

Efficient Conversion From Binary to Multi-Digit Multi-Dimensional Logarithmic Number Systems using Arrays of Range Addressable Look-Up Tables

R. Muscedere¹, V.S. Dimitrov², G.A. Jullien², and W.C. Miller¹

¹Research Centre for Integrated Microsystems,
University of Windsor, Ontario, Canada, N9B 3P4; E-Mail: musced3@uwindsor.ca

²ATIPS Laboratory, University of Calgary, Alberta, Canada, T2N 1N4

Abstract

The Multi-Dimensional Logarithmic Number System (MDLNS), with similar properties to the Logarithmic Number System (LNS), provides more degrees of freedom than the LNS by virtue of having two orthogonal bases and the ability to use multiple digits. Unlike the LNS, there is no direct functional relationship between binary/floating point representation and the MDLNS representation. Traditionally look-up tables (LUTs) were used to move from the binary domain to the MDLNS domain. This method can be unrealistic for hardware implementation when large binary ranges or multiple digits are used. This paper introduces a range addressable technique for table look-up arrays that allows efficient conversion from binary to single or multi-digit MDLNS.

Keywords: Multi-Dimensional Logarithmic Number System; Binary to MDLNS Conversion; Range Addressable Look-Up Tables.

1. Introduction

The Multi-Dimensional Logarithmic Number System (MDLNS) [1], which uses 2 or more orthogonal bases (of which the first is 2), has similar properties to the logarithmic number system (LNS) [2][3]. The MDLNS provides more degrees of freedom than the LNS by virtue of the orthogonal bases and the ability to gain from the use of multiple digits. However, these extra degrees of freedom introduce new complexities in the binary conversion process. The MDLNS is a generalization of the index calculus introduced into the double-base number system [4].

A multidimensional n -digit logarithmic (MDLNS) representation of the real number, x , has the form:

$$x = \sum_{i=1}^n s_i \prod_{j=1}^b p_j^{e_j^{(i)}} \quad (1)$$

where $s_i \in \{-1, 0, 1\}$ and $e_j^{(i)}$ are integers. b is the number of bases used (at least two, where $p_1 = 2$). The case, $s_i = 0$ is required when either the number of digits required is less than n , or the special case $s_i = 0, \forall i$ when $x = 0$.

In this paper we will restrict ourselves to the 2-D MDLNS, with a simpler notation

$$\sum_{i=1}^n s_i 2^{a_i} D^{b_i}, \text{ an } n\text{-digit two-dimensional logarithmic representation, where } n = 2 \text{ will}$$

be a special case. D is a suitably chosen real number (not necessarily an integer but not a multiple of 2).

The MDLNS is defined with integer exponents, which simplifies the hardware implementation of the index calculus operations. An obvious hardware implementation is to use a look-up table (LUT) to store all possible exponent combinations; however, the table grows exponentially with the size of the input data, and can be unrealistically large for many practical applications.

A more efficient method, for a single-digit case, is to use a pattern-matching scheme to search for the best normalized match between the target binary representation and the second MDLNS term (D^{b_i}). Once found, the first MDLNS term (2^{a_i}) can then be selected to compensate for the normalization. In this case a LUT is still required; however, its size is only determined by the second exponent. An efficient pattern-matching procedure, using a modified address decoder in a look-up table, can be used to search for these patterns to generate a fast single or multi-digit MDLNS representation of any binary value (integer or floating point).

2. Single-Digit MDLNS Conversion

Any real number can be represented in the single-digit MDLNS by

$$x = s \cdot 2^a \cdot D^b \quad (2)$$

where $b \in \{-2^{R-1}, -2^{R-1} + 1, \dots, 0, \dots, 2^{R-1} - 2, 2^{R-1} - 1\}$, and $a \in \{\dots, -1, 0, 1, \dots\}$. The capability of this representation to produce a particular real number is based on the choice for D and R .

