

# On the Realization of Asymmetric High Radix Signed Digital Adder using Neural Network

Md. Sumon Shahriar<sup>†</sup>

Dept of Computer and Information science  
University of South Australia  
sumon\_shahriar@yahoo.com

Tofail Ahammad

<sup>†</sup>Dept of Computer Science and Engg.  
BRAC University, Dhaka, Bangladesh  
tofailahammad@yahoo.com

Hafiz Md Hasan Babu

Dept of Comp. Sc. and Engg.  
University of Dhaka, Bangladesh  
hafizbabu@hotmail.com

## Abstract

This paper presents an asymmetric high-radix signed-digit (AHSD) adder that performs addition on the basis of neural network (NN) and also shows that the AHSD number system supports carry-free (CF) addition by using NN. Besides, NN implies the simple construction in high-speed operation. The signed-digit number system represent the binary numbers that uses only one redundant digit for any radix  $r \geq 2$ , the high-speed adder in the processor can be realized in the signed-digit system without a delay of the carry propagation. A Novel NN design has been constructed for CF adder based on the AHSD<sub>(4)</sub> number system is also presented. Moreover, if the radix is specified as  $r = 2^m$ , where  $m$  is any positive integer, the binary-to-AHSD<sub>(r)</sub> conversion can be done in constant time regardless of the word-length. Hence, the AHSD-to-binary conversion dominates the performance of an AHSD based arithmetic system. In order to investigate how the AHSD number system based on NN design achieves its functions, computer simulations for key circuits of conversion from binary to AHSD<sub>(4)</sub> based arithmetic systems are made. The result shows the proposed NN design can perform the operations in higher speed than existing CF addition for AHSD.

## 1 Introduction

Addition is the most important and frequently used arithmetic operation in computer systems. Generally, a few methods can be used to speed up the addition operation. One is by using neural network design convert the operands from the binary number system to a redundant number system, e.g., the signed-digit number system [2][3] or the residue number system, so that the addition becomes carry-free (CF). This Neural Network (NN) design implies fast addition can be done, at the expense of conversion between the binary number system and the redundant number system. In this paper we focus on exploring high radix signed-digit (SD) numbers and we also present the *asymmetric high-radix signed-digit* (AHSD) number system using NN [4][5]. The idea of AHSD is not new. Instead of proposing a new number representation, our purpose is to explore the inherent CF property of AHSD by using NN. The CF addition in AHSD based on NN is the basis for our high-speed addition circuits. The conversion of AHSD to and from binary will be discussed in detail. By choosing  $r = 2^m$ , where  $m$  is any positive integer, a binary number can be converted to its canonical AHSD representation in constant

time. We will also present one simple algorithm for converting binary bit pattern to make pairs in AHSD for radix- $r$ . NN design for converting AHSD numbers to binary: the first stresses high speed and the other provide hardware reusability. Since the conversion from AHSD to binary has been considered the bottleneck of AHSD-based arithmetic computation based on NN, these NN design greatly improve the performance of AHSD systems. For illustration, we will discuss in detail the example on AHSD<sub>(4)</sub>, i.e., the radix-4 AHSD number system. The proposed approach is practical thanks to the simple conversion.

## 2 Basic definitions

Here, few basic definitions will be discussed.

### 2.1 Neural Network

A neural network is a powerful data-modelling tool that is able to capture and represent complex input/output relationships. The motivation for the development of neural network technology stemmed from the desire to develop an artificial system that could perform "intelligent" tasks similar to those

performed by the human brain. Neural networks resemble the human brain in the following two ways:

1. A neural network acquires knowledge through learning.
2. A neural network's knowledge is stored within inter-neuron connection strengths known as synaptic weights.

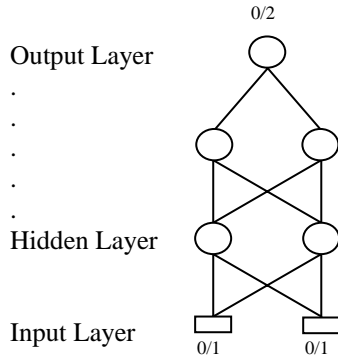


Figure 1: Neural Network prototype for AHSD number system addition.

In the figure, the simple prototype for NN is shown.

## 2.2 Asymmetric Number System [1][3]

The radix- $r$  asymmetric high-radix signed-digit (AHSD) number system, denoted  $AHSD(r)$ , is a positional weighted number system with the digit set  $S_r = \{-1, 0, \dots, r-1\}$ , where  $r > 1$ . The AHSD number system is a minimally redundant system with only one redundant digit in the digit set. We will explore the inherent carry-free property in AHSD and develop systematic approaches for conversion of AHSD numbers from binary ones.

