

Non-linear signal processing using index calculus DBNS arithmetic

R. Muscedere, G. A. Jullien, V. S. Dimitrov and W. C. Miller

VLSI Research Group, University of Windsor
Windsor, Ontario, Canada N9B 3P4 (musced3@uwindsor.ca)

ABSTRACT

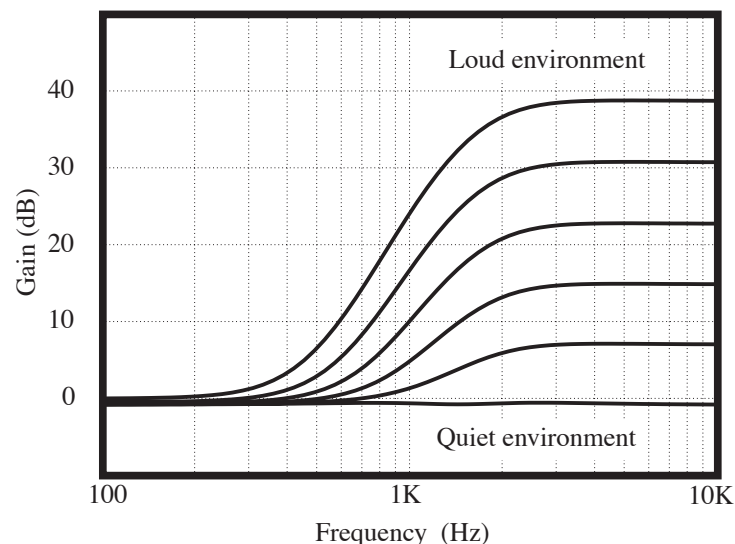
This paper discusses the use of a recently introduced index calculus Double-Base Number System (IDBNS) for representing and processing numbers for non-linear digital signal processing; the target application is a digital hearing aid processor. The IDBNS representation uses 2 orthogonal bases (2 and 3) to represent real numbers with arbitrary precision. By restricting the number of digits to one or two, it is possible to efficiently represent the real number using the indices of the bases rather than the distribution of the digits. In this paper we discuss the use of the two-digit form of this representation (2-IDBNS) to efficiently perform arithmetic associated with the non-linear processing required to correct the usual forms of hearing loss in a digital hearing aid. The non-linear processing takes the form of dynamic range compression as a function of frequency band. Currently developed digital hearing instrument processors require large dynamic range representations (20-24 bits) in order to accurately generate the dynamic range compression associated with typical hearing loss. We show that the natural non-linear representation afforded by the IDBNS provides both a more efficient signal representation and a more efficient technique for processing the dynamic range compression. We pay particular attention to a novel technique of converting from a linear binary input directly to the 2-IDBNS representation using an observation of partial cyclic repetition in the indices along with near unity approximants.

Keywords: Computer arithmetic; Non-linear number representation; double-base number system; Hearing loss digital processing.

1. INTRODUCTION

In a typical digital hearing instrument, the hearing loss compensation is performed by separating the input signal into programmable frequency bands which are then compressed to allow the amplification of low level signals while maintaining the amplitude of high level signals. We thus require a processor that is able to both perform linear processing (band separation), and non-linear processing for signal compression and expansion (see Figure 1).