2.1. Generating the LUT contents

The normalized values and exponents used in the MDLNS approximation can be calculated for all the b 's, given R and normalizing the kernel from Eqn. (2).

$$2 > 2^a \cdot D^b \geq 1 \quad (3)$$

(3) can be rewritten in a logarithmic form:

$$\ln(2) > a \cdot \ln(2) + b \cdot \ln(D) \geq \ln(1)$$

$$1 - \frac{b \cdot \ln(D)}{\ln(2)} > a \geq \frac{-b \cdot \ln(D)}{\ln(2)}$$

and the ceiling function is used to return the integer that matches the condition,

$$a = \text{ceiling}\left(\frac{-b \cdot \ln(D)}{\ln(2)}\right). \text{ An example of this shown in Table 1.}$$

Table 1: Solution for a with $R = 4, D = 3$

x	a	b
1.248590	13	-8
1.872885	12	-7
1.404664	10	-6
1.053498	8	-5
1.580247	7	-4
1.185185	5	-3
1.777778	4	-2
1.333333	2	-1
1.000000	0	0
1.500000	-1	1
1.125000	-3	2
1.687500	-4	3
1.265625	-6	4
1.898437	-7	5
1.423828	-9	6
1.067871	-11	7

These values are then sorted by their real representation (x) as in Table 2.

Table 2: Sorted solution for a with $R = 4, D = 3$

x	a	b
1.000000	0	0
1.053498	8	-5
1.067871	-11	7
1.125000	-3	2
1.185185	5	-3
1.248590	13	-8
1.265625	-6	4
1.333333	2	-1
1.404664	10	-6
1.423828	-9	6
1.500000	-1	1

Table 2: Sorted solution for a with $R = 4, D = 3$

x	a	b
1.580247	7	-4
1.687500	-4	3
1.777778	4	-2
1.872885	12	-7
1.898437	-7	5

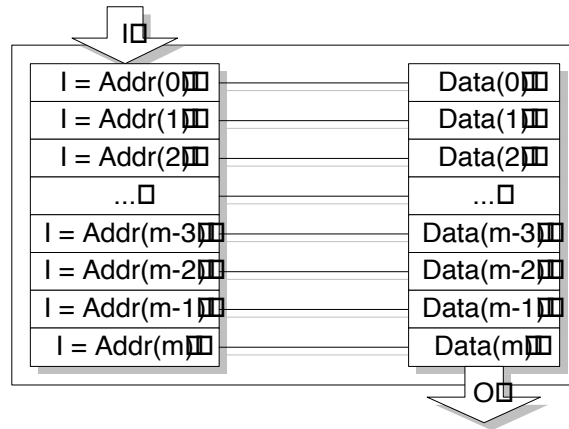
2.2. Conversion Example

We wish to convert 845937 to a single-digit MDLNS representation for $D = 3$ and $R = 4$. The normalized representation of 845937 is $1.613497 \cdot 2^{19}$, and looking up the closest match for 1.613497 in Table 2 gives 1.580247 ($a = 7, b = -4$). Therefore, the most accurate single-digit MDLNS representation for 845937 is $2^{7+19} \cdot 3^{-4} = 828505$. Larger values of R will, of course, yield more accurate approximations.

2.3. The RALUT structure

Searching for the closest match in Table 2 (or any other generated table) can be performed by using a modified address decoder in a standard ROM or LUT. Conceptually, standard LUTs (see Figure 1) use address decoders to match an input (I) to a series of unique values (m elements of $Addr$). Only one of these values will match the input and activate a word enable line which drives the data patterns ($Data$) to the output (O) of the LUT.

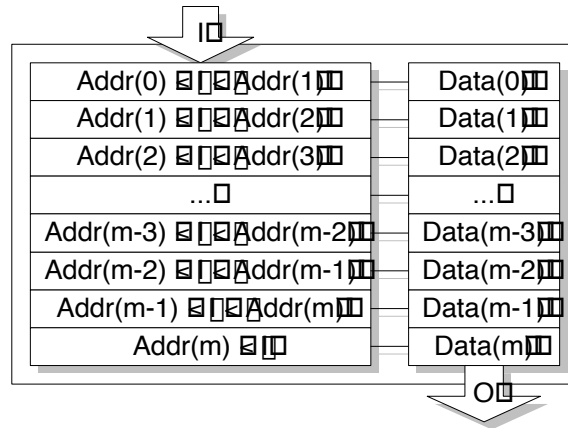
Figure 1: Standard LUT design



However for this application, matching on a particular range of values is necessary. Here we introduce a “range addressable” LUT (RALUT) to perform this function. A RALUT differs from the classic LUT by changing the address decoder system to match on a range of values rather than exact values (see Figure 2). Logically, the input address (I)