An  $n$ -digit number  $X$  in  $AHSD(r)$  is represented as  $X = (x_{n-1}, x_{n-2}, \dots, x_0)_r$ ,

Where  $x_i \in S_r$  for  $i = 0, 1, \dots, n-1$ , and  $S_r = \{-1, 0, 1, \dots, r-1\}$  is the digit set of  $AHSD(r)$ . The value of  $X$  can be represented as

$$X = \sum_{i=0}^{n-1} x_i r^i$$

Clearly, the range of  $X$  is  $\left[\frac{1-r^n}{r-1}, r^n-1\right]$ .

## 2.3 Binary to Asymmetric Number System conversion

Since the binary number system is the most widely used, we have to consider the conversion between the AHSD and binary number systems. Although the radix  $r$  may be any positive integer, simple bi-

nary-to-AHSD conversion can be achieved if  $r = 2^m$  for any positive integer  $m$ . The reason for such simple conversion will be explained later. We assume  $r = 2^m$  in what follows, unless otherwise specified. Note that there may be more than one  $AHSD(r)$  number representation for a binary number. For instance, the binary number  $(0, 1, 1, 0, 0)_2$  can be converted to two different  $AHSD(4)$  numbers, i.e.,  $(1, -1, 0)_4$  and  $(0, 3, 0)_4$ . Hence, the binary-to- $AHSD(r)$  conversion, being a one-to-many mapping, may be performed by several different methods. We would like to find an efficient and systematic conversion method that takes advantage of the carry-free property. Here we follow a general algorithm to make pairs to convert Binary to  $AHSD(r)$ .

*Algorithm for the conversion of AHSD from binary number system:*

- Step 1.** Suppose given binary #bits= $n$
- Step 2.** If radix= $2^m$ , where  $m$  is any positive integer then  $2^p < m < 2^{p+1}$  Where  $p=1,2,3,\dots$
- Step 3.** # Zero (0) will be padded in front of binary bits pattern  $2^{p+1}-n$
- Step 4.** Divide the array by  $m$
- Step 5.** If each sub array is  $=m$   
Then stop
- Step 6.** Else  
Divide each sub array by  $m$

*Proof:*

Recurrence relation for the conversion from binary-to- AHSD numbers system is:

$$T(n) = \begin{cases} c & \text{if } n=m \text{ where } c \text{ is a constant \& } m > 2 \\ m T(n/m) & \text{if } n > m \end{cases}$$

$$T(n) = m T(n/m) \dots \dots (1)$$

If  $n = n/m$

Replace it into  $\dots (1)$

$$T(n/2) = m T(n/m^2)$$

$$\text{So, } T(n) = m^2 T(n/m^2)$$

.

.

$$T(n) = m^k T(n/m^k) \text{ where } k=1,2,3,\dots$$

$$\text{Assume } n = m^k$$

$$T(n) = m^k T(m^k / m^k)$$

$$T(n) = m^k T(1)$$

$$T(n) = m^k \quad n = m^k$$

$$\text{Log}_m n = \text{Log}_m m^k = k$$

*Complexity of the algorithm to make pairs to convert Binary to  $AHSD(r)$ :  $O(\log_m n)$*

## 2.4 Addition of AHSD $(4)$ number system [1]

Here the addition process will be shown for AHSD number system. This addition process can be for 1-bit to n-bit adder design without considering the carry propagation and its delay as well.

*Example:* Here two 4-bit AHSD  $(4)$  numbers are added.

$$\begin{array}{r}
 X \rightarrow 11\ 11\ 11\ 11 \\
 Y \rightarrow 11\ 00\ 01\ 00 \\
 \dots\dots\dots \\
 X\ (\text{ahsd}_{(4)}) \rightarrow (3\ 3\ 3\ 3)_4 \\
 Y\ (\text{ahsd}_{(4)}) \rightarrow (3\ 0\ 1\ 0)_4 \\
 \dots\dots\dots \\
 X+Y(Z): \quad 6\ 3\ 4\ 3
 \end{array}$$

$$\begin{array}{r}
 \text{Transfer digit: } 1\ 1\ 1\ 1\ 0\ c \\
 \text{Interim sum: } \quad 2\ -1\ 0\ 1\ \mu
 \end{array}$$

$$\text{Final sum S: } (1\ 3\ 0\ 1\ 1)_4$$

$$\text{Result} = (01\ 11\ 00\ 01\ 01)_2 = (453)_{10}$$

The final result is in binary format. The given example illustrates the addition process without carry propagation.

