# A 2-DIGIT DBNS FILTER ARCHITECTURE

**J. Eskritt, R. Muscedere, G.A. Jullien, V.S. Dimitrov and W.C. Miller**
VLSI Research Group, University of Windsor
Windsor, Ontario, Canada N9B 3P4
jullien@uwindsor.ca

**Abstract - We have previously reported on a novel number representation using 2 bases which we refer to as the double-base number system (DBNS). Our preferred implementation uses the relatively prime bases {2,3}. If we allow the exponents of the bases to be arbitrarily large signed integers, then we can represent any real number to any arbitrary precision by a single digit DBNS representation. By representing the digit position by the exponent values, we generate a logarithmic-like representation which we can manipulate using an index calculus. A multiplier accumulator architecture for a FIR filter application has been reported which uses a half-index domain to remove the problem of addition within the index calculus. In this paper we show that using a 2-digit DBNS representation for both the input data and the filter coefficients can result in substantial hardware savings compared to both the single-digit a DBNS approach and an equivalent binary implementation of a general multiplier accumulator. In the paper we discuss the filter architecture, techniques for converting between binary and the 2-digit DBNS representations, and also the design technique used to generate the 2-digit DBNS FIR filter coefficients.**

## INTRODUCTION

### Double Base Number System (DBNS)

We have recently introduced a new double-base redundant number system with the number representation, $h = \sum_i a_i \cdot 2^{b_i} \cdot 3^{t_i}$, where $a_i \in \{-1, 0, 1\}$ and $b_i$, 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 [6] or cellular neural networks [7]. An example of a canonic representation of 79 is shown in Figure 1.

0-7803-6488-0/00/$10.00 © 2000 IEEE 

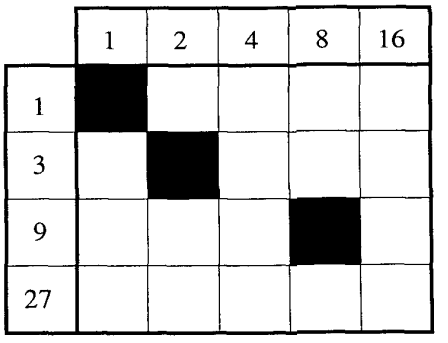


**Figure 1. A 2-D representation of a DBNS representation (79)**

It is important to note that, unlike the binary redundant representation, the canonic form of a DBNS canonic signed-digit 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, $a_j$. By "digit" we mean a 3-tuple $\{s, b, t\}$, where $s$ is a sign, $b$ is a binary exponent, and $t$ is a ternary exponent. 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).

$$h = s2^b3^t \tag{1}$$

where $s \in \{-1, 0, 1\}$, and $b$, $t$ are signed integers.

**Index Calculus**

This exponent representation provides relatively easy multiplication and division operations (by adding and subtracting the exponents); addition and subtraction are, of course, more difficult.

To implement multiplication and division, if we let:

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

$$x \cdot y = (s_x s_y, b_x + b_y, t_y + t_y) \tag{2}$$

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

We can perform addition and subtraction within this index calculus using look-up tables, in a similar fashion to logarithmic number system[2][8].

**Half-Index Domain (IPSP)**

Many of the basic DSP computations involve inner processing where it is only required to have a single multiplication product 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. (4) (e.g. a FIR filter) is shown in Figure 2 [2].

$$y[n+1] = y[n] + h_c(n) \cdot h_d(n) \tag{4}$$

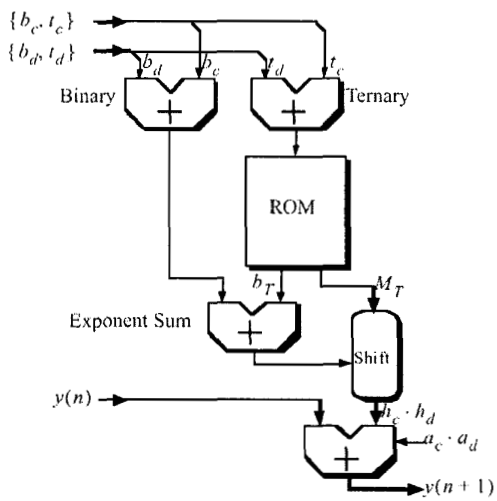


**Figure 2. Single-Digit DBNS ALU**

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.