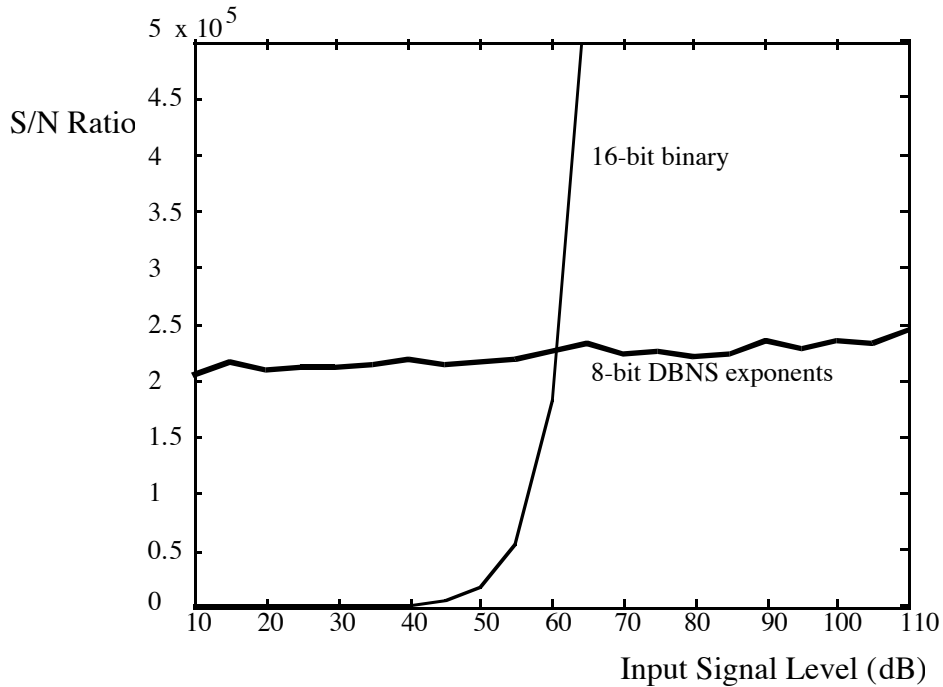
Figure 1. Dynamic signal compression in a hearing instrument



In order to be able to adequately represent the very low level signals, that are subject to the maximum amplification in the processor, very large word-lengths are required, and floating point representation is quite usual in this regard 4. For an auditory canal digital processor, the power consumption is a major component of the cost function, and so we search for any techniques that can be used to reduce the power dissipation. In this paper we propose to use a recently introduced non-linear number representation in order to both reduce the size of the number representation and to allow a relatively low cost implementation of the operations of multiplication, division, and exponentiation. Our number representation uses one and two digit double-base numbers (DBNS) where the digits are represented by the indices of their bases (in our case the bases are 2 and 3). This provides a logarithmic type of representation with the added advantage of allowing orthogonal implementations of the index computation on each base.

In the DBNS domain (as with logarithmic arithmetic 5) we can more accurately represent smaller values compared to an integer binary representation, and the percentage error normalised to the signal magnitude remains quite constant. Figure 2 shows the signal-to-quantization noise ratio vs. input level of an input signal quantized to 8-bit DBNS and an equivalent 16-bit binary representation.

Figure 2. Dynamic range compression of a 64 sample sine wave using binary and DBNS



The DBNS representation provides a constant S/N ratio throughout the range of inputs, while the binary S/N ratio drops with input level. DBNS has a superior S/N ratio below 60dB, the sound level of an average conversation; eqn. (1) shows that the compression gain is nonlinearly related to input signal power.

$$CompressionGain \propto \frac{1}{(SignalPower)^{CR-1}} \quad (1)$$

CR, the selected compression ratio, is constant during the processing.

Because of the need for large binary words in the typical analog to digital conversion techniques used by hearing instrument processors, we have the fundamental problem of binary to DBNS conversion to solve. Although look-up tables have been proposed for other DBNS DSP processors (e.g., ref 1), this approach is only useful for low resolution input words (e.g., below 13-bits). If we propose to convert into the DBNS from a 24-bit A/D converter, then we need a different solution. By analysing cyclic patterns in a sorted DBNS sequence a novel technique has been devised for converting from a binary representation to a

DBNS representation which can be built into a pipelinable structure. This technique is highlighted in this paper.

2. TWO-DIGIT DBNS (2-DBNS)

2.1. The Signed DBNS

The Double Base Number System (DBNS) is an alternative method of representing numbers in a non-linear form. It is a number system using bases 2 and 3, allowing as digits only $\{-1,0,1\}$ and requiring $o(\log N)$ nonzero digits, is the signed double base number system (SDBNS); i.e., a representation having the form of Eqn. (2) 1,

$$x = \sum_{n=1}^N s(n) \cdot 2^{b(n)} \cdot 3^{t(n)} \quad (2)$$

where $s(n) \in \{-1, 0, 1\}$, $b(n) \in \{-2^{B-1}, -2^{B-1} + 1, \dots, 2^{B-1} - 1\}$ and $t(n) \in \{-2^{T-1}, -2^{T-1} + 1, \dots, 2^{T-1} - 1\}$. Clearly the signed binary number system is a special case (and valid member) of the above representation.