## 3 Proposed Design of adder using Neural Network

The neuron will work on the basis of feed-forward network with parallel processing. This technique can be viewed as doing addition in adder in parallel. The concept of parallel addition using neural network can be shown as the block diagram below.

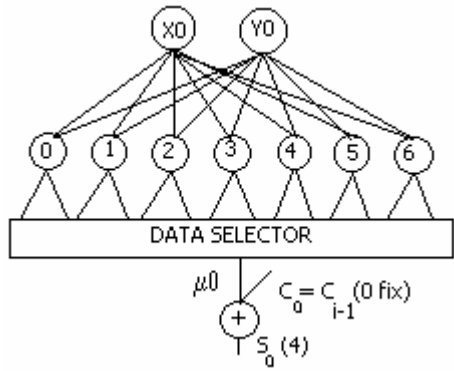


Figure 2: Trivial 1-bit AHSD radix-4 adder using NN.

The figure 2 shows the atomic view of the adder using Neural Network. If we make it more generalized then the figure will be just like the following.

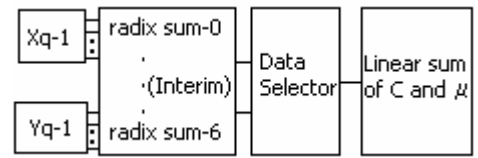


Figure 3: N-bit adder generalization.

The N-bit adder generalization is shown in figure 3.

*Lemma:* The total number of neurons for generating interim sum and carry of a radix-n asymmetric q-bit adder design is  $q \times 2(n-1)$ .

*Proof:* As  $n$  is the radix, so each bit will contain the value of  $n-1$ . So the interim sum will be in the range of 0 to  $2(n-1)$ . The total number of neurons for 1-bit adder design will be  $2(n-1)$ . For the design of a q-bit adder, the total number of neurons will be  $q \times 2(n-1)$ .

Here we proposed an algorithm for n-bit adder from binary to AHSD  $(4)$  using NN.

*Algorithm:*

**Step 1:** Create  $4^n$  input vectors ( $2n$  elements each) that represent all possible input combinations. For example, for 1-bit addition, there will be 4 input vectors (2 elements each):  $\{0, 0\}$ ,  $\{0, 1\}$ ,  $\{1, 0\}$ ,  $\{1, 1\}$ .

**Step 2:** Create  $4^n$  output vectors (ceiling of  $\lceil n/2 + 1 \rceil$  elements) that represent the corresponding target combinations. For example, for 1-bit addition, there will be 4 output vectors (2 elements each):  $\{0, 0\}$ ,  $\{0, 1\}$ ,  $\{0, 1\}$ ,  $\{0, 2\}$ .

**Step 3:** Create a feed-forward back propagation neural network.

**Step 4:** Train the neural network for the input and output vectors of *steps 1* and *step 2*.

**Step 5:** Simulate the neural network with the input vectors of *step 1*, getting the sum in AHSD  $(4)$  as the target.

## 4 AHSD addition for Radix-5

The asymmetric high radix number system considering radix-5 is not well suited for addition. The first conversion from binary to AHSD requires pairing bits by 3-bit binary. It will convey the values from 0 to 7 in decimal. But radix-5 will need 0 to 4 values. So we will get 5, 6, 7 as undetermined values. Hence the addition process will not be possible as well. Thus radix-4 AHSD is best suited for addition that is carry free and fast.

## 5 Simulations

Here the few simulations are described. We performed the simulation using Matlab 7.1 neural network tools in a computer with Pentium IV processor. At first conversion from binary- to- AHSD numbers is done for the manual simulation, then to get self-consistent adjustment (SCA) and adjacent digit modification (ADM) two AHSD numbers is added. Then arithmetic addition of SCA and ADM is done. Now NN design for addition of AHSD is considered. Finally by using Mathlab-7.1 simulation tool simulation is performing for NN design and getting addition result of AHSD numbers. In the figure 4 this workflow procedure is shown.