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.

The accumulation is carried out in the binary number system. The area complexity is $O\left(2^{(t_c + t_d)}\right)$ (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.

## 2 DIGIT DBNS

### Hybrid DBNS

To reduce the area complexity associated with a large ternary exponent look-up table, we introduced *a hybrid representation* [3], in which the data are represented with 2-digits, while the coefficients are represented with a single digit.

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 theory there is no restriction on the relative sizes of the two components). An advantage of the 2 digit representation is that it "smooths out" the non-linear representation of numbers, making the representation more linear.

We are, for example, able to represent 10-bit signed binary data accurately (the map is one-to-one) in 2 digit DBNS using only a 5-bit binary exponent and a 4-bit ternary exponent. The small ternary exponent of the data allows us less restriction on the ternary exponent of the coefficients.

### Full 2 Digit DBNS

A further reduction of the area complexity associated with the ternary ROM is a full 2 digit representation. By allowing two digit coefficients, as well as 2 digit data, the size required by the look-up table diminishes exponentially. The size of the 2 digit representation exponents will be, at worst, half the size of a comparable 1 digit representation exponent.

## FILTER DESIGN

The design technique used for the hybrid DBNS filter coefficients was a genetic programming algorithm [3][9]. This process was computationally complex and did not produce very good results. Unlike designing fixed word length (or integer) coefficients, where the use of an integer programming procedure provides much better results than rounding the results of an infinite precision design, we find that a 2-digit DBNS representation has the ability to more accurately represent the real coefficients produced by such a design procedure. We therefore have a very simple and fast procedure for designing DBNS filter coefficients, namely use a standard infinite precision design procedure (e.g., a Remez Exchange algorithm) and then select the closest 2-digit representation to each of the coefficients. The error produced by this mapping is very small as seen in Figure 3 for an 85dB stopband filter with 53 taps and passband width of 0.3820 of the filter range.

A modification of the greedy algorithm[1][2] was used to produce the 2 digit DBNS approximation of the filter coefficients. The algorithm performs a search over the set of closest DBNS approximations twice (for the first and second digit).

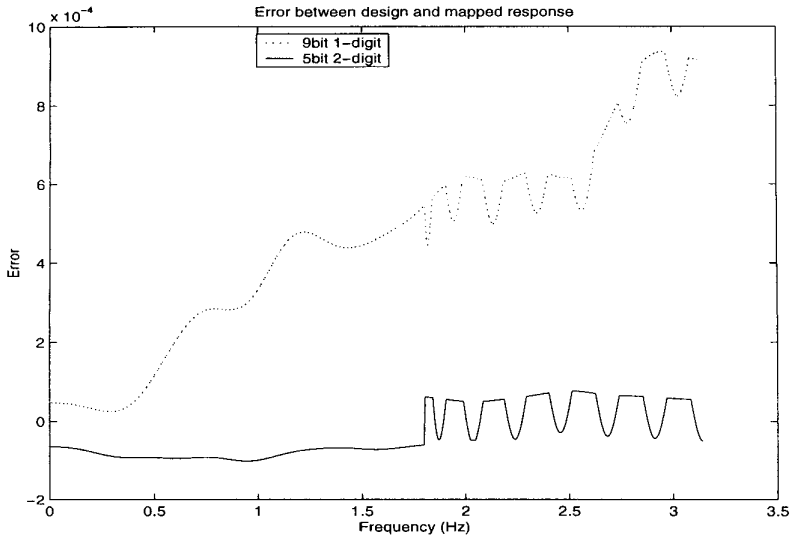


**Figure 3. Error between mapped coefficient magnitude response and designed filter response**

The mapped filters produced a 60dB stopband filter, using 9 bit single digit coefficients, and an 80dB stopband filter using 5bit binary, 4bit ternary 2 digit coefficients. The filter responses can be seen in Figure 4. The error between the magnitude response of the designed filter and the magnitude response of the mapped filter in the order of $10^{-4}$ for the single digit mapping and $10^{-5}$ for the 2 digit mapping.

