http://ansinet.com/itj

ITJ

ISSN 1812-5638

# INFORMATION TECHNOLOGY JOURNAL

ANSInet
Asian Network for Scientific Information
308 Lasani Town, Sargodha Road, Faisalabad - Pakistan

Information Technology Journal 7 (8): 1156-1162, 2008
ISSN 1812-5638


# Direct Exponential Function Computer in Neural Network Application Based on Evolvable Hardware

Z. Huajun and Z. Jin
Department of Control Science and Engineering, Huazhong University of Science and Technology, China

**Abstract:** This study proposes a new direct exponential function computer based on CORDIC algorithm to solve the neuron realization problem on the Evolvable Hardware (EHW). The direct exponential function computer combines the essential character of exponential function with CORDIC algorithm and turns aside the conventional approach which computes the exponential function by adding sinh (x) and cosh (x) indirectly. Because the direct computer executes shift-add operation in allusion to exponential function directly and get rid of the computation of sinh (x) and cosh (x), it not only calculates much faster than the indirect computer, but also saves much more logic gates. In addition, through strict and integrated analysis, the direct computer is proved to own perfect convergence. It converges to the real value with arbitrary precision and operates shift-add operation without any unnecessary rotation. Since of the direct efficient shift-add operation, the direct computer is useful for the application of neural network based on the EHW.

**Key words:** CORDIC, activation function, neurone, field programmable gate arrays

## INTRODUCTION

Evolvable hardware is a new concept in the development of online adaptive machines. In contrast to conventional hardware where the structure is irreversibly fixed in the design process, EHW is designed to adapt to changes in task requirements or changes in the environment through its ability to reconfigure its own hardware structure online (dynamically) and autonomously.

This study is supported by the National Natural Science Foundation of China (No. 60874047) and Hubei Science Foundation of China (No. 2007ABA281) combine the EHW with artificial neural network to design complex configuration which adaptive to changes of the environment (Gallagher *et al*., 2005; Vigraham and John, 2005, 2006; Wang Piao *et al*., 2007; Rocke *et al*., 2005; Saumil *et al*., 2006; Glette *et al*., 2007; Murakawa *et al*., 1999; De *et al*., 2000, 2001). The CAM-Brain Machine (CBM) (De *et al*., 2000, 2001) is the most famous one of the applications which combines EHW with neural network. It is an FPGA based tool for evolving a 75 million neuron artificial brain to control a lifesized kitten robot. So, many neurons require large-scale logic gates to realize the function of neural networks. Obviously, the realization of the activation function of neuron is the key factor which decides the scale of logic gates. Usually the activation function is a nonlinear function such as sigmoid function (Yongquan *et al*., 2006). Direct implementations of nonlinear activation functions can be expensive. In order to simplify the realization of neural network, many implementations used hard limiters or saturated linear activation function to avoid building the sigmoid activation function or hyperbolic tangent function which provide smooth transitions between excitation and inhibition thereby improving neural response (Jiang *et al*., 2005; Rocke *et al*., 2007; De, 2006; Upegui *et al*., 2005). However, different problem will need neural network with different activation function and structure. If the sigmoid activation function or hyperbolic tangent function is necessary for the problem, it will complicate the realization of neural networks. In the piecewise-linear approximation of the non-linear activation function is used to realize sigmoid function (Porrmann, 2002). Although, the piecewise-linear approximation realizes sigmoid function successfully, it needs several different circuits to approximate the entire sigmoid function. Usually, each individual circuit contains multiplier which takes up large scale logic gates. The more precise, the more multiplier needed. In the sigmoid activation function is calculated by the lookup tables (LUTs) method which pre-stores the output values of the activation function in a lookup table (Krips *et al*., 2002). The choice of precision of data used would also be driven by the size of the activation function lookup table. Higher precision would require a bigger table. So far, the most

**Corresponding Author:** Zhang Huajun, Department of Control Science and Engineering, Huazhong University of Science and Technology, China