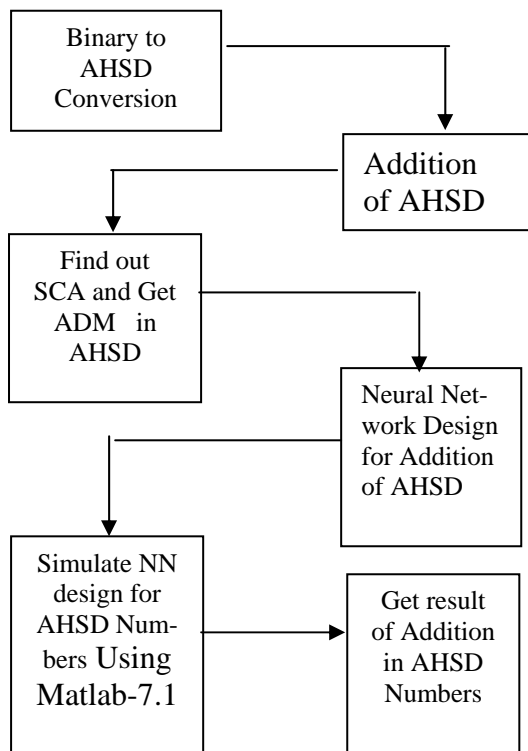


Figure4: The Workflow of simulation procedure for AHSD numbers

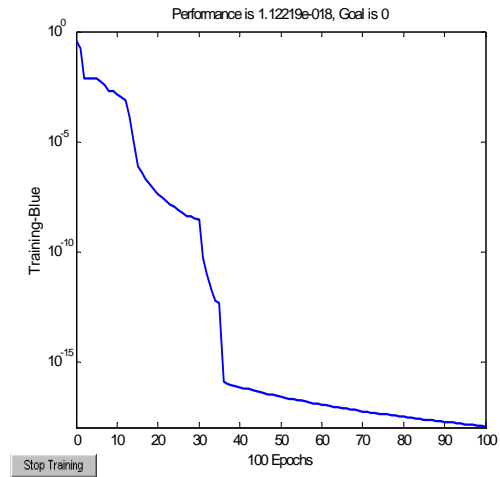


Figure 5 -1:(TF- Logsig) 1-bit adder simulation, and epoch is 100.

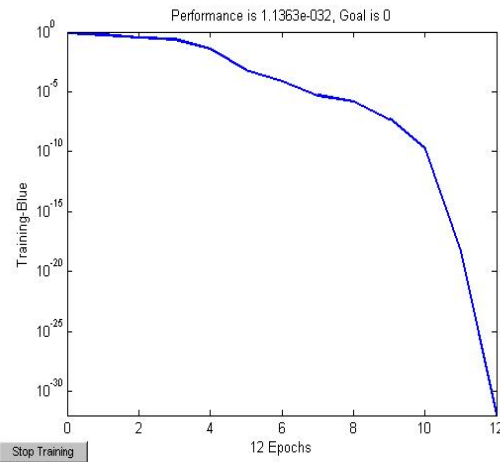


Figure 5-2: (TF-Tansig) 1-bit adder for simulation and epoch is 12.

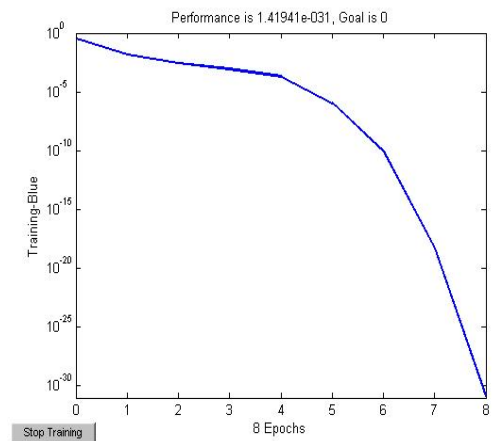


Figure 5-3: (TF-Purelin) 1-bit adder for simulation and epoch is 8.

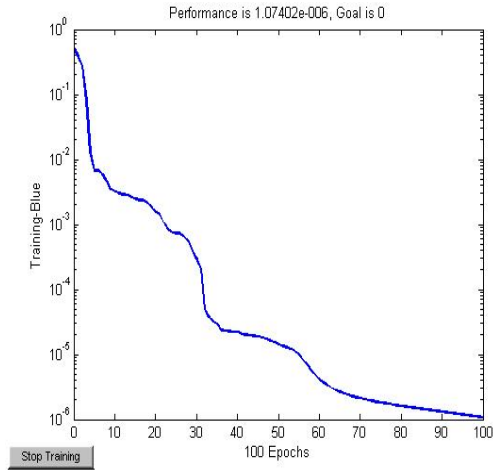


Figure 6-1: (TF-Tansig) 2-bit adder for simulation and epoch is 100 epochs.

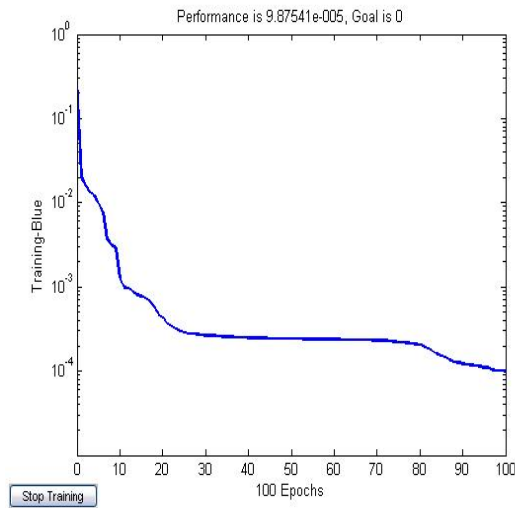


Figure 6-2: (TF-Logsig) 2-bit adder for simulation and epoch is 100 epochs.

