INTERNATIONAL JOURNAL FOR TRENDS IN ENGINEERING & TECHNOLOGY
VOLUME 5 ISSUE 2 – MAY 2015 - ISSN: 2349 - 9303

Implementation and Impact of LNS MAC Units in Digital Filter Application

Hari Krishna Raja .V .S
Sri Shakthi Institute of Engineering and Technology,
Department of Electronics and Communication Engineering,
hari.rajaece205@gmail.com

Abstract— The logarithmic number system (LNS) is an efficient way to represent data in VLSI processors because its round-off error behavior resembles that of floating point arithmetic. LNS reduce the power dissipation in signal-processing-related application such as hearing-aid devices, video processing and error control. This paper presents techniques for low-power addition/subtraction in the LNS and quantifies their impact on digital filter VLSI implementation. The operation of addition and subtraction are difficult to perform in LNS as complex look up tables (LUTs) are needed. The impact of partitioning the look-up-tables required for LNS addition/subtraction on complexity performance and power dissipation is quantified. LNS base and LNS word are the two design parameters exploited to minimize complexity. A round-off noise model is used to demonstrate the impact of base and word-length on SNR of the output of FIR filters. In addition, techniques for low-power implementation of an LNS multiply accumulate (MAC) units are investigated. The proposed techniques can be extended to co-transformation-based circuits that employ interpolators. The results are demonstrated by evaluating the power dissipation, complexity and performance of several FIR filter configurations comprising one, two or four MAC units. Simulation of placed and routed VLSI LNS-based digital filters using Xilinx ISE reveal that significant power dissipation savings are possible by using optimized LNS circuits at no performance penalty, when compared to linear fixed-point two's-complement equivalents.

Index Terms— Computer arithmetic, digital filter, MAC, LNS, LUT

1 INTRODUCTION

Data representation is an important parameter in the design of low-power processors since it affects both the switching activity and hardware complexity [4]. The logarithmic number system (LNS) has been investigated as an efficient way to represent data in special purpose VLSI processors, since it allows for simple arithmetic circuits under certain conditions. In particular, LNS exploits the properties of the logarithm to reduce the basic arithmetic operations of multiplication, division, roots, and powers to binary addition, subtraction, and right and left shifts, respectively. In addition to simplifying several operations, LNS provides efficient data representation because its round off error behavior resembles that of floating-point arithmetic. In fact, LNS-based systems have been proposed that exhibit characteristics similar to 32-bit single-precision floating-point representation [1]. The operations of addition and subtraction are rather awkward to perform in LNS as complex look-up tables (LUTs) or other approximation circuitries are needed. While for short word lengths simple techniques based on LUTs suffice, more elaborate approximation techniques are required for longer word lengths.

1.1 Existing System

Several authors have proposed solutions to reduce complexity of awkward LNS operations. Mahalingam et al. [11] improve Mitchell's Algorithm in terms of the accuracy of the logarithmic operations, while Johansson et al. [10] use a method based on sums of bit products to implement the basic logarithmic functions. Arnold et al. [2] suggest the use of co-transformations for the reduction of the LUT. Very recently, Ismail et al. [9] presented a co transformation procedure and an improved interpolation method that reduce the size of LUT to an extent that allows their easy synthesis in logic. Arnold et al. [3] propose complex LNS as a generalization of LNS, which represents complex values in log-polar form.

For several practical applications, the benefits of LNS are found to be more important than its inherent disadvantages. In particular, several authors have shown that LNS reduces power dissipation in signal-processing-related applications, ranging from hearing-aid devices and sub band coding, to video processing and error control. Moreover, logarithmic techniques have been employed in turbo code decoding for wireless communication applications. In particular, logarithmic representation has been proved to be suited for the implementation of the symbol-by-symbol logarithmic maximum a posteriori algorithm used for iterative decoding. Peng et al. have adopted LNS for the implementation of an FFT based log-sum-product-decoding-algorithm used in decoding of non binary low-density parity check codes. In particular, the impact of the selection of the base b of the logarithm has been investigated as a means to explore tradeoffs between precision and dynamic range given a particular word length. Paliouras et al. address the low-power LNS properties from a representational viewpoint and do not focus on power dissipation estimation data obtained by circuit simulations.