efficient approach to realize sigmoid function is the Coordinate Rotation Digital Computer (CORDIC) which can compute a wide range of functions including certain trigonometric, hyperbolic, linear and logarithmic functions through only shift-and-add operation. In the sigmoid function is realized based on calculating exponential function as $e^x = \sinh(x)+\cosh(x)$ and the hyperbolic sine and cosine function are calculated by CORDIC algorithm (Liddicoat, 2006; Chen *et al*., 2007; Meng, 2006). Although sigmoid function can be calculated by CORDIC algorithm indirectly, there are two defects during the calculation. First, the indirect calculation of exponential function $e^x$ prolong the computing time. For instance, if $e^x = e^{1.39}$ ($e^{1.39} = 4$), the direct CORDIC algorithm will only shift the $e^{0.693}$ ($e^{0.693} = 2$) left with 1 bit. However, the indirect calculation of $e^x = e^{1.39}$ will rotate many shift-add steps by calculating sinh (x) and cosh (x). Compared with the direct CORDIC algorithm, the indirect calculation is inefficient. Second, there is a multiplicative operation in each iteration during the calculating of sinh (x) and cosh (x), it not only prolongs the computing time, but also takes up much more logic gates. Obviously, the defects of individual neuron realization do not attract people's attention. But from the macrostructure, if the neural networks include 75 million neurons as the CBM, the indirect calculation of exponential function will expend a great deal of supererogatory logic gates than direct CORDIC algorithm. In real world application, every EHW chip has finite number of logic gate. The more complex of the problem, the more logic gate needed. With the inefficient realization of activation function, it will be difficult to realize the neural networks even if it is possible in theory.

In order to solve complex problem with simple neural network, it is necessary to use efficient CORDIC algorithm to realize nonlinear activation function. This study proposes a direct CORDIC algorithm which calculates sigmoid function efficiently. It takes advantages of exponential function and CORDIC to calculate exponential function only by shift-add operation. It uses $\ln 2^i$ or $\ln 2^{-i}$ as the rotation angle in each step. Without unnecessary calculation of sin h(x) and cos h(x), the scale of logic gate used to realized neuron is reduced significantly. Further more, the computing time of sigmoid function is also reduced distinctly.

## DIRECT CORDIC IN EXPONENTIAL FUNCTION

**Direct CORDIC:** The symmetric sigmoid function is the most common activation function used in neural network. Its expression written as follow:

$$f(x) = \frac{1-e^{-x}}{1+e^{x}} \quad (1)$$

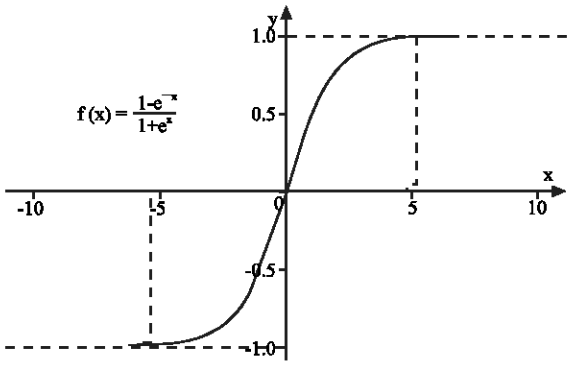


Fig. 1: Curve of sigmoid function

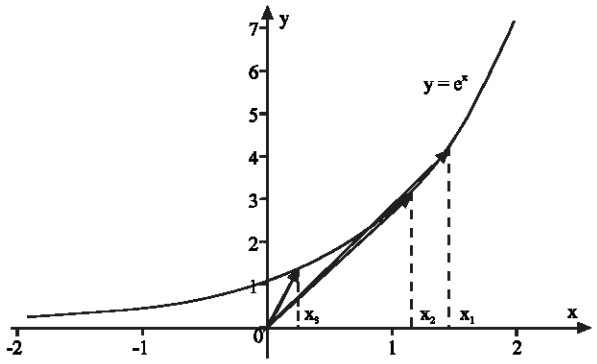


Fig. 2: Character of exponential function

The curve of sigmoid function is shown as Fig. 1. Shown by the curve, if $x \in [-5.3\ \ 5.3]$, sigmoid function has nonlinear character. If $x>5.3$ and $x<-5.3$, sigmoid function keeps 1 or -1 all the way. Because the curve possesses odd symmetric character, it is only need to computer sigmoid function with positive or negative x.

Clearly, the difficulty of sigmoid function realization is to compute the exponential function. Turning aside the relation between exponential function and hyperbolic sine/cosine function, it is necessary to analyze the essential characteristic of exponential function and CORDIC together. For the essential characteristic of exponential function, suppose $x_1$, $x_2$ and $x_3$ are arbitrary positive value and $x_1 = x_2 + x_3$. Figure 2 shows the relation of $e^{x1}$, $e^{x2}$ and $e^{x3}$. If $e^{x2} = y_2$ and $e^{x3} = y_3$, then the $e^{x1}$ can be computed as:

$$e^{x1} = e^{x2} \times e^{x3} = y_2 \times y_3 \quad (2)$$

However, $x_2$ and $x_3$ are arbitrary positive value, it is impossible to get $y_2$ and $y_3$ beforehand. At this time, it is necessary to find the $e^{x2}$ and $e^{x3}$ based on the CORDIC.

