Enhanced Hardware Implementation of Hybrid Stochastic Neural Network using FPGA

Rafid Ahmed Khalil
Email:rafidamori@ymail.com

Mustafa Salim
Email:mshhalh@gmail.com

Abstract

Most of the traditional digital implemented systems uses fixed point or floating point for representing and processing data. An alternative approach is to represent data as random bits that are distributed along the sequence. To be precise, stochastic logic can be considered as a solution for hardware size for application that consume physical area like neural networks as it uses logic gates to implement complex operations and its inherits resistance to bit flips noise. To avoid some of the problems that this type of processing suffers from, a combination of stochastic logic and classical logic (fixed point) is used to implement a neural networks (Fully connected feed-forwards) that is characterized by FPGA large size consuming. The stochastic logic is utilized have to implement part of the multiplication operations in the hidden layers of network and LFSR is used as a random generator for conversion of weights and activation functions outputs. The hardware utilization of Spartan 3E-500K FPGA results are compared with another network of the same size. A discussion of some of the issues that related to this methodology faces is also presented.

Key words: Artificial neural networks, LFSR, Probabilistic computation, Stochastic arithmetic, FPGA, Stochastic logic.

تنفيذ شبكة عصبية عشوائية هيجنة ومحسنة بأستخدام FPGA

رأفد احمد خليل

مصطفى سالم

الخلاصة

أغلب الانظمة الرقمية تطبق العلميات الرياضية عن طريق استخدام التمثيل الرقمي ذو النقطة الثابتة أو الفاصلة العائمة أن البديل هو ترميز الارقام المراد معالجتها بسلسلة من الصفر والواحد وبصورة عشوائية أي المنطق العشوائي، حيث أعتبر كبديل للمنطق الاعتيادي في التطبيقات التي تستهللك مساحه سيلكونية كبيرة كونه يحتاج الى داوئر بسيطة لتنفيذ العمليات الرياضية المعقدة ومقاومته للضوضاء (bit flips).تجنباً لبعض المشاكل التي يعاني منها هذا النوع من المعالجة تم دمج هذا المنطق مع المنطق الاعتيادي في تنفيذ شبكة عصبية نوع-(Fully connected feed-forward)التي تتميز باستهلاكها الكبير لشريحة FPGA حيث تم أستخدام المنطق العشوائي في أجراء جزء من عمليات الضرب في الطبقات الخفية من الشبكة العصبية مع استخدام مولد أرقام عشوائية من نوع (LFSR) في أجراء عملية تحويل الأوزان وأخراجات دوال التفعيل. تم مقارنة النتائج مع شبكة أخرى من نفس الحجم ومناقشة بعض النتائج.

Received: 30 – 7 - 2013 **Accepted: 30 – 10 - 2013**

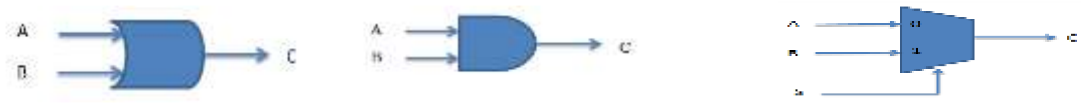
1- Introduction

Artificial neural networks (ANNs) are traditionally implemented using software tools such as MATLAB, the hardware implementations take the advantage of the inherent parallelism of ANNs and also are much more faster if compared with software solutions, also desirable that the circuits involved in the implementation of ANN must be as small as possible in order to increase the number of elements that can be built in a single chip [1], Also increased scaling of semiconductor devices, soft errors in the logic circuits caused by ionizing radiation are a major concern, particularly for circuits operating in harsh environments such as space. Existing fault tolerance methods militate against bit-flips with system-level techniques, such as error-correcting codes and modular redundancy. The stochastic approach, in contrast, naturally tolerates bit errors due to its probabilistic information encoding [2].There are few works on stochastic logic related to neural network, the most impressive work was in [3] where the authors describe for first time the state machine to implement linear and nonlinear function, In Peng Li and et [4]proposed a methodology to implement the non-linear activation function of neurons.Da Zhang and HuiLi [5]applies stochastic theory to the design and implementation of field-oriented control of an induction motor drive using a (FPGA) device and integrated neural network (NN) algorithms.

Thispaper is divided into three sections: section one describes the basic principles of stochastic logic, section two shows the simulation results of some elements and hybrid neural models and finally section three will illustrate the hardware results.

2- Stochastic logic

Stochastic computing is another type of representation that is different from traditional representation like fixed and floating point. Ganines [6] proposed this representation where the real number is converted to a stream of redundant bits that is distributed along the streams. The number of set value of these bits is direct relation and this depends on range. Each range has a link with a coding approach for example if the range [0, 1] or unipolar exactly the stochastic number p = x where x is normal number. So if x=0 the entire stream with bit set to zero, if x=0.5 then half of the bit in stream will be set to one and randomly distributed, for bipolar representation [-1, 1] the p random variable of all bits will be set if x=1 and so on. This representation allows the use of the simple logic gate to implement a complex mathematic element such as adder, multiplier and other elements. Figure (1) shows some of the combinational elements that can be used with such special type of representation.