1.2 Proposed System

The proposed study focuses on the use of partitioning as a technique to limit the exponential growth of the size of LUTs with the word length. The technique is simple and leads to fast circuits. Initially, extending, optimal selection of LNS design parameters is sought, including word length and base assuming a simple

partitioned LUT architecture. Subsequently, in the second stage of the proposed framework the design techniques and the derived architectures are presented, targeting LNS MAC units in 90-nm technology. To illustrate the use of the proposed framework, the area-time-power design space of a low-pass finite impulse response (FIR) filter is explored for several configurations of MAC units. A similar study has recently been performed by Galal and Horowitz in the different context of floating-point arithmetic [7]. Departing from direct LUT organization, a variety of LNS architectures for addition, subtraction, and multi-operand operations, has been proposed in the literature employing interpolation, linear or polynomial approximation aiming at reducing the memory requirements, particularly for larger word lengths, such as 32 bits [8]. These ideas have been combined with mathematical decompositions and transformations of the basic operations, exploiting the particular characteristics of the functions to further simplify approximation [9]. Beyond the representational properties of LNS that have an impact on switching activity, LNS arithmetic units have structural characteristics that can be exploited to reduce power dissipated. In particular, they comprise

- mutually exclusive subunits, which can be used selectively, and
- Imbalanced delay paths.

Therefore, simple low-power design techniques are found to suit an LNS adder/subtractor organization very well; the impact is quantified for the case of lookup-based architectures, but other LNS architectures may benefit as well, in terms of reducing power dissipation. Extension to other architectures is demonstrated using an interpolation-based subtractor as an illustrative example. Partitioned LUT circuits provide high speed; more sophisticated techniques can be used to reduce size. In summary, the contributions of this paper are as follows:

- a low-power design framework for LNS systems,
- the quantification of power dissipation reduction and performance improvement made possible by using LNS ,compared to equivalent binary implementations,
- the design space exploration using the number of LUTs for addition /subtraction as a parameter ,for the case of using combinational logic for LUT implementation, and
- the extensions of SNR models in LNS for the case of b≠2.

2 LNS BASICS

The basic idea in LNS is to use logarithms to represent data. Since the logarithm of a negative number is not real, to represent signed numbers in LNS, the sign information is stored as a separate bit s_x, and used in combination with the logarithm of the magnitude of the number. Furthermore, since the logarithm of zero is not a finite number, an additional single-bit flag z_x is used to denote that a number is zero. Summarizing, X denotes the original number, x denotes the logarithm of the absolute value of |X|, and X_{LNS} is a triplet containing the sign bit, the zero bit and x. Formally in LNS, a number X is represented as the triplet

$$\chi_{LNS} = (z_x, s_x, x), \quad (1)$$

Where z_x is asserted in the case that X is zero, s_x is the sign of X and $x=\log_b(|X|)$, if X is not zero, with b being the base of the logarithm, also called base of the representation. The choice of b plays a crucial role in the representational capabilities of the triplet in (1), as well as the computational complexity of the processing and forward and inverse conversion circuitry. Due to the basic properties of the logarithm, the multiplication of X_{LNS} and Y_{LNS} is reduced to the computation of the triplet Z_{LNS}

$$Z_{LNS} = (z_x, s_x, z), \quad (2)$$

Where $z_Z = z_x \vee z_y$, $s_Z = s_X$ xor s_y, and z = x + y. Similarly, the case of division reduces to binary subtraction. The derivation of the logarithm a of the sum A of two triplets is more involved, as it relies on the computation of

$$a = \max\{x,y\} + \log_b(1+b^{-|x-y|}), \quad (3)$$

$$= \max\{x,y\} + \phi_a(d), \quad (4)$$

Where $\phi_a(d) = \log_b(1+b^{-d})$ and

$$d = |x-y|. \quad (5)$$

Similarly, the derivation of the difference of two numbers, requires the computation of