According to CORDIC principle, it is a bit-recursive implementation of the forward and backward Givens

rotations (Jack, 2000; Walther, 2000). The basic CORDIC iteration equations at the ith step are (Walther, 2000):

$$\begin{cases} X_{i+1} = K_i(X_i - m\sigma_i 2^{-s(m,i)} Y_i) \\ Y_{i+1} = K_i(Y_i + \sigma_i 2^{-s(m,i)} X_i) \\ Z_{i+1} = Z_i - \sigma_i \theta_{m,i} \end{cases} \quad (3)$$

where, the iterations are performed for arbitrary i (i = 0, 1, 2,…, N-1). The coordinate parameter m identifies the coordinate system type (namely, circular, linear and hyperbolic coordinates for m equal to 1, 0 and -1, respectively). The rotation direction $\sigma_i$ for rotation is $\sigma_i$ = -sign($z_i$), while, for vectoring, it is $\sigma_i$ = -sign($X_i$, $Y_i$). For m = 1, the shift sequence S(1, i) is defined by (S(1, i) = 0, 1, 2, 3, 4, 5,….). The rotation angle $\theta_{m,i}$ is given by $\theta_{mi}$ = arctan $2^{-s(1,i)}$ = arctan $2^{-i}$. For each rotation performed, there is a scale factor $K_i$ which corrects the amplification introduced by the linearized rotation in the x and y coordinates. With the above assumptions, $K_i = 1/\sqrt{1+\sigma_i^2 2^{-2i}}$ for the ith iteration. After all iterations, the total scale factor is $K' = \prod_{i=0}^{N-1} K_i = \prod_{i=0}^{N-1} 1/\sqrt{1+\sigma_i^2 2^{-2i}}$. With different coordinate parameters, the COORDIC can calculate different type function. Such as Chen *et al.* (2007), it selects m = -1 and computes exponential function as $e^x$ = sinh (x)+cosh (x). It is a indirect application of CORDIC algorithm and does not apperceive the essence of CORDIC. In Eq. 3, the essential principle of CORDIC is to establish a iteration as:

$$\begin{cases} f(x) = g(f(z_i), 2^i) \\ z_i = x \pm v(2^i) \end{cases} \quad (4)$$

where, g(•) is the mapping function of $2^i$ and target function value, v(•) is the rotation angle which is relative to $2^i$ and f(x). As Eq. 2 shown, in the computation of exponential function, the g(•) can be written as: $g(x) = e^{x_2} \times e^{x_3}$. If $e^{x_2}$ and $e^{x_3}$ are expressed as a product of $2^i$, the $e^{x_1}$ will be computed only by shift-add operation. In order to use $2^i$ to express $e^{x_2}$ and $e^{x_3}$, it is necessary to divide $e^x$ into two parts since of $e^x \in (1, \infty)$ and $2^i \in (0, 1) \cup [2, +\infty]$. When $e^x \in (1\ \ 2)$, it is easy to get $e^x - 1 \in (0\ \ 1)$. If, $e^x - 1 = 2^{-i}$ (i<0) then $x = \ln(2^{-i}+1)$. When $e^x \in [2\ \ +\infty]$, if $ex = 2^i$, then $x = \ln(2^i)$. Table 1 gives out $\ln(2^i)$ and $\ln(1+2^i)$ with $i \in [1\ \ 12]$.

Based on the above analysis, it is easy to computer arbitrary $e^x$ by subtracting a series of $\ln(2^i)$ and $\ln(1+2^{-i})$ from x. According to standard CORDIC principle, the expression of direct CORDIC in the computation of $e^x$ can be synthesized as:

$$\begin{cases} e_n^x = e_{n-1}^x \times E \\ z_n = z_{n-1} - z' \\ E = 2^i, z' = \ln(2^i) \ \text{ if } \ z_{n-1} \geq 0.6931 \\ E = 1 + 2^{-i}, z' = \ln(1+2^{-i}) \ \text{ if } \ z_{n-1} < 0.6931 \end{cases} \quad (5)$$

After some iterative shift-add operation, if $z_n = 0$, the computation will stop and give out the $e^x$. For instance, if x = 5.3, because 5.3>4.852, then $e^{5.3} = e^{4.852} \times e^{0.448}$, $z_2$ = 5.3−4.852 = 0.448. Comparing z with a series of $\ln(1+2^{-i})$, it is easy to get that $e^{0.448} = e^{0.4055} \times e^{0.0308} \times e^{0.0078} \times e^{0.0039}$. Then the integrated computation of $e^{5.3}$ is $e^{5.3} = e^{4.852} \times e^{0.4055} \times e^{0.0308} \times e^{0.0078} \times e^{0.0039} = 2^7 \times (1+2^{-1}) \times (1+2^{-5}) \times (1+2^{-7}) \times (1+2^{-8})$.

