

A Hybrid DBNS Processor for DSP Computation

G. A. Jullien, V. S. Dimitrov, B. Li, W. C. Miller, A. Lee, and M. Ahmadi

VLSI Research Group
University of Windsor
Windsor, Ontario, Canada N9B 3P4
(519) 253-4232 Ext. 2574
jullien@uwindsor.ca

ABSTRACT

This paper introduces a modification to an index calculus representation for the Double-Base Number System (DBNS). The DBNS uses the bases 2 and 3; it is redundant (very sparse) and has a simple two-dimensional representation. An extremely sparse form of the DBNS uses a single non-zero digit to represent any real number with arbitrary precision. In this case the single digit can be identified by its coordinates (indices) in the two-dimensional representation space. The modification proposed in this paper, targeted to DSP inner product computations, uses a single digit representation for the coefficient vector and a 2-digit representation for the data vector. We show that a reduction of over 80% in hardware cost is possible using this hybrid representation compared to the original single-digit technique.

1. INTRODUCTION

We have recently introduced a new double-base redundant number system with the number representation

$$h = \sum_i a_i \cdot 2^{b_i} \cdot 3^{t_i}, \text{ where } a_i \in \{-1, 0, 1\} \text{ and } b_i, \text{ and } t_i,$$

are integers we will refer to as binary and ternary exponents respectively. Clearly the binary number system is a special case (and valid member) of the above representation.

The DBNS has an unusually simple 2-D geometric interpretation, suitable, for example, for implementation via cellular automata [4] or cellular neural networks [5]. An example, a canonic representation of 79, is shown in Figure 1.

	1	2	4	8	16
1					
3					
9					
27					

x = 79

Figure 1 2-D interpretation of a DBNS representation (79)

It is important to note that, unlike the binary canonic signed-digit redundant representation, the canonic form of a DBNS representation is not, in general, unique.

If we extend the exponents within an arbitrary *signed* integer space, then it is possible to represent any real number, with arbitrary precision, using a single non-zero digit, a_j . The single digit can be mapped by its binary and ternary exponents, thus allowing an index calculus with which we can perform arithmetic using logarithmic-like computational units [1]. The single digit representation is shown in eqn. (1).

$$h = s2^b3^t \quad (1)$$

where $s \in \{-1, 0, 1\}$, and b, t are signed integers. We thus can represent h by the 3-tuple $\{s, b, t\}$. This exponent representation means that multiplication and division are easy (adding and subtracting the exponents); addition and subtraction are, of course, more difficult.

To implement multiplication and division, if we let:

$$x = (s_x, b_x, t_x) \text{ and } y = (s_y, b_y, t_y), \text{ then:}$$

$$x \cdot y = (s_x s_y, b_x + b_y, t_x + t_y) \quad (2)$$

$$x/y = (s_x s_y, b_x - b_y, t_x - t_y) \quad (3)$$

We can perform addition and subtraction within this index calculus using the identities:

$$2^a 3^b + 2^c 3^d = 2^a 3^b (1 + 2^{c-a} 3^{d-b}) \quad (4)$$

$$\approx 2^a 3^b \Phi(c-a, d-b)$$

$$2^a 3^b - 2^c 3^d = 2^a 3^b (1 - 2^{c-a} 3^{d-b}) \quad (5)$$

$$\approx 2^a 3^b \Psi(c-a, d-b)$$

We precompute and store the functions containing the approximation of:

$$\Phi(x, y) = 1 + 2^x 3^y \approx 2^\alpha 3^\beta \quad (6)$$

$$\Psi(x, y) = 1 - 2^x 3^y \approx 2^\gamma 3^\delta \quad (7)$$

Addition (subtraction) of two numbers is now mapped into the following two steps:

1. Find the corresponding element (α, β) in the table;
2. Add (subtract) (a, b) with (α, β) .

This is very similar to the techniques used for addition and subtraction in the Logarithmic Number System [6].