2.2. Index Calculus SDBNS

The n-tuple, $\left\{ 2^{i_x} 3^{j_x} = 2^{i_x + i_y} 3^{j_x + j_y} \right\}$, immediately leads to an implementation of multiplication using index addition, where the index mapping of x , for example, is simply the n-tuple $\{i_x, j_x\}$. For cases where we are approximating the reals by fixed point numbers, it is possible to find a *single index* SDBNS for any real number with arbitrary precision by using diophantine approximation algorithms 2, 3 (see Theorem 3 from reference 1). This leads us to a multiplication technique only involving a single 2D shift, which corresponds to a pair of index additions. If only one of the numbers to be multiplied is in the single index form, multiplication will only require $O\left(\frac{\log x}{\log \log x}\right)$ addition pairs 1.

2.3. Two-Digit SDBNS

Based on the single digit arbitrary approximation result, it is clear that any real number can be represented by eqn. (3) for $N=2$ (Two-Digit SDBNS).

$$x = s(1) \cdot 2^{b(1)} \cdot 3^{t(1)} + s(2) \cdot 2^{b(2)} \cdot 3^{t(2)} \quad (3)$$

We will also use the following restriction, eqn. (4), in our definition of the two-digit SDBNS for the conversion process:

$$2^{b(1)} \cdot 3^{t(1)} \gg 2^{b(2)} \cdot 3^{t(2)} \quad (4)$$

It should be noted that the restriction imposed by eqn. (4), in general, produces an inferior approximation compared to the optimal value, but is a necessary limitation for our conversion technique.

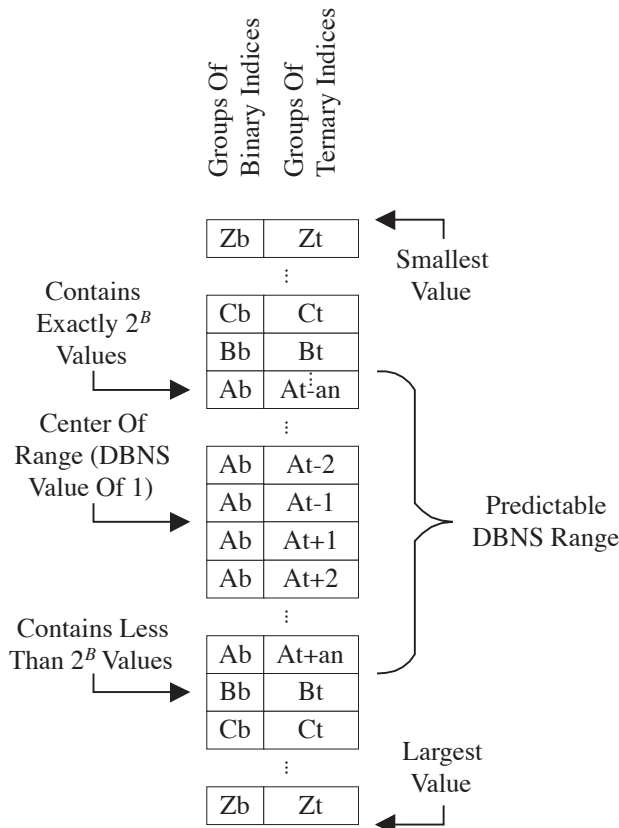
3. DBNS SEQUENCE PATTERNS

For simplicity in searching for initial patterns in a DBNS sequence we will only consider single digit ($N = 1$), positive ($s(1) = 1$) DBNS values. Therefore $x = 2^b \cdot 3^t$. Although the remaining discussion will revolve around positive DBNS representations, the methods presented here can easily be extended to negative representations as well.

3.1. Coarse Patterns

Given a sorted ascending sequence of DBNS values the binary index, b , is found to repeat every 2^B values if $T \geq B$. After each ascending replication the associated ternary index t is incremented (as demonstrated by $2^b \cdot 3^{t+1} > 2^b \cdot 3^t$) (See Figure 3). This predictable sequence of binary groups (BG) remains true (until the ternary index t overflows) for the centrally located values which are approximately $1 - \left(\frac{2^B}{2^T} \cdot \frac{1293}{2048}\right)$ percent of the total DBNS values (2^{B+T}).