Clearly, without any multiplicative operation, it is only need five shift-add operations to computer the $e^{5.3}$. In order to interpret direct CORDIC distinctly, Fig. 3 shows the flow of direct CORDIC in the computation of $e^x$.

**Convergence of direct CORDIC:** In standard CORDIC algorithm, the rotation of different angle does not assure convergence of the computation at all time (Lang and Antelo, 1998). Thus, it is necessary to analyze the convergence of the direct CORDIC in computation of exponential function. Suppose x is an arbitrary positive vale. The analysis, it is also necessary to divide x into two parts to analyze the convergence.

At first, if $x_i$<0.6931, the $e^{x_1} \in (1\ \ 2)$ and $e^{x_1} - 1 \in (0\ \ 1)$. According to computer science theory (Cowlishaw *et al.*, 2001), $e^x - 1$ can be expressed by a binary code easily. With different precision, the binary code is different. For instance, if we use 20 bits to express an arbitrary positive value, the $e^{x_1}$ can be expressed as the sum of 1 and a decimal fraction as follow:

Table 1: The $\ln(2^i)$ and $\ln(1+2^{-i})$

| i | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| $\ln(2^i)$ | 0.6931 | 1.3863 | 2.0794 | 2.7726 | 3.4657 | 4.1589 | 4.8520 | 5.5452 | 6.2383 | 6.9315 |
| $\ln(1+2^{-i})$ | 0.4055 | 0.2231 | 0.1178 | 0.0606 | 0.0308 | 0.0155 | 0.0078 | 0.0039 | 0.0020 | 0.0010 |
| i | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| $\ln(2^i)$ | 0.0004 | 0.0002 | 0.0001 | $6\times10^{-5}$ | $3\times10^{-5}$ | $1\times10^{-5}$ | $7\times10^{-6}$ | $3\times10^{-6}$ | $1\times10^{-6}$ | $9\times10^{-7}$ |
| $\ln(1+2^{-i})$ | 7.6246 | 8.3177 | 9.0109 | 9.7041 | 10.3972 | 11.0904 | 11.7835 | 12.4766 | 13.1697 | 13.8629 |

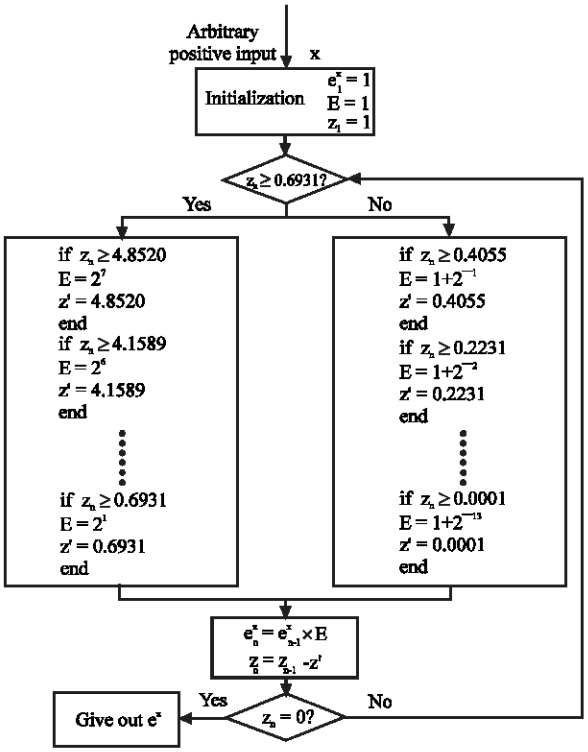


Fig. 3: The flow of direct exponential function computer

$$e^{x_1} = 1 + d_1 = 1 + \sum_{i=1}^{20} \lambda_i 2^{-i} \tag{6}$$

where, $d_1$ is the decimal fraction and $d_1 \in (0\ 1)$. $\lambda_i$ is weight of each $2^{-i}$ and $\lambda_i = 1$ or 0. Suppose, $e^{x_2} = 1 + 2^{-m}$, $\lambda_m = 1$ and $2^{-m}$ is bigger than other $\lambda_i 2^{-i}$ in Eq. 6. Then Eq. 6 can be transformed into:

$$e^{x_1} = 1 + \sum_{i=1}^{20} \lambda_i 2^{-i} = 1 + 2^{-m} + \sum_{i=1, i\neq m}^{20} \lambda_i 2^{-i} = (1 + 2^{-m}) \times (1 + \frac{\sum_{i=1, i\neq m}^{20} \lambda_i 2^{-i}}{1 + 2^{-m}}) = e^{x_2} \times (1 + d_2) \tag{7}$$