2. HALF-INDEX DOMAIN IPSP

Many of the basic DSP computations involve inner product processing where it is only required to have a single multiplication in any data path. We can use this to our advantage by only using the index representation for the multiplication, and then converting to a binary representation for the inner product accumulation; this avoids the addition problem and also provides the output as a binary number. An inner product step processor (IPSP) structure that performs the inner product computation of eqn. (3) (e.g. a FIR filter) is shown in Figure 2 [2].

$$y[n+1] = y[n] + h_c(n) \cdot h_d(n) \quad (3)$$

Here h_c represents the coefficient and h_d the data. We will drop the sample index, n , of the data and coefficient when it is obvious by context.

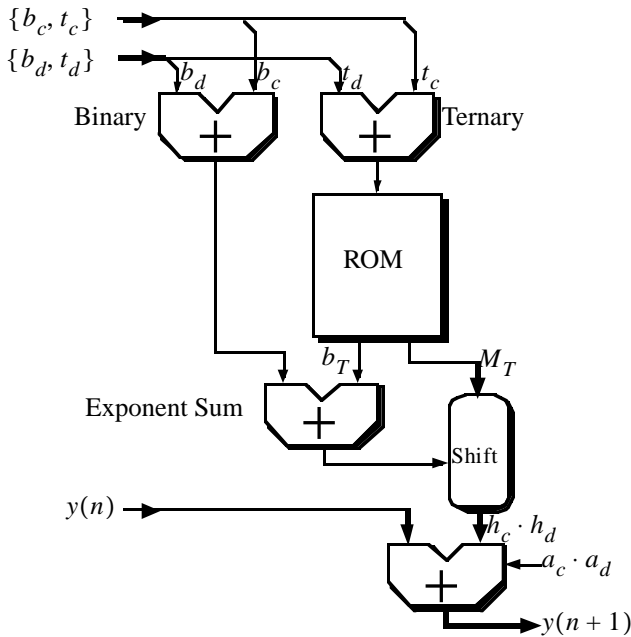


Figure 2 Single-digit DBNS ALU

In Figure 2, the binary and ternary exponents are independently added and the resulting ternary exponent is mapped to a floating point binary number $m_T \cdot 2^{b_T}$. We perform a further binary exponent addition and then convert the floating point representation to a fixed point representation using a barrel shifter. A very interesting result is that since we know

the dynamic range of the final binary representation (this can be independently computed from a knowledge of the data and coefficient range) we have an apriori limit on the maximum left and right shift of the barrel shifter. As an example, we may use 12-bit binary exponents but have a limit of ± 15 shifts on the barrel shifter. Thus, in this case, we only need to

compute the binary exponents with a $\text{Mod}(2^5)$ adder; i.e., a 5-bit 2's complement adder. We have therefore gained extra computational efficiency in addition to the independent binary/ternary adders.

The accumulation is carried out in the binary number system. The area complexity is $O(2^{(t_c + t_d)})$ (dominated by the ROM), and so it is important to be able to reduce the ternary exponents while maintaining the required precision over the entire dynamic range. Since the index mapping is logarithmic in each base, we experience the usual problem of having good precision for small numbers and poor precision for large numbers. For FIR filter coefficients, this does not turn out to be a problem since most of the coefficients are clustered at the low end of the dynamic range (see Figure 3 [3]).

