Al-khwarizmi Engineering Journal

Al-Khwarizmi Engineering Journal, Vol.3, No.2, pp1-7 (2007)

Reduction of the error in the hardware neural network

Dr. Dhafer r. Zaghar
Computer and programming Engineering Department
College of EngineeringAL-Mustanserya
University / Baghdad/ Iraq

(Received 7 March 2006; accepted 24 April 2007)

Abstract:-

Specialized hardware implementations of Artificial Neural Networks (ANNs) can offer faster execution than general-purpose microprocessors by taking advantage of reusable modules, parallel processes and specialized computational components. Modern high-density Field Programmable Gate Arrays (FPGAs) offer the required flexibility and fast design-to-implementation time with the possibility of exploiting highly parallel computations like those required by ANNs in hardware. The bounded width of the data in FPGA ANNs will add an additional error to the result of the output. This paper derives the equations of the additional error value that generate from bounded width of the data and proposed a method to reduce the effect of the error to give an optimal result in the output with a low cost.

Key Words: Neural, co-processor, DSP, FPGA, ISE 4.1i software, adder, multiplier.

1.Introduction

Artificial neural networks have for many years proved themselves effective tools in classification and even prediction of arrays of patterns. Today, artificial neural networks are used in diverse applications, ranging from voice recognition to weather prediction models [1,2]. As industry has developed the need for rapid prototyping and denser devices, FPGA have become excellent tools for the implementation of complex digital systems [3,4]. By taking advantage of the dynamic flexibility that FPGAs have to offer, a digital implementation of a general-purpose ANN in the form of a co-processor is created and presented in this paper. This ANN co-processor has been designed to work in conjunction with a DSP or general-purpose processor to create a fast and flexible intelligent ANN system with the potential to be used in a wide range of applications.

By realizing specific hardware entities commonly used in ANNs like multiplication and addition and exploiting any parallelism found in the net, the hardware implementation of an ANN can very well outperform its software counterpart when comparing the throughput of both methods. One, nevertheless, has to keep in mind that the use of FPGAs implies the sacrifice of flexibility for speed. While the FPGA we use has a relatively large quantity of resources, we make a conscious effort to improve speed while keeping the gate count to a minimum. For this reason, each layer in the ANN is composed of parallel neurons that individually compute a summation of products, and all the internal components of the network are designed to take advantage of the

available hardware resources like fast-carry chains, lookup tables and distributed RAM blocks [5].

2. Hidden and Output Layer Design

The target device for the co-processor design is Xilinx's Spartan II FPGA. This device is chosen because of its distributed memory RAM banks which allow implementation of fast register files where inputs and weights are stored, its versatile I/O interface that supports 3.3V outputs and TTL voltage levels and its low market cost. The complete co-processor design is written in VHDL (**V**HSIC **H**ardware **D**escription **L**anguage and the acronym **V**HSIC, in turn, refers to the **V**ery **H**igh-**S**peed **I**ntegrated **C**ircuit program) using a structural approach with the help of Xilinx Foundation Series ISE 4.1i software [6].

The hardware implementation of the neurons realizes the following two equations:

$$a_j^s = \sum_{i=0}^{N-1} w_{ij}^s x_i \quad j = 0,1,2,....M-1 \ldots \qquad (1)$$

$$o_j^s = f(a_j^s) \ldots\ldots \qquad (2)$$

Where a_j^s in (1) is the j[th]-accumulated sum for layer *s* with M neurons and N inputs, w_{ij}^s is the weight between node i in layer s-1 and node j in layer s, x_i is the i[th] input and o_j^s in (2) is the output of the j[th] neuron of layer *s* given by the activation function [5].

3. **Hardware** neural node

The corresponding block diagram in fig.(1) is a single neuron that has N input data and N input weight each one has k-bit bus width. The calculation of the output pass in to two stages first is the multiplication then second is the accumulation of the result of the multiplication stage.

The multiplication stage has N multiplier each multiplier has two inputs with K-bit one for the data and the other for weight. Each multiplier give 2K-bit output but the next stage has K-bit input, therefore the output must be truncated to K-bit. This truncation represents a positive error in the range 0 *to* 2^{-K} that has average value equal to 2^{-K-1}.