$$c = \max\{x,y\} + \log_b(1-b^{-|x-y|}), \quad (6)$$

$$= \max\{x,y\} + \phi_s(d), \quad (7)$$

Assume that a TC word is used to represent the logarithm x, composed of a k-bit integral part and an l-bit fractional part. The range D_{LNS} spanned by x is given by

$$D_{\mathrm{LNS}} = \left[-b^{2^{k-1}-2^{-l}}, -b^{2^{-l}}\right] \bigcup \{0\} \bigcup \left[b^{2^{-l}}, b^{2^{k-1}-2^{-l}}\right]. \quad (8)$$

a linear TC representation of i integral bits and f fractional bits. In general, LNS offers a superior range, over the linear TC representation. This is achieved using comparable word lengths, by departing from the strategy of equispaced representable values and thus resorting to a scheme that resembles floating-point arithmetic. The basic organization of an LNS adder/subtractor is shown in Fig. 1. The parallel subtraction are implemented, followed by a multiplexer, which computes *d* according to the rule

$$s_1 = x - y, \quad (9)$$

$$s_2 = y - x, \quad (10)$$

$$d = |x-y| = \begin{cases} s_1, & s_1 > 0, \\ s_2, & \text{otherwise.} \end{cases} \quad (11)$$

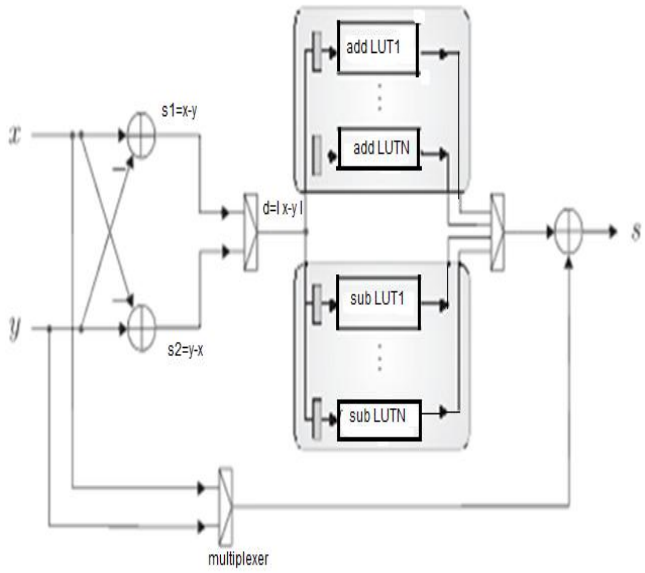

Fig. 1. The organization of an LNS adder/subtractor

The choice exploits the sign of either (9) or (10), as a select signal for the multiplexer. The same signal is used to select the maximum of x and y, required for the computation of (4) and (7). The complexity of LNS circuitry arises from the fact that the values of functions ϕ_a and ϕ_s should be computed by the LNS addition/subtraction circuit hardware for all required values of d. There are two main approaches to implement the evaluation of functions, namely the hardware implementation of an approximation algorithm or the offline pre-computation and storage of all required values in an LUT. The former approach is generally adopted for high-precision applications, while the latter approach is generally preferable for smaller word lengths, i.e., in relatively low-precision applications where the size of the required LUTs is moderate. Both approaches have been extensively studied in the context of elementary function approximation. Let x denote the base-b logarithm of X and x_2 denote the base-2 logarithm of X. Since $x = \log_b x = x_2(\log_b 2)$, the conversion between a base-b LNS and a base-2 LNS requires scaling by a constant factor. Several authors have studied hardware implementations of converters to/from base-2 LNS. In this paper, although conversion is neglected the conclusions about power consumption are valid for the complete application. To better clarify this we assume an FIR filter of order N, requiring about NMAC operations for each input conversion and each output conversion. If E_{in} is the average energy for one input conversion, E_{out} is the average energy for one output conversion, and E_y is average energy for one MAC, the total energy (after initialization) for the FIR filter to produce each result is $E_{FIR}=E_{in}+E_{out}+N.E_y$. For sufficiently large values of N, the percentage of energy consumed in the multiply-add units may approach 100 percent of the total as $\lim_{N\to\infty} E_y/E_{FIR} = 1.0$.