Figure 3. Coarse Patterns



3.2. Semi-Coarse Patterns

Within each sequence of the BG (repeated groups of 2^B values) for $B > 6$, it is found that *approximately* every $\frac{2^B}{21}$ values, the binary index increases by 8 and the ternary index decreases by 5 (See Figure 4). This occurs since $2^8 \cdot 3^{-5} = \frac{256}{243} = 1.05349794$ (which is a good DBNS approximation of 1) and $\frac{\log(3)}{\log(2^8 \cdot 3^{-5})} \cong 21$ dictating the maximum number of segments in a BG. The overall effect is such that a number is multiplied by a slightly larger representation 1 thus generating a new number slightly higher in the sorted sequence of DBNS values. The values of 8 and -5 are very desirable because they are significantly smaller than the index range; hence, when adding or subtracting (DBNS multiplication or division) them to the indices, overflow is likely to only occur once in a single group.

Figure 4. Semi-Coarse Patterns

Binary Index	Ternary Index	
		⋮
b0+8n	t0-5n	}
b1+8n	t1-5n	
		⋮
bm+8n	tm-5n	}
b0+8(n+1)	t0-5(n+1)	
b1+8(n+1)	t1-5(n+1)	
		⋮
bm+8(n+1)	tm-5(n+1)	
		⋮

Approximately $2^B/21$ Values

3.3. Fine Patterns

To generate a sorted sequence of DBNS values each value is multiplied by the smallest representation of 1 (for the limits provided by B and T) to produce the next value in the sequence. However, due to the index representation of this approximation of 1 (large indices) overflow is very likely to occur. To correct for this, multiplication by a larger representation of 1 it is necessary to place the indices back into the operating range thus requiring at least two operations. By analyzing the trend of the indices in the sorted DBNS sequence it is found that when in the central portion of the sequence there are only 3 possible changes in the indices. These changes also resolve the overflow problem therefore resulting in a single cycle operation (See Table 1). The choice of the appropriate indices change is found by checking if overflow occurs with each possibility in priority (See Figure 5).

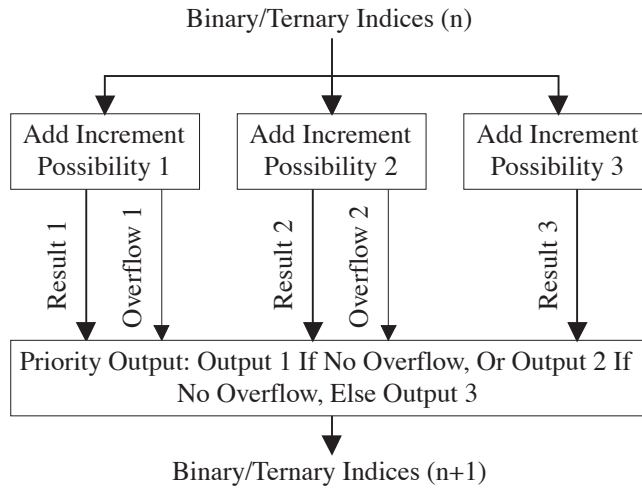
Table 1. Portion of a DBNS sequence with deltas and priority for $B=8$ and $T=8$

Real Value	Binary Index	Ternary Index	Binary Delta	Ternary Delta	Priority
1.015762097519506	-103	65	-84	53	2nd
1.025329407756846	46	-29	149	-94	3rd
1.027472668214611	-38	24	-84	53	2nd
1.029620408759804	-122	77	-84	53	2nd
1.037150278553880	111	-70	233	-147	1st
1.039318248343857	27	-17	-84	53	2nd

4. BINARY TO SINGLE DIGIT DBNS CONVERSION

By using the patterns found in the DBNS sequence above an approach to converting values in binary representation to a single digit DBNS representation is possible. This method will require numerous steps to perform; however, these steps are only forward dependent and therefore can be implemented in a pipelined structure. The basic approach of the converter is to continuously compare a computed DBNS and binary representation to the target binary value. At each step this comparison will provide information to the next stage of the conversion process to best estimate a DBNS representation.