The accumulation stage has M layer that can be calculated from equation (3), each one (i[th] layer) will have 2^{M-i} adder unit.

$$M = \log_2^{N^*} \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (3)$$

Where N* is the nearest maximum power of 2 number to N and N is the number of input data buses in neuron.

Each adder unit in the layers of the accumulation stage has two K-bit inputs with single output that has (K+1)-bit but the next layer is design to K-bit input, therefore the output must be truncated to K-bit. This truncation represents a positive error in the range 0 *to* 2^{-K-1} that has average value equal to 2^{-K-2} [7, 8]. However, the average value of the error in each layer *Ei* calculated from equation (4) and the total average error in accumulation stage calculated *Ea* from equation (5) below.

$$E_i = 2^{-k-2} * 2^{M-i} = 2^T \ldots \qquad (4)$$

where $T = M-K-i-2$

$$E_a = \sum_{i=1}^{M} \frac{E_i}{2^{M-i}} = M^* 2^{-K-1} \ldots\ldots \qquad (5)$$

The total error in the neuron *En* is the sum of the multiplication stage and the accumulation stage with the effect of truncation in the next stage to the error of the past stage as in equation (6).

$$E_n = N * \frac{2^{-K-1}}{2^M} + E_a \approx 2^{-K-1} + M * 2^{-K-1} = (M+1) * 2^{-K-1} \quad \ldots\ldots \quad (6)$$

The result of error in output in range from $0 \; to \; (M+1)2^{-K}$ with positive average equal to $(M+1)2^{-K-1}$ and the mean square error (MSE) equal to $(M+1)^2 2^{-2K-3}$

However, the number of data inputs in ANN is varied from 5 to 1000, that is mean the value of M is varied generally between 3 to 10 and it can be writen as: $M = 2^v$.

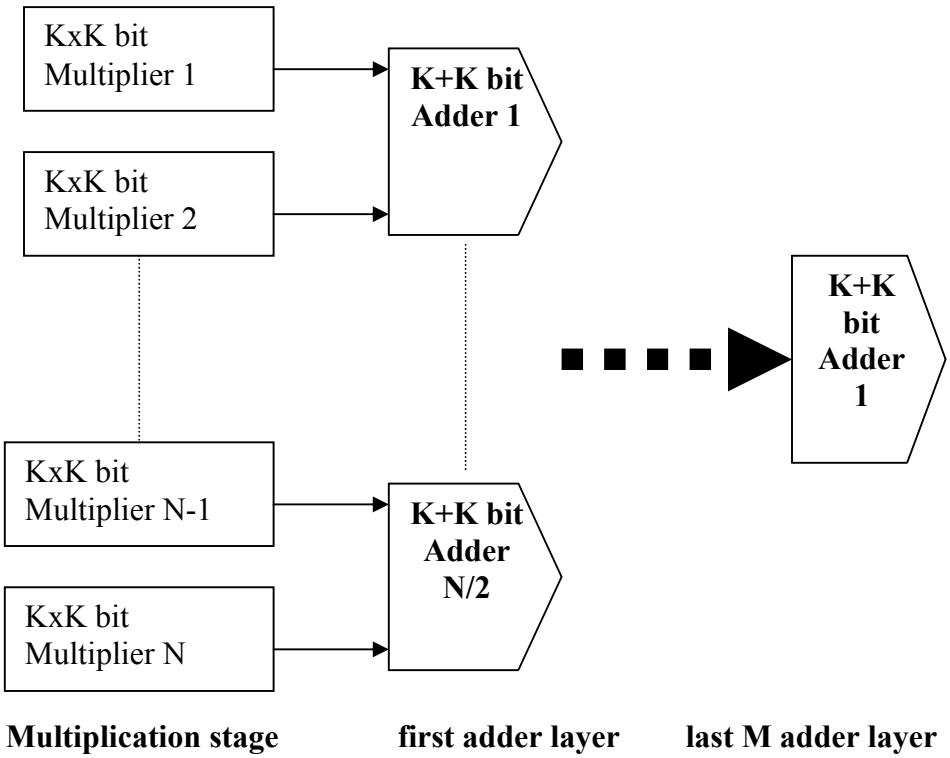

Fig. (1): Block diagram for implementing a single neuron.