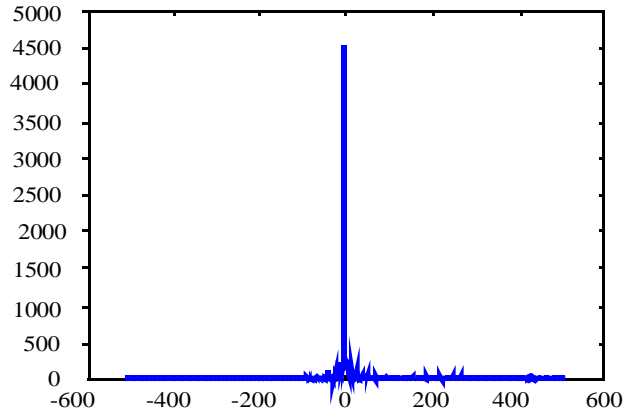


Figure 3 Distribution of filter coefficients for a mix of 200 L-P, B-P and H-P FIR filters

In addition, we also design the filters within an envelope and so an optimization algorithm can be used to minimize the maximum value of the ternary exponent. Figure 4 shows a low-pass filter designed with a genetic algorithm [7] for which the maximum ternary index is 196 (Table 1); the filter has a symmetric (linear phase) 53-tap impulse response.

The precision problem is in the data mapping, which is normally required to have less than $\pm 1/2$ -bit precision over the entire dynamic range. A histogram of $\pm 1/2$ -bit and ± 1 -bit errors for a 12-bit ternary exponent representation of 12-bit integers is shown in Figure 5. Note the histogram growth as the represented number is increased. We require 13-bit ternary exponent precision to guarantee less than $1/2$ -bit error.

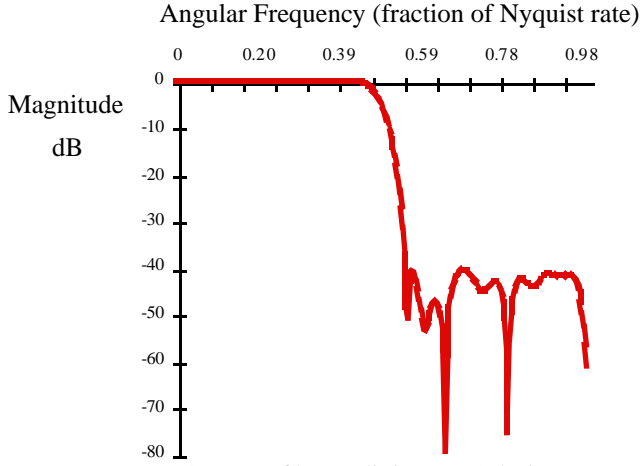


Figure 4 53 tap filter 1-digit DBNS design

Table 1: 1-Digit filter coefficients

#	a	b	t	#	a	b	t
0	1	-55	28	15	1	-95	56
1	-1	-273	160	16	-1	-155	85
2	-1	-260	156	17	-1	110	-73
3	1	194	-135	18	-1	-36	16
4	1	-283	171	19	1	158	-103
5	-1	-248	150	20	1	182	-125
6	-1	-280	170	21	-1	-152	93
7	1	67	-48	22	-1	23	-25
8	-1	157	-111	23	1	-264	164
9	-1	100	-68	24	1	-97	51
10	1	-45	15	25	-1	-184	114
11	1	10	-11	26	-1	-139	76
12	1	250	-169	27	1	309	-196
13	-1	-21	9	28	1	83	-53
14	-1	6	-13				

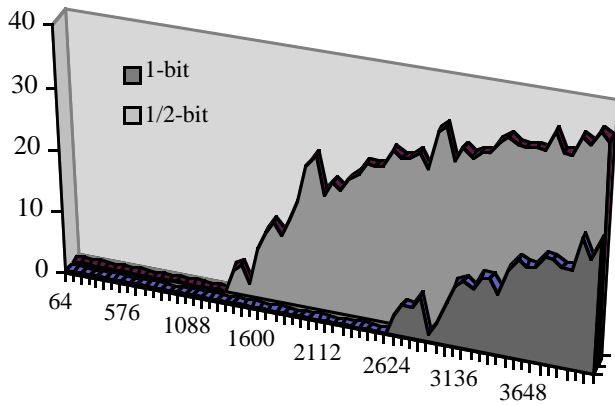


Figure 5 $\pm 1/2$ -bit and ± 1 -bit errors for 12-bit exponents

3. A HYBRID DBNS ARITHMETIC

To reduce the area complexity associated with a large ternary exponent look-up table, we introduce a hybrid representation, in which the data is represented with 2-digits, as in eqn. (4), while the coefficients are represented with a single digit.