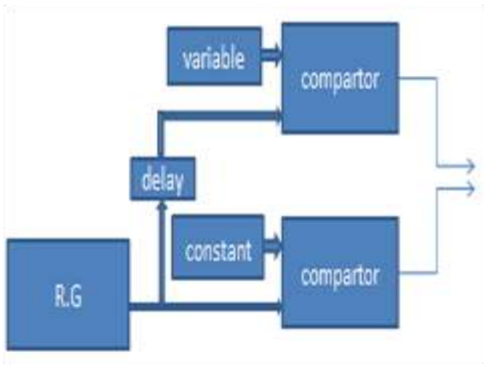
P(C)= P(A)+P(B)-P(A)P(B) **P(C)= P(A)P(B)*P*(*C*) = *P*(*S*) *P* (*B*) + (1- *P*(S)) *P* (*A*)**

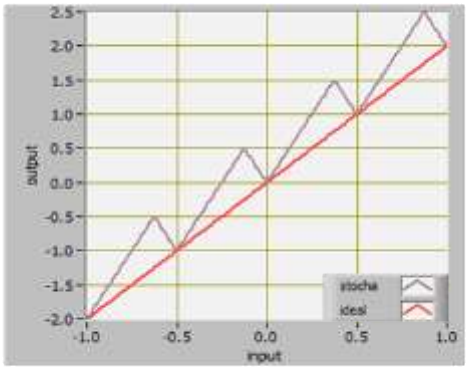
Figure (1): Combinational elements.

3- Simulation results of stochastic elements and proposed neural network

Simulations have been made using a Lab View program for most combinational stochastic elements that can be used by artificial neural networks while comparing them with the double precision floating point representation results which may be considered as optimal results for the majority of the systems. The comparisons are used to see the degree of validity of the theory of random and the possibility of finding the best use of such theory. Also, intentionally, two types of random generator have been used, the embedded Lab View random number and Fibonacci LFSRs that is built to see the effect of the random fluctuation error and the correlation while investigating their its effect on the response of the stochastic components. Finally, the goal is to find the best neural models that can be implemented on a physical entity that is suitable for many applications.

The general style of the multiplication operation in the random logic is illustrated in Figure (2).a, where the constant has a value equal to 2 and multiplied by the variable and that change from -1 to 1 with delta=0.01.Also, the streams of random numbers is generated using LFSR and has a 3CC6H as initial seed.In addition for the variable, there are two registers for delay with 00 as initial value for input variable and this delay register is inserted to reduce the correlation due to that different bits will be generated .





(a) Stochastic multiplication method **(b) Stochastic versus ideal multiplication**

Figure (2): Stochastic multiplication.

The result of multiplication is shown in Figure (2).b, the correlation effect appears clear where there is a massive error. The delaying register fails to remove the correlation. In addition, this method was not feasible due to an added extra digital and thus depleting the amount of physical entity, A second method is introduced which is based on building another random engine. Figure (3) demonstrates the results of the same experiment but with two 14 bits random engine; the first has an initial seed equal to 3cc6h and the second has a seed equal to 0279h. These two values have been carefully chosen and are practical. Accordingly the correlation effect has been reduced and almost totally removed. Despite that this method adds extra digital logic, this method is preferable because the generator is simply just only a collection of D flip-flops with some XOR gate, so this method will be followed in reading all other elements.

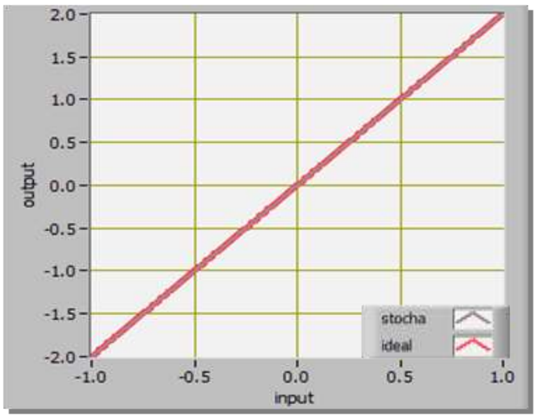
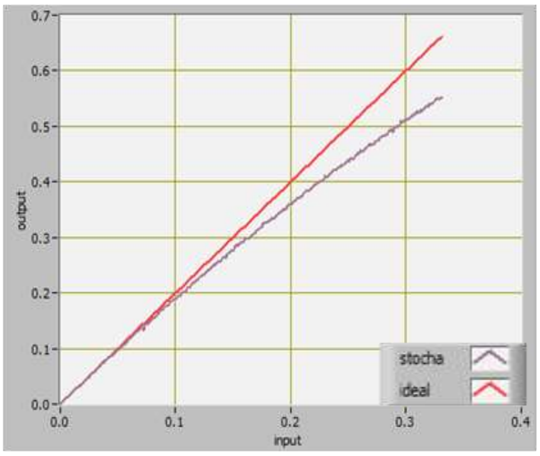