4. Reduction of the Error

There are three suggestion steps to reduce the total error in the output of the hardware neuron as follows:

1- Expanded data bus in the internally layers of the accumulation stage and make all the truncation in the last layer. That means if the first layer has K-bit the second layer will have (K+1)-bit and the i^{th} layer will have (K+i-1)-bit and so on until to the last (M) layer that will truncate the data to the original data bus K-bit. This process will reduce the error in the accumulation stage to range $0 \; to \; 2^{-K}$ that has average value equal to 2^{-K-1}. That means that the total error output is in range $0 \; to 2^{-K}$ with positive average equal to 2^{-K-1} and MSE equal to 2^{-2K-3}.

2- Add a round unit to the multipliers of the multiplication unit as shown in fig. (2).The round unit sums up the truncated values of each two-neighbor multiplier then approximate the

result to the LSB value. Hence, if this value is greater than the half of the LSB it will add to the accumulator else it will neglect if it is smaller than the half of the LSB. This process will reducte the error in the multiplication stage to range -2^{-K-2} *to* 2^{-K-2} that has average value equal to zero and MSE equal to 2^{-2K-4}.

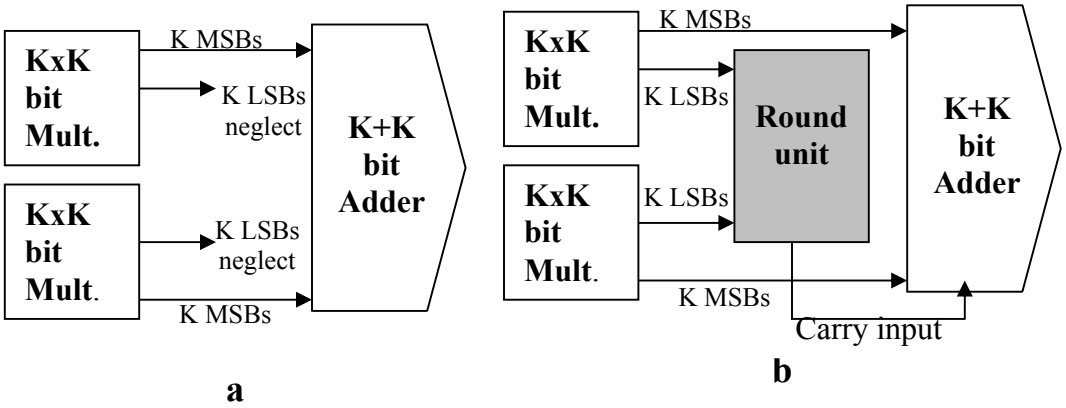

Fig.(2): Block diagram for multiplier and the first layer in accumulation unit in neuron (a) Original and (b) after step 2.

3- Add a round unit to the last layer of the accumulation as shown in fig. (3). The round unit approximates the truncated value to the LSB value if this value is greater than half of the LSB and neglected the truncated value if it is smaller than half of the LSB. This process will reduce the error in the multiplication stage to range -2^{-K-2} *to* 2^{-K-2} that has average value equal to zero and MSE equal to 2^{-2K-4}.

The total error of the neuron output after applying the three steps is in range -2^{-K-1} *to* 2^{-K-1} that has average value equal to zero and MSE equal to 2^{-2K-2}.