The results of the direct mapping to DBNS was compared to a direct mapping to 10 and 12 bit fixed point binary coefficients. This comparison showed that DBNS numbers are much more suited to a direct mapping scheme and the error was two orders of magnitude better for the 2 digit mapping (Figure 5).

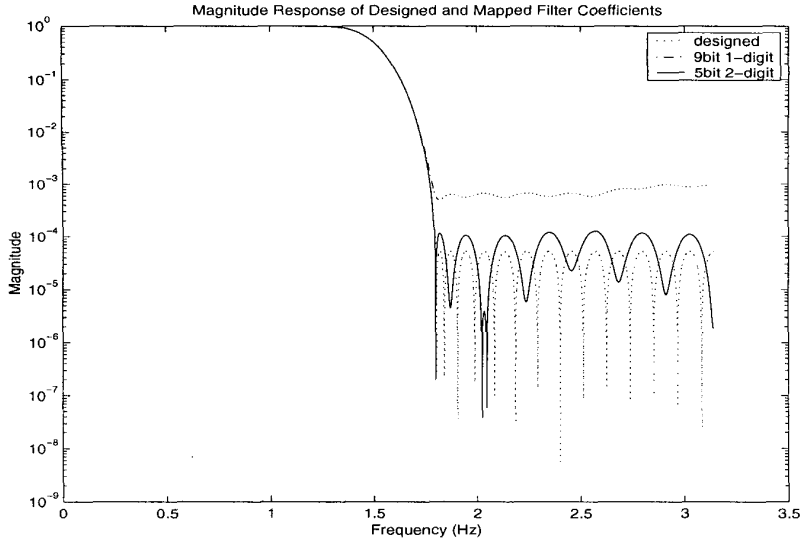


**Figure 4. Magnitude response of designed and mapped DBNS filters**

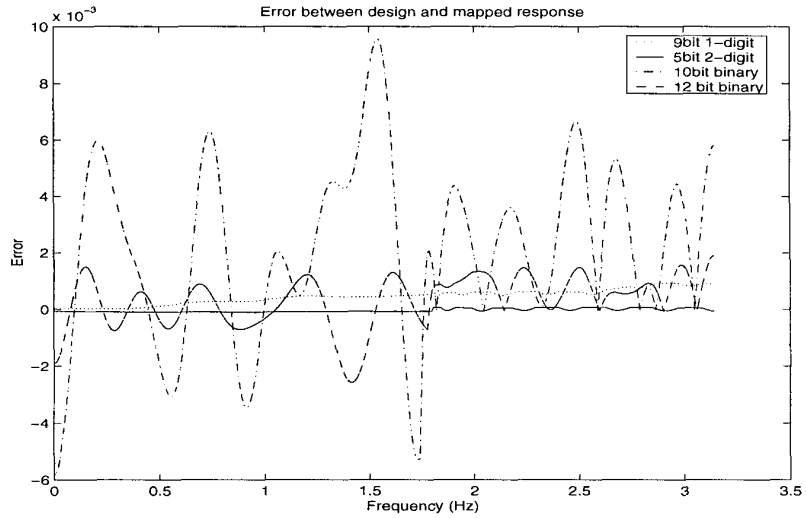


**Figure 5. Error between designed filter response and mapped binary and DBNS filters**

## ARCHITECTURE

The architecture of a single channel of the filter is based on a standard systolic array convolver, Figure 6 [10]. The number of channels needed by the filter is equal to the number of coefficient digits, multiplied by the number of data digits. Each node of the systolic array is made up of a single digit DBNS ALU (Figure 2). The output end of each channel is simply summed with the output of all the other channels to produce the final result.