Figure 5. Fine Patterns



4.1. Stage 1: Course Approximation

From the above description of the Coarse Patterns in a DBNS sequence the first stage of the converter will determine the BG in which the target value lies. The most efficient manner in doing this operation is to perform a binary search. To do this the usable range of the DBNS representation will be assumed to be a power of two BGs. For example, given 16 BGs it will require at least 4 or $\log_2 16$ steps to determine the final BG. The limit of 16 BGs yields $16 \cdot 2^B$ unique DBNS representations with a maximum value of $3^{16} - \epsilon$ (ϵ is the smallest DBNS value increment for B and T). Regardless of the value of B or T the first binary index in a BG (starting from the center of the sequence) is always 0. For implementation, each step will require an increasing address size LUT which contains the binary equivalent for 3^n ; the ternary indices can be computed from step to step (See Figure 6).

4.2. Stage 2: Semi-Coarse Approximation

Once the BG (or offset for t) has been determined a semi-coarse approximation is performed.

As mentioned earlier, at approximately every $\frac{2^B}{21}$ values in the DBNS sequence, the binary and ternary indices change by 8 and -5 respectively. By selecting the proper base for b each BG can be divided into 21 sections with one or no instances of overflow. Again different search algorithms may be used but the most efficient is a binary pattern. For the particular case of $B = 8, T = 8$ two options present themselves, either a datum of 0 or 4. By observation all BGs start at a datum of 0; however, the center of each BG starts approximately with a datum of 4. A datum of 0 provides us with two separate but overlapping sub-groups of 2^{B-1} elements to search; this is ideal for a binary pattern (this sub-grouping effect is due to the small size of B and T). A datum of 4 provides us with one continuous sub-group but with a non-power-of-two range (See Figure 7). For this discussion, a datum of 0 will be chosen; the datum of 4 is still under investigation. Since a datum of 0 provides two sub-groups a decision must be made on which group to use since neither support the whole BG range (due to binary overflow). A single step test is performed to check whether, or not, the target value is in the upper or lower range of the BG. Once this test is performed, the binary search can then begin. Similar to the operation above in the coarse approximation this stage should require $\log_2(B-1)$ steps (not including the sub-group decision). This procedure will also require an increasing series of LUTs as reference for the target binary comparison.

Figure 6. Steps In Coarse Approximation

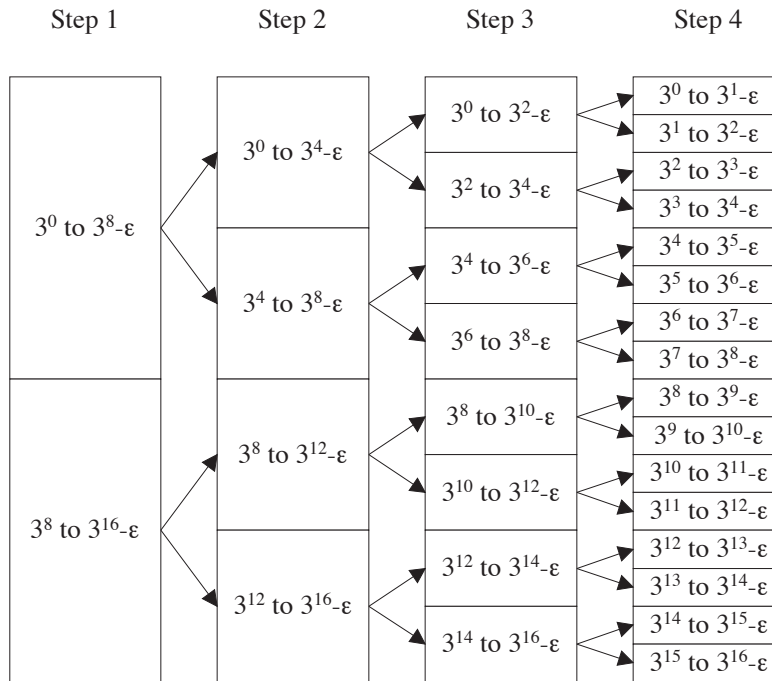
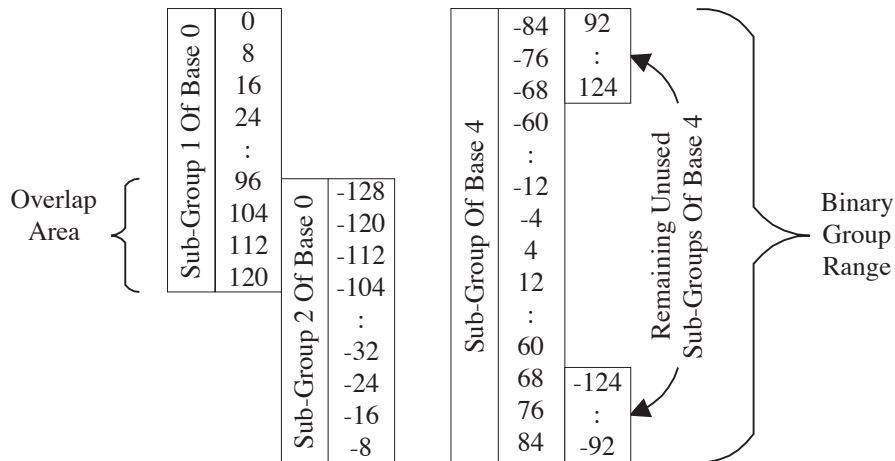