Since of $d_2 < 1$, it is equal to $d_2 = \sum_{i=1}^{20} \lambda_i 2^{-i}$. At this time, suppose, $e^{x_3} = 1 + 2^{-n}$, $\lambda_n = 1$ and $2^{-n}$ is bigger than other $\lambda_i 2^{-i}$. Then Eq. 7 can be transformed into:

$$e^{x_1} = e^{x_2} \times (1 + d_2) = e^{x_2} (1 + \sum_{i=1}^{20} \lambda_i 2^{-i}) = e^{x_2} (1 + 2^{-n} + \sum_{i=1, i\neq n}^{20} \lambda_i 2^{-i}) = e^{x_2} (1 + 2^{-n}) (1 + \frac{\sum_{i=1, i\neq n}^{20} \lambda_i 2^{-i}}{1 + 2^{-n}}) = e^{x_2} \times e^{x_3} \times (1 + d_3) \tag{8}$$

Obviously, if $d_3 > \frac{1}{2^{20}} = 9.5 \times 10^{-7}$, the Eq. 8 also can be written as:

$$e^{x_1} = e^{x_2} \times e^{x_3} \times e^{x_4} \times (1 + d_4) \tag{9}$$

where, $e^{x_4} = 1 + 2^{-\beta}$ and $d_4 > d_3 > 1$. Analogous, if the decimal fraction $d_i > 9.5 \times 10^{-7}$, the transformation of Eq. 9 will go on. Because of $d_i < d_{i-1} < ... < d_2 < d_1 < 1$, the limit of $d_i$ is 0. When $d_i > 9.5 \times 10^{-7}$, it is impossible to use $2^{-i}$ express it and $1 + d_i = 1 + \sum_{i=1}^{20} 0 \times 2^{-i} = 1 = e^0$. Thus, the final expression of $e^{x_1}$ is:

$$e^{x_1} = e^{x_2} \times e^{x_3} \times e^{x_4} \times \cdots \times e^{x_r} \times e^0 = (1 + 2^{-m}) \times (1 + 2^{-n}) \times (1 + 2^{-\beta}) \times \cdots \times (1 + 2^{-r}) \tag{10}$$

where, $x_i = \ln(1+2^{-i})$. By shift-add operations, the direct CORDIC converges at $e^{x_1}$ accurately. The longer of the binary code which used to express decimal fraction, the more accurately direct CORDIC converges at $e^{x_1}$. Based on the above proof, the direct CORDIC converges at real $e^x$ obviously.

Second, if $x_1 \geq 0.6931$, it is necessary to compare $x_1$ with a series of $\ln(2^i)$. Since of the x-axis is divided into some seriate districts such as $(\ln(2^i)\ \ln(2^{i+1}))$ $(i \geq 0)$.

Consequently, $x_1$ should belong to one of the districts by the comparison. Then $e^{x_1}$ can be expressed as

$$e^{x_1} = e^{\ln(2^i)} \times e^{x_1 - \ln(2^i)} = 2^i \times e^{x_1 - \ln(2^i)} \tag{11}$$

After shift operation in Eq. 11, it is need to computer $e^{x_1 - \ln(2^i)}$. Because of $\ln(2^{i+1}) - \ln(2^i) = \ln(2) = 0.6931$, the range of each district is 0.6931. So, $x_1 - \ln(2^i) - \ln(2^{i+1}) - \ln(2^i) = 0.6931$. Suppose $x_2 = x_1 - \ln(2^i) < 0.6931$, according to Eq. 11, the final computation of $e^{x_1}$ is:

$$e^{x_1} = 2^i \times e^{x_2} = 2^i \times (1 + 2^{-m}) \times (1 + 2^{-n}) \times (1 + 2^{-\beta}) \times \cdots \times (1 + 2^{-r}) \tag{12}$$

Of course, by shift-add operations in Eq. 12, the direct CORDIC also converges at $e^{x_1}$ accurately.

By analyzing the convergence with $0 < x < 0.6931$ and $x \geq 0.6931$, respectively, the direct CORDIC is proved convergent at all time. Without any vibration around the $e^x$, it approaches to $e^x$ step by step.