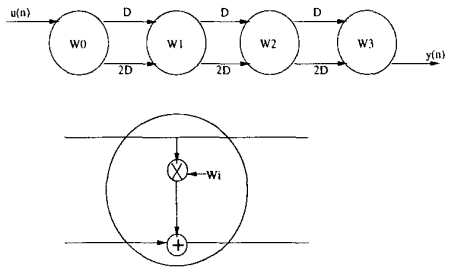


**Figure 6. Systolic array convolver architecture**

The multiplication stage shown in Figure 6, is demonstrated by equation 2. The physical design of this stage consists of two binary adders (the top two adders in Figure 2). In the case of full 2 digit implementation a single multiplicative step across all four channels is encapsulated in equation 5. Each element represents an operation within a single channel. The variables $a_i$ and $b_i$ are very small integers (5 bit for $a_i$ and 4 bit for $b_i$).

$$\begin{aligned}&(\pm 2^{a_1}3^{b_1},\pm 2^{a_2}3^{b_2})\cdot(\pm 2^{a_3}3^{b_3},\pm 2^{a_4}3^{b_4})\\ =\ &(\pm 2^{a_1+a_3}3^{b_1+b_3},\pm 2^{a_1+a_4}3^{b_1+b_4},\pm 2^{a_2+a_3}3^{b_2+b_3},\pm 2^{a_2+a_4}3^{b_2+b_4})\end{aligned} \tag{5}$$

## DESIGN AND FABRICATION

Using the 10 bit binary, 10 bit ternary 57 tap hybrid 2-digit design (see [3]), we described the architecture using VHDL, and synthesized, simulated and successfully submitted the chip for fabrication. The layout is shown in Figure 7. The entire procedure only took 2 weeks, demonstrating that the design of DBNS filters does not require exceptional effort.

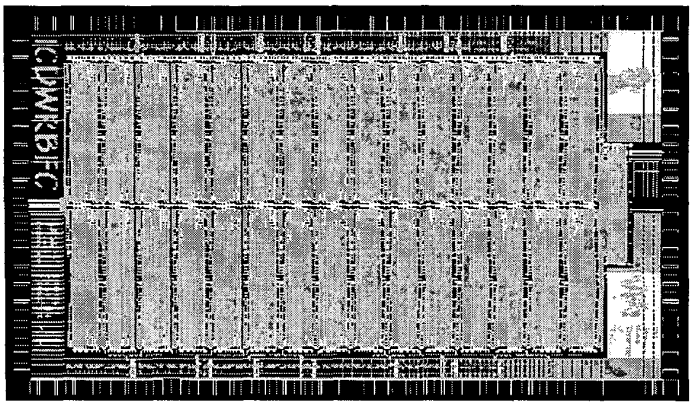

**Figure 7. 53-tap 2-digit DBNS filter layout**

## CONCLUSIONS

In this paper we have introduced a FIR filter architecture that uses a 2-digit DBNS representation for both the input data and filter coefficients. We have shown that the more precise mapping of infinite precision filter coefficients to the 2-digit representation allows a very simple filter design procedure to be used.

We have also demonstrated that the physical design of a DBNS filter chip is straight forward, and does not require any special effort, or tools.

The full 2 digit DBNS filter produces a much smaller ternary coefficient. This represents a great reduction in the area consumed by the look-up table that is an integral part of the half index domain architecture used for the filter. For the full 2-digit implementation, the size of the lookup table ROM is reduced by nearly an order of magnitude.

## ACKNOWLEDGMENTS

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 Cana-

dian Microelectronics Corporation for their equipment loan and fabrication services.

## 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," IEEE Trans. Computers, vol. 48, No. 10, Oct. 1999, pp.1098-1107

[3] G. A. Jullien, V. S. Dimitrov, B. Li, W. C. Miller, A. Lee, and M. Ahmadi, "A Hybrid DBNS Processor for DSP Computation," IEEE Int. Symp. on Circ. and Syst., May 1999, Orlando, FL, pp. 29-32

[4] R. Muscedere, G. A. Jullien, V. S. Dimitrov and W. C. Miller, "Non-linear signal processing using index calculus DBNS arithmetic," in print, SPIE 2000 Conference on Advanced Algorithms and Architectures for Signal Processing, San Siego, CA

[5] 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.

[6] E. Swartzlander, "Digital optical computing," Applied Optics, vol.25, 1986, pp. 3021-3032

[7] 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.

[8] D. Lewis, "An accurate LNS arithmetic unit using interleaved memory function interpolator," Proc. 11th IEEE Symp. on Computer Arithmetic, Windsor, 1993, pp. 2-9.

[9] 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.

[10] S. Y. Kung, "VLSI Array Processors", Prentice Hall, 1988