Figure 7. Binary base options available

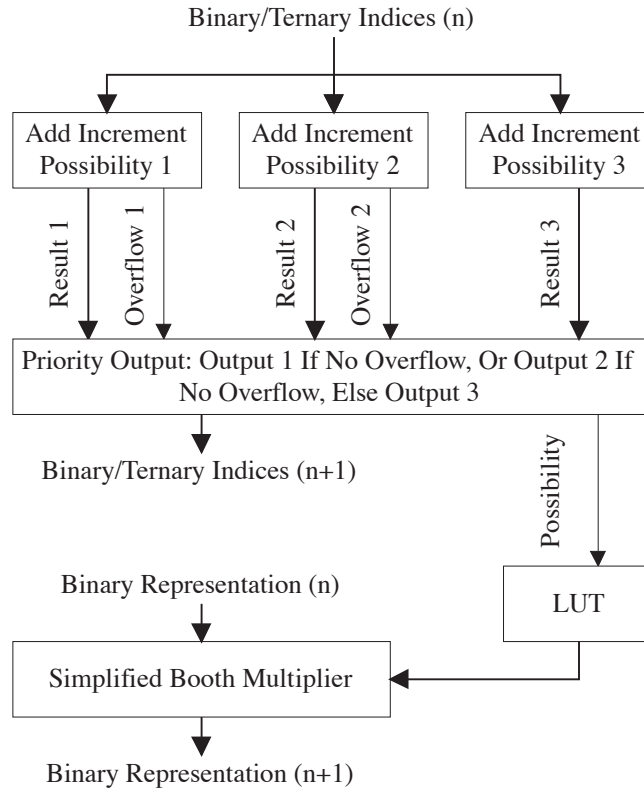


4.3. Stage 3: Fine Approximation

After the semi-coarse approximation is complete a simple series of comparisons with the “next” possible value in the DBNS sequence is performed. This stage will require up to $\frac{2^B}{21}$ steps since this is the approximate gap between the semi-coarse groups. Referring to the fine patterns described above each step performs a comparison to determine the appropriate changes in the indices. Once the right choice is made a binary equivalent of the near one multiplication is also calculate to maintain the reference to the target binary representation. Depending on the accuracy required a typical Booth or canonic signed digit mul-

multiplier with extra optimization can be implemented here since one of the multiplicands is very close to 1; see Figure 8. This stage only increments the approximation until it is greater than the target; however, it may be possible to decrease the number of steps required by introducing a decrementing mirror of this stage. The semi-coarse approximation can stop at a value above and below the target value and proceed to determine the best possible direction of estimation by the least difference from the two approximations to the target value. Fine approximations can then proceed with either an incrementing or decrementing method. This modification may require more hardware but the number of replications of the hardware pipeline will be reduced.

Figure 8. Block Diagram Of Fine Approximation Implementation



5. EXAMPLES

The following are examples (Table 2 and Table 3) showing the stage-by-stage process of the conversion. Note that the “Fine” processing stage is unidirectional only.