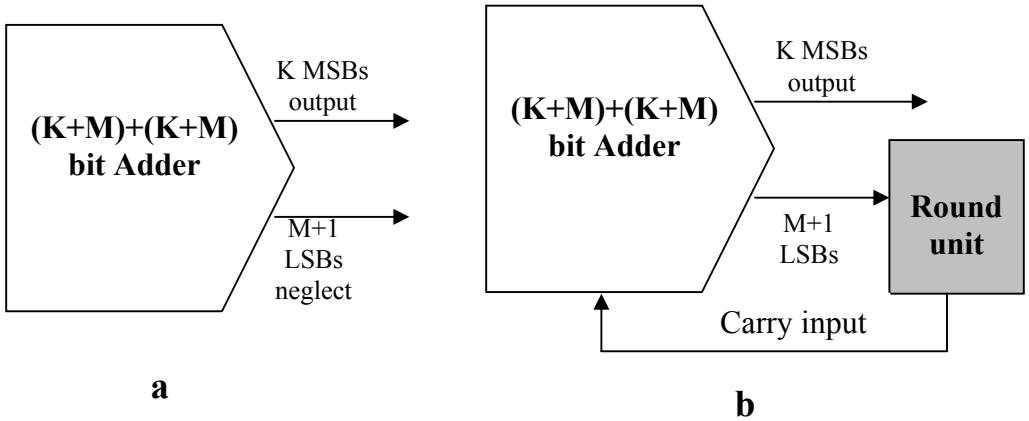

Fig. (3): Block diagram of the last layer in accumulation unit in neuron (a) Original and (b) after step 1 and step3

5. Software Implementation

The proposed steps to reduce the error is tested using software simulation to implement an ANN equivalent to hardware circuit of this network.

At first the original ANN is implemented using back propagation algorithm [9] using C++ or matlab, then it will be modified to represent the hardware ANN by applying the following steps:

a) Define all values in the program as integers fill in rang $\pm (2^{k-1}-1)$.
b) Replace each addition process c= a+b with c= (a+b)/2.
c) Replace each multiplication process z= x*y by the following steps:

1- Convert the second number (y) to binary number (yy).
2- Multiply the digits of the binary number (yy) by weights that are the powers of two to calculate the set {u}.
3- Use shift and add (under condition b) approach [10] between the number a and {u} to satisfy the multiplication process.

The simulation of step 1 in section (4) require to replace condition (b) $\{c = \sum_{i=0}^{N-1} a_i$ (under condition (b)) $\}$ in the original program to become $c = \sum_{i=0}^{N-1} a_i / 2^M$. The simulation of step 2 in section (4) require to replace condition (b in condition c3) to become c= (a+b+0.5)/2. While the simulation of step 3 in section (4) require to replace condition (b) to become c= (a+b+0.5)/2.

6. Example

This example discus the effect of the reduction of the error in a single neuron from the hidden layer in a neural network that has 54 nodes in input layer (M=6), 30 nodes in hidden layer and 10 nodes in output layer and use 8-bit data bus (k=8) for all nodes. The error of the output in table (1) is calculated with respect to the maximum amplitude of the input and output (maximum value equal to 1). The original error is calculated from section (3) and the other values of the error is calculated from the steps of section (4) using M=6 and k=8.

Table (1): The error of the output calculates with respect to the maximum amplitude.

Error	Range *1000	Average *1000	MSE $*10^{-6}$
Original	0 to 27.34	13.67	93.46
Step1	0 to 7.81	3.9	30.52
Step1 with step 2	-3.9 to 5.86	1.953	24.79
Step1 with step 3	-3.9 to 5.86	1.95	24.79
All steps	-3.9 to 3.9	0	15.25

The above ANN is simulated using C++ program to represent an ANN that is used to detect the ten digits. The learning of the original software simulation (without any truncation) gives an error 3.7%. The reuse of the original software simulation with additional function to represent the effect of the truncation as in section (5) will give an error 14.2%. The modification of the truncation function to represent the effect of the steps in section (4) will give the results in table (2). Table (2) shows the practical results of the simulation of the ANN in the example, this table shows that the truncation increases the total error from 3.7% to about 14.2%, while the proposed method will reduce this to 5.9%. The cost of the neuron in table (2) is calculated by the classic design of adder and multiplier in [7], while the additional cost for the steps 1, 2 and 3 must be calculated from the specific design with routes of the reference [7].