$$h = s_1 2^{b_1} 3^{t_1} + s_2 2^{b_2} 3^{t_2} \quad (4)$$

One way of looking at the 2-digit representation is to regard the second digit of the data as providing the precision “fine-tuning” of the number represented by the first digit (although in practice there is no restriction on the relative sizes of the two components). Since the purpose of the 2-digit representation is to reduce the sum of the ternary exponents in an index calculus multiplication, we need to verify that each of the 2-digit ternary exponents are smaller than the 1-digit exponents by at least 1-bit. In practice, the reduction is much more than that. Table 2 shows some examples of 2-digit exponents, with an error $< 1/2$ -bit, compared to their 1-digit counterpart.

Table 2: Comparison between 1-digit and 2-digit exponents

h_d	a_{d1}	b_{d1}	t_{d1}	a_{d2}	b_{d2}	t_{d2}	b_d	t_d
3568	-1	-13	14	1	-7	12	2394	-1503
3569	1	4	4	1	27	-10	2394	-1503
3570	-1	-9	12	1	9	2	-2876	1822
3571	1	-30	28	-1	-16	19	-2876	1822
3572	1	-31	27	1	25	-13	2879	-1809
3573	1	18	-4	1	29	-13	1825	-1144
3574	-1	2	5	1	28	-10	-2391	1516

We see that the 2-digit ternary exponents require only 6-bits (including the sign) versus the 13-bits required by the 1-digit ternary exponents.

4. A HYBRID 2-DIGIT PROCESSOR

A diagram of the hybrid processor is shown in Figure 6. The processor is nothing more than 2 independent 1-digit channels (Figure 2) with a final binary adder to sum the channel components. Note that if we use 2-digit representations for the filter coefficients, then 4 channels are required with a more complex summing architecture. In Figure 6 we have removed all signal notation except for the binary and ternary exponents. Since the hybrid processor requires 2 channels versus one channel for the original processor, then if the 2-digit approximation reduces the precision required of the ternary exponent sum by n bits, then we will have a reduction in the complexity by 2^{n-1} ; i.e. we will “break even” at $n=1$.

Our data fitting optimizer reveals a ternary exponent for each digit within the range $-20 \leq t_1, t_2 \leq 28$. The ternary index, t , for the filter coefficients is in the range $-196 \leq t \leq 171$. This means that the ternary index sum is $-206 \leq t_s \leq 199$ which fits into a 9-bit table. Since the original exponent sum required 13-bits, we have $n=4$ and so the hybrid processor will be 8 times more area efficient than the original single-digit processor. This translates to more than an 80% reduction in area with an attendant reduction in power.

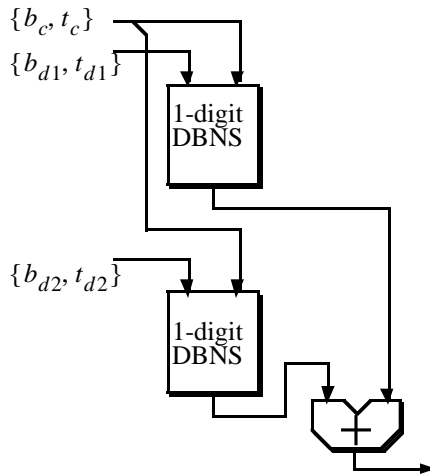


Figure 6 Hybrid DBNS Processor

Another interesting result is that the 1-digit processor is quite similar to the Fermat ALU step processor [3], and so library elements can be shared for the 2 processors. The similarity arises from the fact that the Fermat ALU uses finite field indices for the multiplication and a look-up table and modified binary arithmetic for the accumulation stage.