3 LOW-POWER DESIGN OF LNS CIRCUITS

In this section, low-power LNS architectures for addition and subtraction are presented. The memory structure is organized as a collection of LUTs and is the most complex part of the LNS adder/subtractor. Several designs were investigated, distinguished by two choices, i.e., first, the choice of using either latches or D flip-flops (DFFs) to freeze the addresses of inactive sub-LUTs, and, second, the choice to select the active sub-LUT either based on the most significant bits (MSB) or on the least significant bits (LSB) of d in (11). In the proposed design framework, power dissipation reduction is sought by partitioning the particular LUTs into smaller LUTs, called sub-LUTs, only one of which is active per operation. This organization is shown in Fig. 1. To guarantee that no dynamic power is dissipated in the inactive sub-LUTs, the corresponding sub-LUT addresses are latched and remain constant throughout a particular operation. Complexity reduction in LNS processors by partitioning of the LUTs has been successfully applied. Here, we focus on combinational logic implementation of LUTs, instead of memory-based implementation. The organization of the LNS adder/subtractor comprises N sub-LUTs per operation, as shown in Fig. 1. The upper sub-LUT system corresponds to function $\phi_a(d)$ required for LNS addition, i.e., addition of operands having the same sign, while the lower sub-LUT system is used for LNS subtraction, i.e., addition of operands of different signs.

3.1 Organization And Complexity Of LUT Subsystem In LNS Adder/Subtractor

Assume that b denotes the logarithmic base and l is the number of the fractional bits employed in the representation of the logarithms. Therefore, differences among the values stored in LUT2 are limited to their less significant part; therefore, the less significant part is the only one that needs to be stored for each value. Hence, fewer bits per entry are required to be stored in LUT2 than in LUT1. Sub-LUTs that correspond to the upper parts of the interval need to store data words of reduced length, since stored values share a common most significant part. The possibility to determine the active sub-LUT using the LSBs of d is of interest, as LSBs are available early in the computation of d; thus, allowing the fast generation of selection signals. However, a partitioning scheme based on LSBs does not facilitate memory compression since consecutive function samples are stored in different sub-LUTs.

3.2 Implementation Of LNS Adder/Subtractor

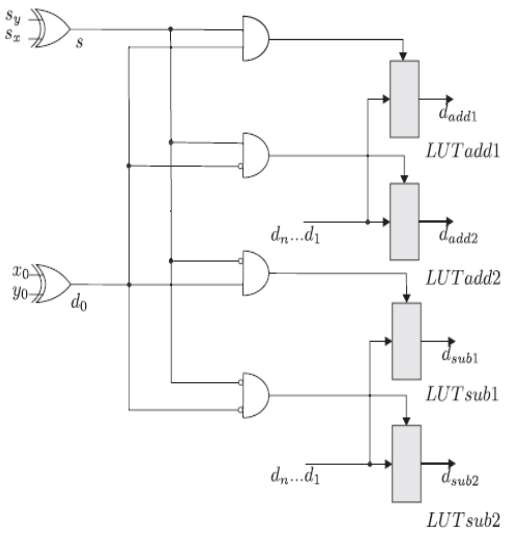

Fig. 2. Four-latch organization using LSB

Fig.3. depicts a DFF-based architecture. It is noted that a latch-based gated clock is used for the DFFs, since additional signals are used to enable the corresponding flip-flops. Significant advantage is obtained since gated clocks achieve further power savings and also the problem of setup and hold time violations is easier to resolve, since glitches are avoided.

Since the utilization of the MSB for LUT selection is not efficient for a latch-based design due to additional hardware used to introduce the required delay to fast paths of the circuit, a solution based on DFFs is preferable.