Table 2. Best case: estimate of 20782 with $B=8, T=8$

Stage	Binary Index	Ternary Index	Binary Reference	Processing Stage
1	0	8	6561	Coarse
2	0	12	531441	Coarse
3	0	10	59049	Coarse
4	0	9	19683	Coarse
5	-88	65	33284	Semi-Coarse Adjust
6	64	-31	29864	Semi-Coarse
7	32	-11	24245	Semi-Coarse
8	16	-1	21845	Semi-Coarse
9	8	4	20736	Semi-Coarse

Table 2. Best case: estimate of 20782 with $B=8, T=8$

Stage	Binary Index	Ternary Index	Binary Reference	Processing Stage
10	73	-37	20975	Fine
11	-76	57	20779	Fine

Table 3. Worst case: estimate of 4300750 with $B=8, T=8$

Stage	Binary Index	Ternary Index	Binary Representation	Processing Stage
1	0	8	6561	Coarse
2	0	12	531441	Coarse
3	0	14	4782969	Coarse
4	0	13	1594323	Coarse
5	-88	69	2696044	Semi-Coarse Adjust
6	-64	54	3152306	Semi-Coarse
7	-32	34	3882959	Semi-Coarse
8	-16	24	4309533	Semi-Coarse
9	-24	29	4090689	Semi-Coarse
10	125	-65	4129219	Fine
11	-43	41	4146500	Fine
12	106	-53	4185555	Fine
13	-64	53	4203071	Fine
14	87	-41	4242660	Fine
15	-81	65	4260415	Fine
16	68	-29	4300543	Fine
17	-100	77	4318541	Fine
18	-16	24	4309533	Fine
19	68	-29	4300543	Fine

6. BINARY TO MULTI-DIGIT DBNS CONVERSION

The above conversion process describes a method only suitable for single digit DBNS; however, this procedure can be extended for multi-digit DBNS.

By performing a binary to single digit conversion, the first digit of the DBNS representation can be found. This digit will represent the high or most significant component of the DBNS representation. The difference between the target binary value and the upper and lower binary approximations can then generate a starting point for the second digit. Both the upper and lower values need to be considered which will also determine the sign for the second digit (See Figure 9). The accuracy of the second digit (number of bits in the exponent representations) need not be the same as the first digit depending on the application for which the number system is being used. After the second binary to DBNS conversion is complete a comparison for the closest value to the target binary value is performed. This will return a sign and exponents for the second digit along with the correct choice for the first digit indices. Although two converters are not necessary (the sign for the second digit may be fixed) two will improve the resolution of the DBNS representation by 100% (See Figure 10). The size of the first and second digit ranges should be chosen carefully since variations in these ranges can effect the overall resolution (See Table 4, $B=T$).