5. SIMULATION AND TEST

The entire hybrid processor has been simulated at the behavioural level, and the concept is shown to be sound. Figure 7 shows a screen shot of a simulation of a 5-tap filter; the 2 independent channels with the final accumulator are clearly shown. We have also simulated a 0.35 μ CMOS test chip of components for the hybrid processor and our initial measurements reveal a power savings of over 50% compared to an equivalent binary arithmetic processor. A complete 0.35 μ CMOS processor for a programmable 53-tap filter is currently in the design stage.

6. CONCLUSIONS

We have introduced a modification to the DBNS index calculus inner product processor. The area complexity associated with the ternary converter has been reduced by coding the input data vector with 2-digits and the coefficient vector with 1-digit. The new architecture requires 2 independent pro-

cessing channels but each channel has a much smaller address space for the ternary look-up table. The final architecture provides more than 80% area savings and 50% power savings over its 1-digit counterpart.

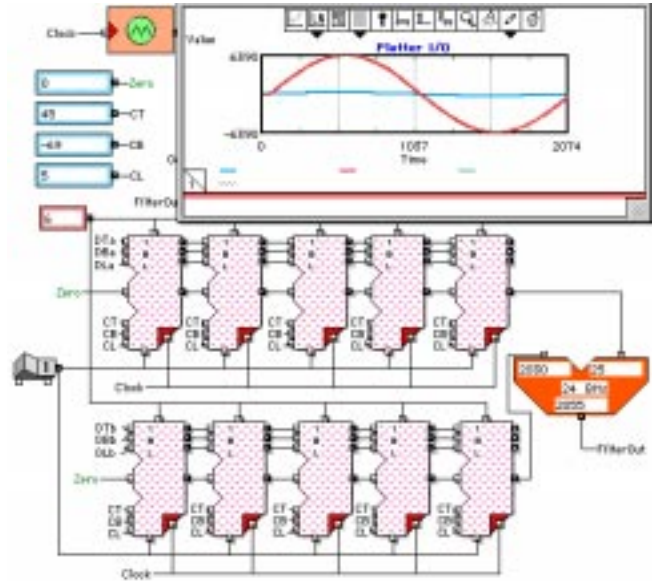


Figure 7 5-tap filter simulator

7. ACKNOWLEDGMENT

The authors would like to acknowledge financial support from the Natural Sciences and Engineering Research Council and the Micronet Network of Centres of Excellence. We are indebted to the Canadian Microelectronics Corporation for their equipment loan and fabrication services.

8. REFERENCES

- [1] V. Dimitrov, G.A. Jullien, W.C. Miller, 1997, "Theory and Applications for a Double-Base Number System," Proc. 13th IEEE Symp. on Comp. Arithmetic, pp. 44-53.
- [2] V. Dimitrov, G.A. Jullien, W.C. Miller, 1999, "Theory and Applications of the Double-Base Number System", submitted to IEEE Trans. Computers.
- [3] M. Shahkarami, G.A. Jullien, W.C. Miller, "Designing FIR Filters With Enhanced Fermat ALUs," Proc. 1999 Inter. Symp. on Circ. and Syst., Paper 17.7.
- [4] E. Swartzlander, "Digital optical computing", Applied Optics, vol.25, 1986, pp. 3021-3032
- [5] S. Sadeghi-Emamchaie, G.A. Jullien, V. Dimitrov, W.C. Miller, 1998, "Digital Arithmetic using Analog Arrays," Proc. 8th GLS on VLSI, pp. 202-207.
- [6] D. Lewis, "An accurate LNS arithmetic unit using interleaved memory function interpolator", Proc. ARITH11, Windsor, 1993, pp. 2-9.
- [7] A. Lee, M. Ahmadi, G. A. Jullien, W. C. Miller, and R. S. Lashkari, "Design of 1-D FIR Filters with Genetic Algorithm", Proc. 1999 Inter. Symp. on Circ. and Syst., 17.9.