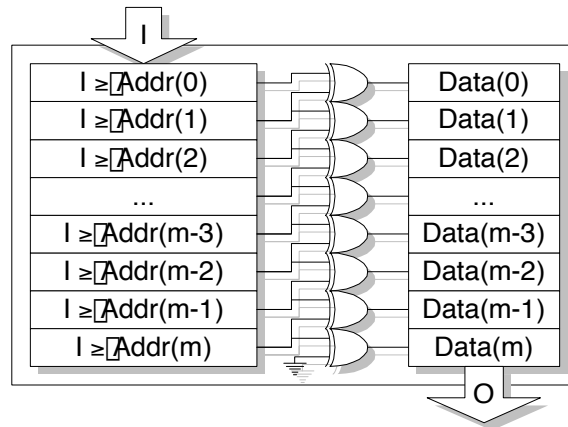
would be compared within the range of two neighboring addresses (i.e. $Addr(1)$ and $Addr(2)$). Again, only one of these comparisons will match the input and activate a word enable line which drives the data patterns ($Data$) to the output (O) of the RALUT.

Figure 2: RALUT structure



We can optimize the hardware by removing the “lesser-than” component of each address element. The outputs of the remaining two neighboring “greater-than-equal” can be XORed to find the transition point from all the “greater-than-equal” components (see Figure 3). Only one of the outputs of the XORs would be active, therefore producing the appropriate word enable line which drives the data patterns ($Data$) to the output (O) of the RALUT.

Figure 3: Optimized RALUT structure



In our example above, we need to find the closest value in Table 2 to our normalized value. By placing the elements of Table 2 into the RALUT, we will only find a value which is less-than-or-equal to the input, not necessarily the closest. To solve this issue, the output of the RALUT should also include the next MDLNS value in the sequence (for error calculations), and the associated exponent information (for representation generation). The RALUT will return the two closest approximations (one lower-than-or-equal and one higher-than) to the normalized value, but the conversion process must test both to find the

closest.

Table 3: RALUT contents for $R = 4, D = 3$

Input	Outputs					
Address	x_i	a_i	b_i	x_{i+1}	a_{i+1}	b_{i+1}
1.000000	1.000000	0	0	1.053498	8	-5
1.053498	1.053498	8	-5	1.067871	-11	7
1.067871	1.067871	-11	7	1.125000	-3	2
1.125000	1.125000	-3	2	1.185185	5	-3
1.185185	1.185185	5	-3	1.248590	13	-8
1.248590	1.248590	13	-8	1.265625	-6	4
1.265625	1.265625	-6	4	1.333333	2	-1
1.333333	1.333333	2	-1	1.404664	10	-6
1.404664	1.404664	10	-6	1.423828	-9	6
1.423828	1.423828	-9	6	1.500000	-1	1
1.500000	1.500000	-1	1	1.580247	7	-4
1.580247	1.580247	7	-4	1.687500	-4	3
1.687500	1.687500	-4	3	1.777778	4	-2
1.777778	1.777778	4	-2	1.872885	12	-7
1.872885	1.872885	12	-7	1.898437	-7	5
1.898437	1.898437	-7	5	2.000000	1	0

3. Binary to Multi-Digit Conversion

Multi-digit approximation provides many different possibilities to represent a real number in MDLNS. To limit these possibilities for real-time hardware implementation, the kernel sequence DBNS (sequence of numbers with a specific finite precision) is forced to be monotonic. This ensures that the next digit in the sequence always is smaller in magnitude therefore limiting the complexity of the problem. The process of finding each digit is a simple greedy algorithm; a single-digit approximation is made, and subsequent single-digit approximations are made on the reducing error sequence, until the number of digits is satisfied.

3.1. Two-Digit Conversion Example

To find a two-digit representation for the previous example, we simply continue the conversion process on the error. The target was 845937 with a computed approximation of 828505. This is an error of 17432 or $1.063965 \cdot 2^{14}$, yielding an approximation of $1.067871 \cdot 2^{14}$ or $2^{-11+14} \cdot 3^7 = 17496$ from Table 2. The final representation is $2^{7+19} \cdot 3^{-4} + 2^{-11+14} \cdot 3^7 = 846001$, an error of 64 from the target.