Figure 9. Conversion with fixed and non-fixed signs

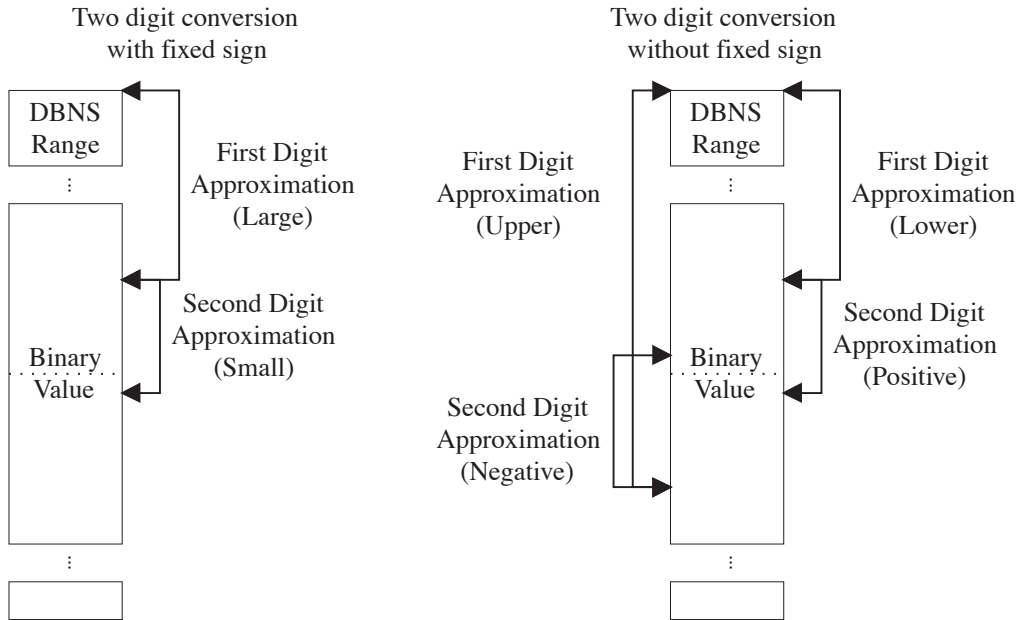


Figure 10. Implementing Two-Digit DBNS Conversion

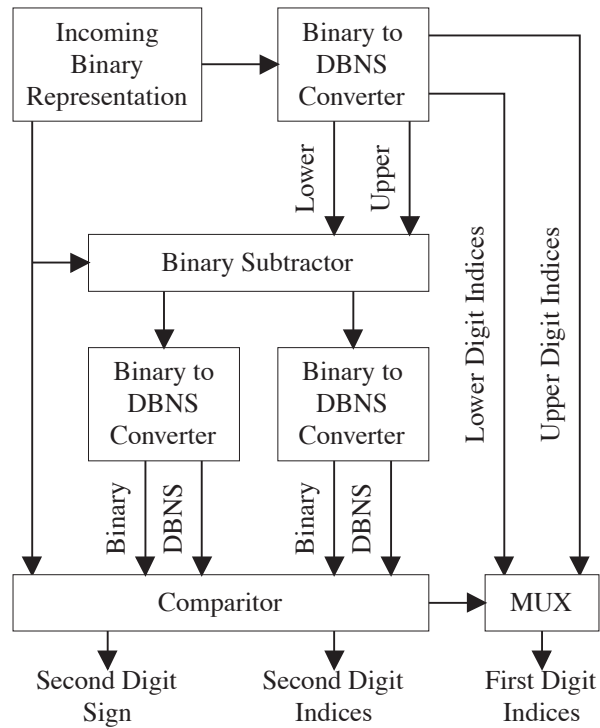


Table 4. Unique Integer Values Generated With Varying Bit Ranges

Second Digit Index Range (Bits)	First Digit Index Range (Bits)				
	6	7	8	9	10
2	-	10864	20315	38696	71679
3	20252	37559	69318	130533	237399
4	67448	121001	214614	392141	680150
5	150726	260897	446760	794254	1326712
6	278297	472950	793599	1385359	2253090
7	452422	758197	1248541	2141531	3388573
8	764620	1252248	2009798	3375348	5174199
9	1315714	2112857	3313076	5427546	7995565

7. CONCLUSIONS

In this paper we have discussed the application of a two-digit signed double-base number system for efficient representation of signal samples in hearing instrument processors. The non-linear representation, using exponents of the bases, allows a considerable reduction in the number of bits to be used to represent each signal sample. We demonstrate that a reduction from 24 to 16 bits is achieved for typical hearing instrument dynamic ranges. The logarithmic-like representation also allows simple implementation of multiplication, division and exponentiation operations, which are required in the non-linear compression algorithms used for correcting hearing impairments.

The major problem associated with mapping large binary A/D conversion outputs to the 2-digit SDBNS has been solved by studying the patterns associated with DBNS representations of input sequences. The hardware required for the mapping process is quite modest and is able to be pipelined.

We have demonstrated the application of this procedure with several examples.

ACKNOWLEDGMENTS

The authors acknowledge financial support from the Natural Sciences and Engineering Research Council of Canada, the Micronet Network of Centres of Excellence, Gennum Corporation, and design tools, workstations and fabrication services from the Canadian Microelectronics Corporation.

REFERENCES

1. V. S. Dimitrov, G.A. Jullien and W.C. Miller, 1999, "Theory and Applications of the Double-Base Number System", IEEE Trans. Computers, Vol. 48, 10, pp. 1098-1106
2. B.M.M. de-Weger, "Algorithms for Diophantine equations", CWI Tracts-Amsterdam, vol.65, 1989
3. A.J. Brentjes, Multi-dimensional continued fraction algorithms, Mathematical Centre Tracts, Amsterdam, vol.145, 1981
4. N. Magotra, P. Kasthuri, Y. Yang, R. Whitman and F. Livingston, "Multichannel Adaptive Noise Reduction in Digital Hearing Aids", Proc. IEEE Int. Symp. on Circ. and Syst., 1998, paper TAA11-5.
5. D. Lewis, "An accurate LNS arithmetic unit using interleaved memory function interpolator", Proc. of ARITH-11, Windsor, 1993 pp. 2-9.