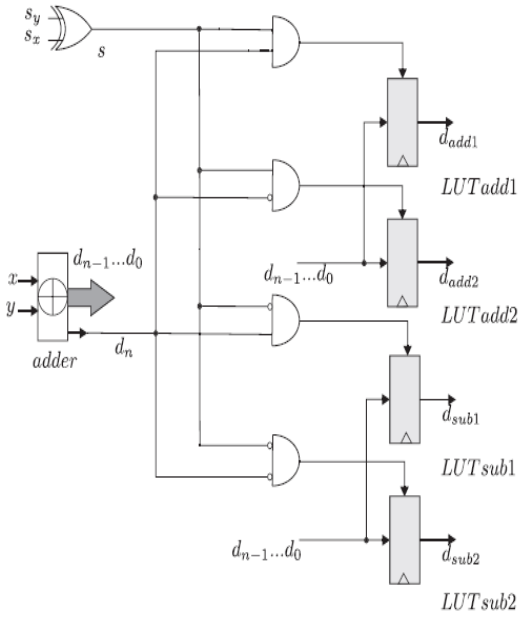

Fig. 3. DFF organizations using the MSB

4 PROPOSED LNS MAC ARCHITECTURES

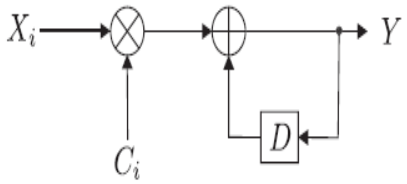

Fig. 4. Organization of single-MAC architecture.

The basic structure of the single-MAC unit is shown in Fig.4. Symbols * and + denote a multiplier and an adder, respectively, while D denotes a delay unit, implemented as a register. The LNS equivalent to single-MAC architecture is depicted in Fig.5, where the binary multiplier has been replaced by an adder, and the binary adder is mapped to an LNS adder/subtractor. The LNS adder/subtractor is augmented with saturation circuitry and exploits a zero flag to avoid unnecessary activation of LUT partitions and further reduce power dissipation.

In the implementation of Fig.5, it is evident that the paths to the inputs of the final adder are not balanced; thus, leading to excessive switching activity at the adder following the memory structure.

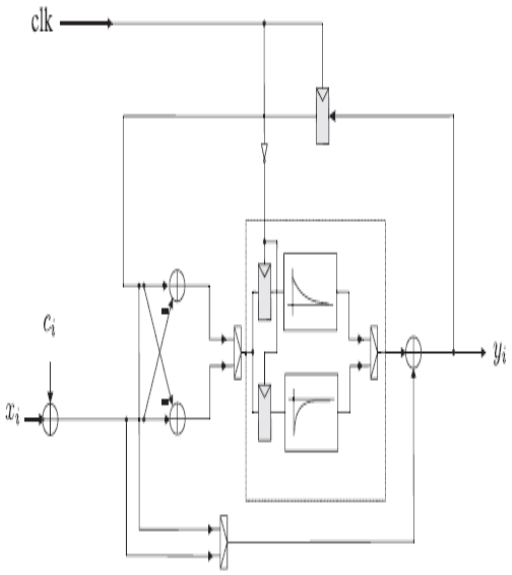

Fig. 5. LNS MAC unit.

5 RESULT ANALYSIS

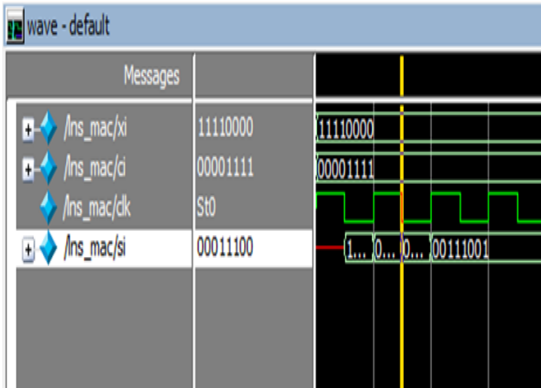

Fig. 6. Simulation of LNS MAC

The result analysis is made to compare the area, power, and timing constraints between single MAC architecture and LNS MAC architecture. From the synthesized results, LNS MAC architecture produces better performance. The estimated result of LNS MAC Units using Xilinx ISE is shown below in Fig.6.

