1 | = log_2 | x + y |$$
$$= log_2 | x (1 + (y / x)) |$$

Corresponding Author: R.C. Ismail, School of Microelectronic Engineering, Universiti Malaysia Perlis, 02600 Arau, Perlis.
E-mail: rizalafande@unimap.edu.my

$$= log_2 | x | + log2 | 1 + (y / x) |$$
$$= log_2 | x | + log2 |1 + log_2 | y | - log2 | x |) |$$
$$= i + log_2 | 1 + 2^{j-i} |$$
$$= i + log_2 | 1 + 2^r |$$
$$= i + sb(r) \quad (1.0)$$

$L_2 = x - y$

$$log_2 | L_2 | = log_2 | x - y |$$
$$= log_2 | x (1 - (y / x)) |$$
$$= log_2 | x | + log2 | 1 - (y / x) |$$
$$= log_2 | x | + log2 |1 - log_2 | y | - log2 | x |) |$$
$$= i + log_2 | 1 - 2^{j-i} |$$
$$= i + log_2 | 1 - 2^r |$$
$$= i + db(r) \quad (2.0)$$

As usual, the implementations of LNS addition and subtractions always have to limit the variable *r* to either positive or negative value. But usual the *r* restricts to negative value because at certain point the function of *sb(r)* and *db(r)* have an output of zero or known as the essential zero.

Interpolation:

Interpolation as a general is finding a point between two known points. It is the other technique to avoid the problem of memory space direct lookup table. But it will increase the delay and also introduces its own error [1]. The direct interpolation technique by Arnold [3] was first introduced to provide only for addition algorithm, *sb* which is require a multiply unit in the hardware system. For the interpolator, *r* is dividing into 2 parts, *rh* and *rl*, hence r = *rh* + *rl*. *rh* encompasses the highest bit of the variable, whereas *rl* is represent the lowest bits [1].
In the general, the direct interpolation can be written as:

$$sb(r) = sb\ (r_h + r_l) \sim sb(r_h) + C(r_h)\ .\ r_l \quad (3.0)$$

Where the lean of $C\ (r_h)$ can be choose from any method such as Lagrange. Memory usage can be reduced by increasing the lower bit, but the accuracy of the approximation decreases too.

Another important interpolator was purposes by Taylor in 1983 [3]. It name linear interpolator. Taylor approximation *sb(r*) as:

$$sb(r) = sb(r_h) + sb\ (r_h).\ r_l \quad (4.0)$$

From this technique, a multiply still required to compute the function. Another disadvantage is this technique is not reasonable for subtraction algorithm. Another suggested interpolation procedure was proposed by Henkel in 1989[9]. This technique also like before, not reasonable for subtraction algorithm.

Table Partitioning:

Usually, partitioning is often combined with an interpolation technique. Instead of using a single uniform partition (direct lookup table approach) [2], the technique can be realized by segregating the ROM into various sizes of interval mapping with the domain function of addition and subtraction algorithms. These intervals are distributed in smaller regions with similar widths of partition endpoints, hence providing substantial saving in ROM area.

Taylor [5] suggested a 20-bits LNS processor using a table partitioning method for both addition and subtraction in 1998.

Another researches, Stouraitis [9] produced an enhanced version of Taylor's architecture by compressing the lookup table address space and inserting pipelining registers in the addition and subtraction data path.

Another technique was presented in 1990 by Lewis [6] using a partitioning procedure with linear interpolation. An integer multiple-of-two format was adopted at each interval of *r* less than -1 for subtraction, an in all cases of region *r* for addition. For subtraction in the region $-1 < r < 0$, the power of two format was proposed. In 1994, Lewis [7] applied the table partitioning concept with an interleaved memory scheme.

Co-Transformation:

Based on the earlier discussion, most of the techniques presented so far has the same problem of solving the *db(r)* function when *r* is close to zero. They incline to be either higher in cost, in terms of lower in accuracy, memory or else. Other techniques which can avoid this problem are using the co-transformation procedure. The

idea is to convert the argument of *db(r)* into modified value that are guaranteed to avoid the singularity of the function [1].

The first important co-transformation technique was known by Coleman in 1995[12], with applying the concept in the region $-0.5 < r < 0$ for the *db(r)* function. For this technique, the need for interpolation in the region $-0.5 < r < 0$ can be eliminated, thus substantially reducing the size and complexity of the lookup tables required. Note for the s*b(r)* function, an interpolation scheme was applied throughout all regions. In 2000, Coleman [13] presented in details the implementation of this co-transformation together with interpolation in a 32-bits system. With significant improvements in accuracy over FLP, a total of 321 kbits were required in order to execute the LNS addition and subtraction. Recently, Coleman [14] conducted an experiment to determine the feasibility of integrating the LNS system into a microprocessor based on the proposal in [13].