## CIRCUIT DESIGN AND SIMULATION OF DIRECT CORDIC

In order to design an efficient direct CORDIC circuit, it is not only need to pay attention to convergence and compute speed, but also the logic gates scale. Based on the analysis of the section II, the length of binary code used to express decimal fraction of $\ln(2^i)$ and $\ln(1+2^{-i})$ decides the precision of direct CORDIC. The more precise of computation, the more logic gate needed. So, it is

Table 2: Precision and error of direct CORDIC

| n | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| s | 0.5000 | 0.7500 | 0.8750 | 0.9375 | 0.96875 | 0.984375 | 0.9921875 | 0.99609375 |
| a | 129.9627 | 56.9007 | 26.6745 | 12.9206 | 6.35940 | 3.154800 | 1.5713 | 0.7841 |
| n | 9 | 10 | 11 | 12 | 13 | 14 | 15 | |
| s | 0.998046875 | 0.9990234375 | 0.99951171875 | 0.999755859375 | 0.9998779296875 | 0.999755859375 | 0.9998779296875 | |
| a | 0.3917 | 0.1957 | 0.0978 | 0.0489 | 0.0245 | 0.0122 | 0.0061 | |

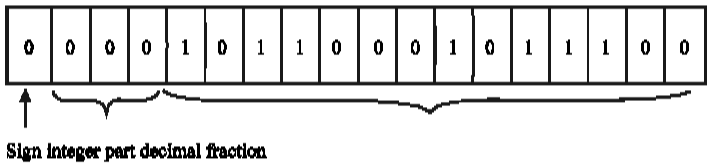


Fig. 4: The data format of $\ln(2^i)$ and $\ln(1+2^{-i})$

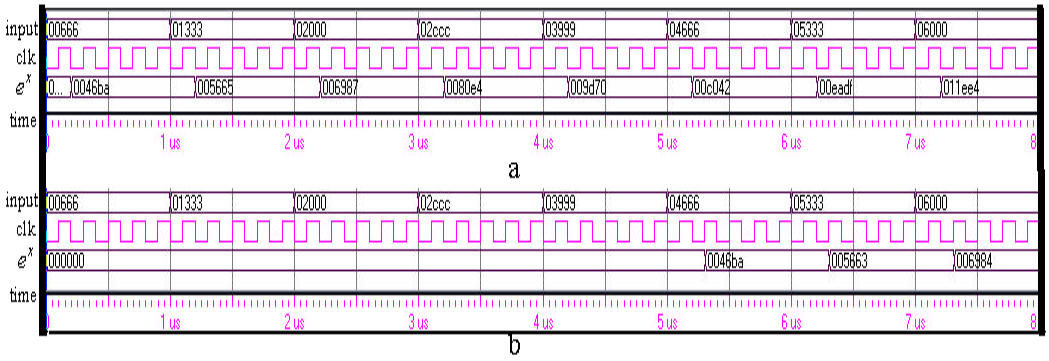


Fig. 5: The simulation result of (a) direct and (b) indirect computer

important to design the data format for the series of $\ln(2^i)$ and $\ln(1+2^{-i})$. If the decimal fraction is expressed by n bits, then the code precision is $1/2^n$. Suppose $s = \sum_{i=1}^{n} 2^{-i}$ is the biggest decimal value expressed by binary code. No matter what n is selected, s is smaller than 1. According to Eq. 6 and 12, the error between direct CORDIC output and real $e^x$ is $2_i \times (e^{1-s} - 1)$ (i>0). With different precision, the error appears distinct difference. Since of $x \in (0\ \ 5.3)$ in the computation of sigmoid function, the biggest error brought out by different precision is $a = 2^7 \times (e^{1-s} - 1)$. Table 2 shows the s and a. Because high precision need more logic gates to construct CORDIC circuit, it is necessary to consider precision and logic scale together. When error is 0.01, the computation result satisfies the requirement of engineering application. From Table 2, the reasonable length of binary code is n = 14. Because $x \in (0\ \ 5.3)$ in the computation of sigmoid function, the integer part of x can be expressed by Eq. 3 bits binary code. With the sign bit, the shortest reasonable length of binary code is 18. For instance, if x = 0.6931, the binary code is x = 000010110001011100h, the data format is shown by Fig. 4. With the reasonable data format, the direct CORDIC is a real efficient exponential function computer.

This study uses verilog language to describe the direct CORDIC circuit and selects Xinlinx Sparton3E500 as the platform to simulate the function of direct CORDIC. Based on the above analysis, the program is constituted according to Fig. 3. Because the neuron computers at each biologic neural impulses (Schreiner, 2001), then the exponential function computer is triggered by clock pulse. Through different clock period testing, the shortest clock period is 200 ns. The function simulation result of exponential function computer is shown as Fig. 5a. In Fig. 5a, it is clearly that the exponential function computer calculates exponential value during the positive pulse period. Then the computation time of direct exponential function computer is 100 ns. In order to compare with other approach, this study also simulates the function of indirect CORDIC according to (Chen *et al*., 2007). With the same clock period and data format, the simulation result is shown as Fig. 5b. Because of indirect computation, the computation time continues 5 us.