In this example, $s_2 = 1$, however, $s_2 = -1$ may also be true. For this representation to be valid, the first digit must be the next highest in the MDLNS sequence (see Table 2, 1.687500 is the next entry from 1.580247), $1.687500 \cdot 2^{19}$ or

$2^{-4+19} \cdot 3^3 = 884736$. The error from the target is -38799 or $-1.184052 \cdot 2^{15}$, and its approximation is $-1.185185 \cdot 2^{15}$ or $-1 \cdot 2^{5+15} \cdot 3^{-3} = -38836$ from Table 2. The final representation for this case is $2^{-4+19} \cdot 3^3 - 2^{5+15} \cdot 3^{-3} = 845900$, an error of 37 from the target. The latter is a better representation than the first. Therefore, for two or more digit approximations, all possibilities of s_i should be explored.

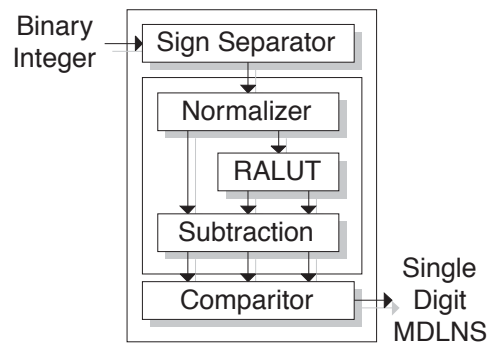
For very large R 's, this method of using an RALUT for searching for appropriate patterns can be very costly in terms of hardware (i.e. doubling as R increases). To solve this problem, a single MDLNS sequence RALUT can be separated into two or more RALUTs to reduce the overall table entries (the method for doing this is quite detailed and will not be discussed here, but the results will be shown). This separation process does require extra logic and computation time in-between each table, however, its implementation size may be negligible compared to the extra entries in the RALUT. For example, if $R = 16$ and $D = 3$ a resulting 65536 entry RALUT can be divided into a 332 and 250 entry RALUT (0.9% original size), or a 26, 29 and 150 entry RALUT (0.3% original size). Both of these options represent considerable hardware savings.

4. Circuit Implementation

Binary to single-digit MDLNS conversion can be implemented in a feed-forward circuit, and therefore able to be pipelined for high-speed applications.

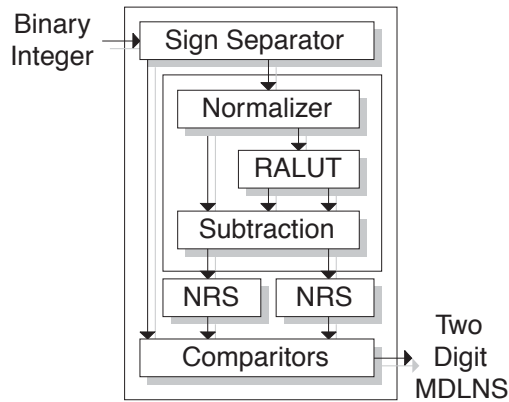
When converting to the MDLNS, the sign and magnitude of the target integer must be separated. An input of zero is a special case, since it cannot be represented with any exponent combination in the MDLNS (or LNS). Otherwise, the magnitude is normalized, matched, and compared with those matches to find the minimal error to the original target. Either the higher or lower approximation will be more accurate (see Figure 4).

Figure 4: Feed-forward Single-Digit Block Diagram



For multi-digit conversions, the Normalization, RALUT and Subtractions (NRS) blocks are connected in a binary tree structure in order to calculate all possible approximations. Each NRS block passes the total error and previous exponents of the approximation chain to the next block until the final sequence of comparators determine the most accurate result (see Figure 5 for a two-digit block diagram).

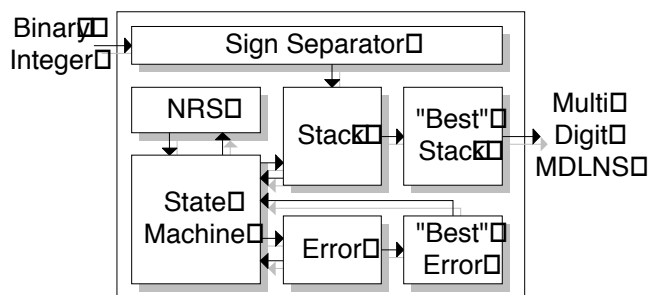
Figure 5: Feed-forward Two-Digit Block Diagram



In some applications, the need for high-speed conversion may not be necessary (an example is a digital hearing-aid processor). For a multi-digit MDLNS in the feed-forward circuit, $2^n - 1$ NRS blocks would be duplicated in the design. To improve the usage of resources, a feed-back circuit can be implemented which uses a single NRS block.