A chip of a 32-bits LNS microprocessor, named the European Logarithmic Microprocessor (ELM), was manufactured using 0.18 μm CMOS technology. This was compared with the existing FLP DSP device from Texas Instruments, which has one of the fastest speeds obtainable in 0.18 μm technology. Besides clearly that the results were more accurate, the speed of the ELM was also substantially improved over the FLP device, at 24ns whilst performing addition and direct subtraction, and for subtraction using co-transformation.

A different but related co-transformation technique to Coleman's was given by Arnold in 1999[13]. Unlike Coleman method, which a value at the singularity to a negative argument of *db* that will fall in the region to the left of -0.5, Arnold method avoids the singularity by transforming to a positive argument of *sb* which does not have a singularity. Hence whenever $r > 0$, Arnold's technique is the most appropriate due to the positive value generated for the interpolation after being transform. If $r < 0$, then Coleman's technique is the most natural to adopt because the transformed argument provided to the interpolation is negative. For that reason, Coleman's method is preferable given that LNS researcher tended to apply a negative value of r, since this reduces the ROM size dramatically when approaching essential zero.

Discussion:

The main concern of this paper is to determine which technique that can provide the best approach to approximate LNS addition and subtraction. Through observation and analysis, it seems that combination of two or more techniques can generate better result either from size of ROM, complexity, accuracy and speed. Some simple modification can obviate the need for largest ROM device, reduce the delay, and thus provide for a neat integration of the two subsystems. As prove a smaller modification to the interpolator, also reducing storage and significantly reducing delay, has been proposed [1].

Conclusion:

Four techniques namely direct lookup table, interpolation, table partitioning and co-transformation have been discussed as to compute the addition and subtraction operations. In summary, it can be observed that the majority of the suggested designs collaborate with the interpolation method for approximating the addition and subtraction functions. It is known that the combination between the co-transformation and interpolation methods able to produce better speed than FLP system.

ACKNOWLEDGMENT

This work was financially supported by Ministry of Higher Education Malaysia (MOHE) under the Exploratory Research Grant Scheme (ERGS 9010-00020).

REFERENCES

[1] Ismail, R.C. and J.N. Coleman, 2011. "ROM-less LNS", *IEEE Symposium on Computer Arithmetic*, pp: 43-51.

[2] Earl E. Swartzlander and A.G. Alexopoulus, 1975. "The Sign/Logarithm Number System," *IEEE Transactions on Computers,* pp: 1238-1242.

[3] Taylor, F., 1983. "An Extended Precision Logarithmic Number System," *IEEE Transactions on Acoustics, Speech and Signal Processing,* 31: 232-234.

[4] Coleman, J.N., E.I. Chester, C.I. Softley and J. Kadlec, 2000. "Arithmetic on the European Logarithmic Microprocessor," *IEEE Transactions on Computers,* pp: 702-715.

[5] Taylor, F.J., R. Gill, J. Joseph and J. Radke, 1988. "A 20 Bit Logarithmic Number System Processor," *IEEE Transactions on Computers,* pp: 190-200.

[6] Lewis, D.M., 1990. "An Architecture for Addition and Subtraction of Long Word Length Numbers in the Logarithmic Number System," *IEEE Transactions on Computers,* pp: 1325-1336.

[7] Lewis, D.M., 1994. "Interleaved Memory Function Interpolators with Application to an Accurate LNS Arithmetic Unit," *IEEE Transactions on Computers,* 43: 974-982.

[8] http://en.wikipedia.org/wiki/Lookup_table.
[9] Arnold, M.G., J. Cowles and T. Bailey, 1988. "Improved Accuracy for Logarithmic Addition in DSP Applications," *International Conference on Acoustics, Speech, and Signal Processing,* 3: 1714-1717.
[10] Stouraitis, T. and F.J. Taylor, 1988. "Analysis of Logarithmic Number System Processor" ", *IEEE Transactions on Circuit and System,* 35: 519-527.
[11] Coleman, J.N., 1995. "Simplification of Table Structure in Logarithmic Arithmetic," *Electronic Letters,* 31: 1905-1906.
[12] Mark G. Arnold, Thomas A. Bailey, John R. Cowles and M.D. Winkel, 1998. "Arithmetic Co-Transformations in the Real and Complex Logarithmic Number Systems," *IEEE Transactions on Computers,* 47: 777-786.
[13] Coleman, J.N., C.I. Softley, J. Kadlec, R. Matousek, M. Tichy, Z. Pohl, A. Hermanek and N.F. Benschop, 2008. "The European Logarithmic Microprocessor ", *IEEE Transactions on Computers,* pp: 532-546.