Table 3: Logic gate scale in exponential function computer

| | Slice flip flop | 4 input LUT | Logic slice | LUT used as route-thru |
|---|---|---|---|---|
| Indirect CORDIC | 2140 | 2071 | 1112 | 81 |
| Direct CORDIC | 23 | 1296 | 658 | 3 |

Table 4: The output of exponential function computer

| | 0.1 | 0.3 | 0.5 | 0.7 | 0.9 | 1.1 | 1.3 | 1.5 | 1.7 | 1.9 | 2.1 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Direct CORDIC (hex) | 46ba | 5665 | 6987 | 80e4 | 9d70 | c042 | eadf | 11ee4 | 15e6d | 1abf2 | 20aba |
| Indirect CORDIC (hex) | 46ba | 5663 | 6984 | 80e3 | 9d71 | c042 | eade | 11ee5 | 15e6f | 1abf0 | 20abc |
| Real value (hex) | 46bb | 5664 | 6984 | 80e1 | 9d6a | c044 | ead5 | 11ed3 | 15e55 | 1abe5 | 20aa2 |
| | 2.3 | 2.5 | 2.7 | 2.9 | 3.1 | 3.3 | 3.5 | 3.7 | 3.9 | 4.1 | 4.3 |
| Direct CORDIC (hex) | 27e74 | 30bc5 | 3b878 | 48b1f | 58ce0 | 6c762 | 8477b | a1cd3 | c5a6e | f160e | 126d23 |
| Indirect CORDIC (hex) | 27e78 | 30bcc | 3b879 | 48b1a | 58ce4 | 6c760 | 8477b | a1ce1 | c5a70 | f1613 | 126d22 |
| Real value (hex) | 27e59 | 30bad | 3b84d | 48b25 | 58cab | 6c735 | 84763 | a1ca0 | c59c1 | f15c7 | 126cc9 |

Obviously, the direct exponential function computer calculates much faster than the indirect computer. With regard to logic gate scale in realization, the number of slice and look up table (LUT) used in the exponential function computer is shown as Table 3. From the data of Table 3 the direct exponential function computer saves much more logic gate than indirect computer.

Beside the computation time and logic scale, the computation precision is shown as Table 4. In order to test the smoothness of exponential function computer in different district, the input data is distributed uniformly over $x \in (0 \quad 5.3)$. As Table 4 shows, the precision of two approaches is equivalent.

## CONCLUSION

This study proposes a new direct CORDIC to compute arbitrary $e^x$ value in EHW application. By comparing with indirect CORDIC approach, the direct computer not only calculates much faster than the indirect computer, but also saves much more logic gates. The improved computer is useful for real world application. Regard to logic gate scale, the direct computer increases the probability to solve complex problem since of the researchers can design more neurons under the restrict of chip resource. As for the computation time in many fields, the faster of computation, the better of performance. It is useful for field such as control system, signal processing field and computer field.

## ACKNOWLEDGMENT

Thanks for the supports of the National Natural Science Foundation of China (No. 60874047) and Hubei Natural Science Foundation of China (No. 2007ABA281).

## REFERENCES

Chen, X., G. Wang and K. Liu, 2007. Implementation of activated function of neural networks by using hybrid CORDIC. J. Huazhong Univ. Sci. Technol. (Natural Sci. Edn)., 35: 114-117.

Cowlishaw, M.F., E.M. Schwarz, R.M. Smith and C.F. Webb, 2001. A decimal floating-point specification. Proceedings of the 15th IEEE Symposium on Computer Arithmetic, June 11-13, Vail, Co. USA., pp: 147-154.

De, G.H., M. Korkin, P. Guttikonda and D. Cooley, 2000. Evolving detectors of 2D patterns on a simulated CAM-Brain Machine, an evolvable hardware tool for building a 75 million neuron artificial brain. Proceeding of Critical Technologies for the Future of Computing, July 31, San Diego, USA., pp: 13-13.

De, G.H., A. Buller, P.L. De, M. Korkin, G. Fehr and T. Chodakowski, 2001. Initial evolvability experiments on the CAM-Brain machines (CBMs). Proceeding of Congress on Evolutionary Computation, May 27-30, Seoul, South Korea, pp: 635-642.

De, G.H., 2006. Hardware accelerators for evolving building block modules for artificial brains. Proceeding of the 1st NASA/ESA Conference on Adaptive Hardware and Systems, Jun 15-18, Piscataway, NJ 08855-1331, USA., pp: 222-224.