6 CONCLUSION

The adoption of LNS can lead to very efficient circuits for digital filtering applications when appropriately selecting the logarithmic base and the word length in a contemporary 90-nm technology outperforming circuits based on TC arithmetic. An LNS-based system using the proposed adder/subtractor offers substantial power dissipation savings at no performance penalty. Partitioning of the LUTs is employed to create parts in the circuit that can be independently activated, thus, reducing power dissipation. Power has been reduced by latching the inputs to the LUTs. Furthermore, the gated clock technique has been used to further reduce power consumption performed to the latched inputs due to the clock signal. It has been shown that the choice of number of sub-LUTs is an important design parameter that can be

employed for exploration of the area, time, and power design space. Furthermore, the application of retiming is particularly useful in avoiding unnecessary switching activity, due to unbalanced delay paths in LNS arithmetic circuits. By properly defining wordlength, base, circuit architecture and LUT organization it has been shown that the LNS-based MACs can outperform the corresponding TC ones in both power and delay complexities, for specific practical word lengths.

REFERENCES

[1]. Arnold .M.G, Bailey .T.A, Cowles .J.R, and Winkel .M.D, (1992) "Applying Features of the IEEE 754 to Sign/Logarithm Arithmetic,"
IEEE Trans. Computers, vol. 41, pp. 1040-1050.

[2]. Arnold .M.G, Bailey .A.T, Cowles .J.R, and Winkel .M.D, (1998) "Arithmetic Co-Transformations in the Real and Complex Logarithmic Number Systems," IEEE Trans. Computers, vol. 47, no. 7, pp. 777-786.

[3]. Arnold .M and Collange .S, (2011) "A Real/Complex Logarithmic Number System ALU," IEEE Trans. Computers, vol. 60, no. 2, pp. 202-213.

[4]. Chen .K.-H and Chiueh T.-D, (2006) "A Low-Power Digit-Based Reconfigurable FIR Filter," IEEE Trans. Circuits and Systems II: Express Briefs, vol. 53, no. 8, pp. 617-621.

[5]. Coleman .J, Softley .C, Kadlec ,Matousek .J , Tichy .R .M, Pohl .Z, Hermanek .A, and Benschop .N, (2008) "The European Logarithmic Microprocesor," IEEE Trans. Computers, vol. 57, no. 4, pp. 532-546.

[6]. Collange .S, Detrey .J, and Dinechin F.de, (2006) "Floating-Point or LNS: Choosing the Right Arithmetic on an Application Basis," Proc. Ninth Euromicro Conf. Digital System Design (DSD '06), pp. 197-203.

[7]. Galal .S and Horowitz .M, (2011) "Energy-Efficient Floating-Point Unit Design," IEEE Trans. Computers, vol. 60, no. 7, pp. 913-922.

[8]. Henkel .H, (1989) "Improved Addition for the Logarithmic Number System," IEEE Trans. Acoustics, Speech, and Signal Processing, vol. 37, no. 2, pp. 301-303.

[9]. Ismail R.C and Coleman J.N, (2011) "ROM-less LNS," Proc. IEEE Symp. Computer Arithmetic, pp. 43-51.

[10]. Johansson .K .V, Gustafsson .O .S, and Wanhammar .L, (2008) "Implementation of Elementary Functions for Logarithmic Number Systems," IET Computers and Digital Techniques, 2008.

[11]. Mahalingam .V and Ranganathan .N, (2006) "Improving Accuracy in Mitchell's Logarithmic Multiplication using Operand Decomposition," IEEE Trans. Computers, vol. 55, no. 12, pp. 1523-1535.