The circuit uses a state-machine to control the input and output flow of data from a stack (register file, 2^n entries) to the NRS block. The current and previous results from the NRS block are stored and retrieved from the stack as the state-machine traverses each possibility of the MDLNS representation “tree”. When each of the leaves of the tree has been reached, the approximation error is compared with the “best” approximation error, and, if smaller, the whole stack is duplicated into a mirrored stack where it becomes the “best” approximation. Once all the branches of the tree have been processed, the mirrored stack is processed and outputs the sign and exponent values for each digit for the most accurate MDLNS approximation (see Figure 6). This circuit is scalable and optimal for $n > 2$ since the number of possible approximations is 2^{n-1} . For $n = 2$, a simplified circuit containing fewer resources can be used (since the stack depth in the scalable design never exceeds 1), which generates the same approximation as the scalable circuit.

Figure 6: Feed-back Multi-Digit Block Diagram



5. Design Details

The feed-forward and two feed-back circuits have been fully described in parameterized

Verilog code where R , D and n are adjustable. The designs have been verified by simulation at both the behavioral and structural (gate-synthesis) levels.

The synthesized scalable feed-back converter design ($n = 2$, $R = 4$ and $D = 3$) requires 24 cycles per conversion and consumes a $125,000\mu\text{m}^2$ area (in a TMS320C40 0.18 μm process). The simplified feed-back converter design requires 19 cycles per conversion and consumes a $85,000\mu\text{m}^2$ area. The critical path in both circuits is 15ns. This path is partially in the RALUT circuit, which is expected to decrease significantly if implemented using modern dynamic address decode techniques [5].

The simplified feed-back circuit is currently being installed in a hearing-aid filter bank test chip where it converts 16-bit binary inputs into two-digit MDLNS.

6. Conclusions

This paper has briefly discussed techniques for converting binary input data into a multi-dimensional logarithmic form. For simplicity, the paper has concentrated on two-dimensional logarithms with multiple digits.

We have discussed basic techniques for performing the conversion process, and introduced a Range Addressable Look-Up Table (RALUT) that provides single approximations for a range of input data. Examples of a single and two-digit conversion procedure have been provided where a greedy algorithm is used for multi-digit conversion. A complete conversion processor has also been described, and is currently being tested for a low-power hearing-aid application.

The MDLNS representation is a recent innovation, and much more work has to be performed in order to establish its utility in main-stream DSP applications. Initial results on our target hearing-aid processor show that it improves upon reported results for binary implementations. We have previously demonstrated that the MDLNS provides improvements over the LNS for the multi-digit case [1]. The proof provided in [1] shows that a multi-digit LNS having the same base for each digit offers no improvement in representational power, whereas an MDLNS using different bases does offer an improvement in representational power. We point out here that this reference, however, did not prove that the overall cost of all conventional LNS applications are higher than the overall cost of MDLNS applications, especially in applications where the results of addition are to be in MDLNS-this is an area that is yet to be studied in detail. However, the availability of the conversion processor described in this paper has now provided a vital component for practical applications of this new logarithmic number system and has advanced our ability to provide future in-depth studies.

7. Acknowledgment

The authors would like to acknowledge financial support from the Natural Sciences and Engineering Council (NSERC) of Canada, the Micronet Network of Centres of Excellence and Gennum Corporation. The authors also acknowledge the important contribution of the Canadian Microelectronics Corporation (CMC) for their equipment and software loan and fabrication services.

8. References

- [1] V. S. Dimitrov, J. Eskritt, L. Imbert, G. A. Jullien and W.C. Miller, 2001, "The use of the multi-dimensional logarithmic number system in DSP applications", Proceedings of the 15th IEEE Symposium on Computer Arithmetic, June, pp. 247-254.
- [2] E. E. Swartzlander and A. G. Alexopoulos, "The Sign/Logarithm Number System", *IEEE Trans. Computers*, Vol. 42, pp 1238-1242, 1975.
- [3] M. G. Arnold, T. A. Bailey, J. R. Cowles and J. J. Cupal, "Redundant Logarithmic Arithmetic", *IEEE Trans Computers*, vol. 39, No 8, pp 1077-1086, 1990.
- [4] V. S. Dimitrov, G. A. Jullien and W. C. Miller, "Theory and applications of the double-base number system", *IEEE Trans. on Computers*, vol. 48, No. 10, pp. 1098-1106, Oct. 1999.
- [5] B. Keeth, R. J. Baker, "DRAM Circuit Design", IEEE Press, 2001.