Gallagher, J.C., K.B. Sanjay and S. Vigraham, 2005. A reconfigurable continuous time recurrent neural network for evolvable hardware applications. Proceedings of 2005 NASA/DoD Conference on Evolvable Hardware, 29 June-July 1, Washington, DC. USA., pp: 247-250.

Glette, K., J. Torresen and M. Yasunaga, 2007. An online EHW pattern recognition system applied to sonar spectrum classification. Proceeding of the 7th International Conference on Evolvable Systems: From Biology to Hardware, 2007 Springer Verlag, Heidelberg, pp: 1-12.

Jack, V.E., 2000. The birth of cordic. J. VLSI Signal Proc., 25: 101-105.

Jiang, W., S.G. Kong and G.D. Peterson, 2005. ECG signal classification using block-based neural networks. Proceeding of the International Joint Conference on Neural Networks, 31July-August 4, Montreal, QC., Canada, pp: 326-331.

Krips, M., T. Lammert and A. Kummert, 2002. FPGA implementation of a neural network for a real-time hand tracking system. Proceedings of the 1st IEEE International Workshop on Electronic Design, Test and Applications, January 29-31, Christchurch, New Zealand, pp: 313-317.

Lang, T. and E. Antelo, 1998. CORDIC vectoring with arbitrary target value. IEEE Trans. Comput., 47: 736-749.

Liddicoat, A.A., L.A. Slivovsky, T. McLenegan and D. Heyer, 2006. FPGA-based artificial neural network using CORDIC modules. Proceedings of SPIE-The International Society for Optical Engineering, August 15, San Diego, CA., USA., pp: 63130-63130.

Meng, Q., 2006. Application of CORDIC algorithm to neural networks VLSI design. Proceeding of the Multiconference on Computational Engineering in Systems Applications, October 4-6, Beijing, pp: 504-508.

Murakawa, M., S. Yoshizawa and I. Kajitani, 1999. GRD chip: Genetic reconfiguration of DSPs for neural network processing. IEEE Trans. Comput., 48: 628-639.

Porrmann, M., U. Witkowski, H. Kalte and U. Ruckert, 2002. Implementation of artificial neural networks on a reconfigurable hardware accelerator. Proceedings of the 10th Euromicro Parallel, Distributed and Network-based Processing, 2002 Los Alamitos, CA., USA., pp: 243-250.

Rocke, P., J. Maher and F. Morgan, 2005. Platform for intrinsic evolution of analogue neural networks. Proceeding of International Conference on Reconfigurable Computing and FPGAs, September 28-30, Piscataway, USA., pp: 8-8.

Rocke, P., B. McGinley, F. Morgan and J. Maher, 2007. Reconfigurable hardware evolution platform for a spiking neural network robotics controller. Proceeding of the 3rd International Workshop on Applied Reconfigurable Computing, 2007 Springer Verlag, Heidelberg, pp: 373-378.

Saumil, M., D. Peterson Gregory and K. Park Sang, 2006. FPGA implementation of evolvable block-based neural networks. Proceedings of IEEE Congress on Evolutionary Computation, 2006 Vancouver, BC., Canada, pp: 3129-3136.

Schreiner, K., 2001. Neuron function: The mystery persists. IEEE Intelligent Syst., 16: 4-7.

Upegui, A., C.A. Pena-Reyes and E. Sanchez, 2005. An FPGA platform for on-line topology exploration of spiking neural networks. Microprocessors Microsyst., 29: 211-223.

Vigraham, S.A. and C.G. John, 2005. A case for using minipop as the evolutionary engine in a CTRNN-EH control device: An analysis of area requirements and search efficacy. Proceedings of NASA/DoD Conference on Evolvable Hardware, 29 June-July 1, Washington, DC, USA., pp: 221-228.

Vigraham, S.A. and C.G. John, 2006. CTRNN-EH in silicon: Challenges in realizing configurable CTRNNs in VLSI. IEEE Congress on Evolutionary Computation, 2006 Vancouver, BC., Canada, pp: 2807-2813.

Walther, J.S., 2000. Story of unified cordic. J. VLSI Signal Proc., 25: 107-112.

Wang Piao, J., H. Chang and H. Lee Chong, 2007. FPGA implementation of evolvable characters recognizer with self-adaptive mutation rates. Proceedings of the 8th International Conference on Adaptive and Natural Computing Algorithms, 2007 Warsaw, Poland, pp: 286-295.

Yongquan, Z., L. Daozhu and Y. Yindong, 2006. Functional network and tunable activation function neural network. Proceeding of 6th World Congress on Intelligent Control and Automation, 2006 Dalian, pp: 2757-2762.