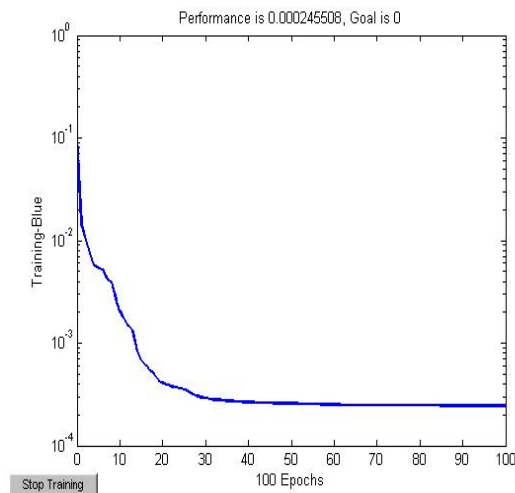


Figure 7: (TF-Logsig) 3-bit adder for simulation and epoch is 100 epochs.

## 6 Experimental Results

The table 1 is summarized here from the experiments done.

Table 1: NN experimental results for AHSD adder design.

| Bit   | Layer | Neuron numbers | Activation Function    |
|-------|-------|----------------|------------------------|
| 1-bit | 2     | I=2/O=2        | Tan-sig/Logsig/Purelin |
| 2-bit | 2     | I=4/O=2        | Tansig/Logsig          |
| 3-bit | 2     | I=6/O=3        | Logsig                 |

Satisfactory performance is found in the design of AHSD-radix-4 adder using NN. This performance shows the better implementation of the adder of redundant number system than radix-2 signed digital adder using NN [5].

## 7 Conclusion

In this paper one CF digital adder for AHSD<sup>(4)</sup> number system based on NN has been proposed. Additionally, if  $r = 2^m$  for any positive integer  $m$ , the interface between AHSD and the binary number system can be realized and it can be easily implement able. To make pairs at the time of conversion from binary to-AHSD, an algorithm has also been proposed. Besides, NN design with Matlab-7.1 simulation tool is used to achieve high speed for addition on AHSD. Even though hardware implementation is not done here but realization shows, the proposed NN design can be flexible. Since both the binary-to-AHSD and AHSD-to-binary converter .CF adder operate in a constant time, we can conclude that the AHSD-to-binary converter dominates the performance of the entire AHSD based on NN design system. The time complexity of the entire AHSD CF adder is  $O(\log_m n)$ . Finally, it is appealed that additional studies is necessary including not only the implementation of CF adder with different design skills, but also the applications of various CF adders to design fast arithmetic. Hopefully, in future this paper can be extended for the digital adder of AHSD number system based on NN and which can be realized by using hardware, using current mode CMOS or voltage mode CMOS [6] and using SPICE simulation tool for hardware realization.

## References

- [1] S.H. Sheih, and C.W. Wu, "Asymmetric high-radix signed-digit number systems for carry-free addition," *Journal of information science and engineering* 19, 2003, pp.1015-1039.
- [2] B. Parhami, "Generalized signed-digit number systems: a unifying framework for redundant number representations," *IEEE Transactions on Computers*, vol. 39, 1990, pp.89- 98.
- [3] S.H. Sheih, and C.W. Wu, "Carry-free adder design using asymmetric high-radix signed-digit number system," in the *Proceedings of 11<sup>th</sup> VLSI Design/CAD Symposium*, 2000, pp. 183-186.
- [4] M. Sakamoto, D. Hamano and M. Morisue, "A study of a radix-2 signed-digit al fuzzy processor using the logic oriented neural networks," *IEEE International Systems Conference Proceedings*, 1999, pp. 304-308.
- [5] T. Kamio, H. Fujisaka, and M. Morisue, "Back propagation algorithm for logic oriented neural networks with quantized weights and multilevel threshold neurons," *IEICE Trans. Fundamentals*, vol. E84-A, no.3, 2001.
- [6] A. Moushizuki, and T. Hanyu, "Low-power multi-ple-valued current-mode logic using substrate bias control," *IEICE Trans. Electron.*, vol. E87-C, no. 4, pp. 582-588, 2004 .