Table (2): The simulation results of the error and the FPGA cost of the network

Method	Average of the percentage error	FPGA Cost Cell / Neuron
software	3.7	------
After truncation	14.2	14880
Step1	9.6	15036
Step1 with step 2	8.3	15792
Step1 with step 3	8.4	15048
All steps	5.9	15804

7. Conclusion

The implementation of ANNs in hardware has been proven to be beneficial for intelligent systems requiring fast computation of data. But the bounded width of the data in the hardware implementation will generate an error in the output of the neuron, this error will add to the result of the output as additional error. The bounded width of the data will generate an additional error value. The additional error value will fall in the range 10^{-4} per single process (addition) but it will represent an accumulated additional error in the result reach to about 10%.

The proposed method reduces the effect of the accumulated error depend on rounding the truncated values to give an optimal result in the output. This process will reduce the accumulated additional error value from about 14% to about 6%. However, it will increase the efficiency of the output 8% with a small additional cost less than 6%.

8. References

[1] Christodoulou, C., S. Michaelides, C. Pattichis, and K. Kyriakou, "Classification of Satellite Clouds Imagery Based on Multi-feature Texture Analysis and Neural Networks," *IEEE International Conference on Image Processing*, V.1. 2001, p 497-500.

[2] Soren, K. R., "Hidden Neural Networks: Application to Speech Recognition," *Neural Computation* V.11 1999, p. 54.

[3] Jayaraman, R., "Physical design for FPGAs," *Proceedings of the 2001International Symposium on Physical Design*, Sonoma, CA, p 214-221.

[4] Dimond, K., and K. Pang, "Mapping VHDL descriptions of digital systems to FPGAs," IEE Colloquium Digest: Computing and Control Division Colloquium on Software Support and Cad Techniques for FPGAS (Field Programmable Gate Arrays), n 094, p 9/1-9/3, 1994.

[5] Contreras, G, and Nava, P, **"**Design, Implementation and Testing of an FPGA-based Neuro-Coprocessor", Dept. of Electrical and Computer Engineering, The University of Texas at El Paso 500 W, 2002.

[6] Advance Product Specification, "Virtex-E 1.8 V Field Programmable Gate Arrays", DS022 (v1.0) December 7, 1999.

[7] Hikawa, H. "Implementation of Simplified Multilayer Neural Network with On-chip Learning," *Proc.of the IEEE International Conference on Neural Networks (Part 4)*, Vol. 4, 1999, pp 1633-1637.

[8] Schelin, C.W., "Calculator Function Approximation", Am. Math. Monthly, vol.90, 1983.

[9] Max van Daalen, Peter Jevons, and John Shawe Taylor. A stochastic neural architecture that exploits dynamically recon_gurable FPGAs. Proceedings IEEE Workshop on FPGAs for Custom Computing Machines, pages 202{211, April 1993.

[10] Oudjida, A.K., "High Speed and Very Compact Two's Complement Serial/Parallel Multipliers Using Xilinx's FPGA ", CDTA/Microelectronics laboratory 128 Chemin Mohamed Gacem El-Madania 16075, Algiers,Algeria,1996.

تقليل مستوى الخطأ لبناء الشبكة العصبية

د . ظافر رافع زغير

قسم هندسة الحاسبات و البرامجيات / كلية الهندسة
الجامعة المستنصرية

الخلاصة: -

ان عملية بناء الشبكات العصبية الذكية (ANNs) باستخدام المكونات المادية يكسبها سرعة عالية مقارنه بالبرامجيات التي تنفذ على معالج احادي مايكروي و ذلك بسبب كون البناء باستخدام المكونات المادية يعتمد على المعالجة المتوازية. ان واحدة من احدث طرق البناء المادي المستخدمه هي مصفوفة البوابات الواسعة القابلة للبرمجة (FPGA) و التي تتميز بالمرونة و السرعة العالية. ان من محددات البناء باستخدام المكونات المادية هي كون ناقل البيانات محدد بسعة معينة ثابته و هذا التقييد يسبب اضافة نسبة خطاء الى النتائج النهائية. سيقوم هذا البحث باشتقاق المعادلات التي تمثل نسبة الخطاء الاضافي و تقترح طريقة مناسبه لتقليل هذا الخطاء و بزيادة كلفة قليلة للحصول على نسبة خطاء قليلة مع كلفة غير عالية.