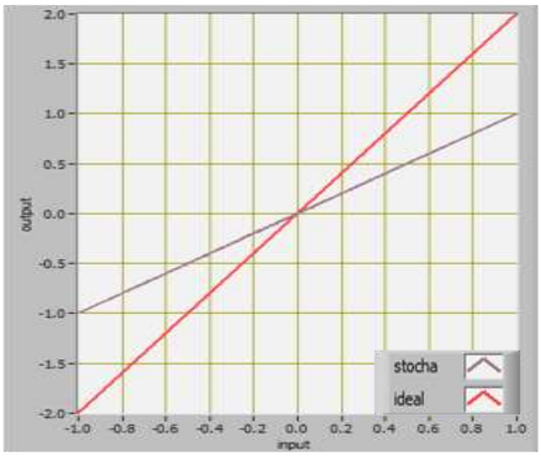
Figure (3): Stochastic versus ideal multiplication.

For the addition operation, the result for OR gate can be seen in Figure (4).a, using Lab View random generator for each input. As noted, the element is able to produce the correct output at a small value of input variables, after that the OR gate no longer cannot be used as an adder operator due to effect of saturation and increase in the value of p*q (where p and q are represent inputs value).

The addition with scaling results are shown in Figure (4).b,where the two inputs change from-1 to 1 with delta=0.001 between each iteration. Also, the selector is achieved using a counter that flips each time between 0 and 1 and thus provides a probability with 0.5 value for each given input. Finally the element will generate an output with a probability which is a scaled sum of the input probabilities. It should be noted that the shift operation can solve the problem and thus increase the number range of the specific system.

In addition to mathematical operations, the second important task is the generation of complex nonlinear activation functions. For a small number of neurons, the look-uptableor piece wise approximation can be used but for a large number of neurons, a stochastic approximation can be used.





(a) OR result.(b) Mux result.
Figure (4): Addition results.

The tanh(x) function can be easily generated either by state-machine [4] or count- comparator [7] methods and both cases have a scalar factor.

Figure (5) illustrates the experimental results of the stochastic tanh(x) function for different value of N (where N is the number of states that has been used). Note that 14 bit LFSR has been used for bipolar randomization process with 0803h as a seed.

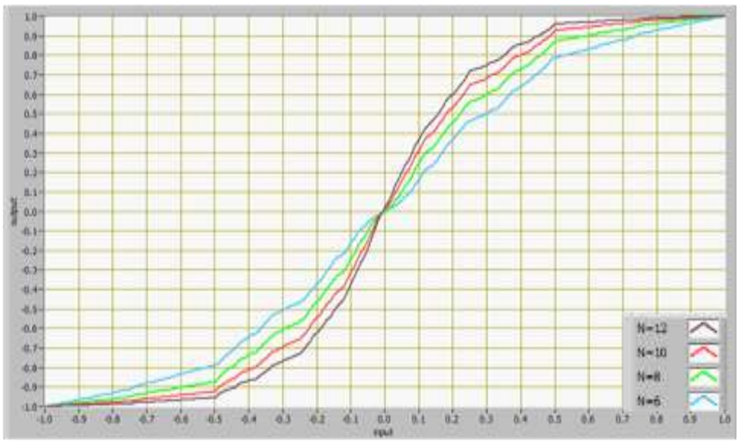


Figure (5) Hyperbolic sigmoid based stochastic logic based LFSR.

Bends are apparent and can be seen on the function curves clearly in Figure (5). The reason for that is the random number generator correlates between adjusting bits which is just a shift and thus correlation is obtained in the entire generated numbers. If the Lab View random generator is used instead of 16383 bits length.The graph of the generated functions is shown in Figure (6) for different N.

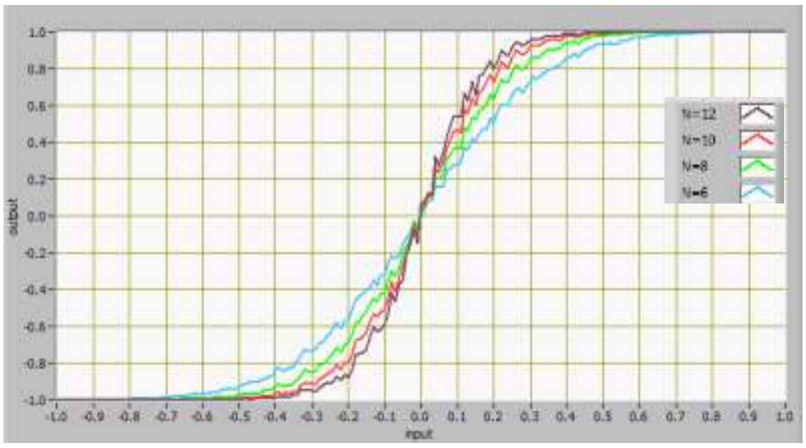


Figure (6): Hyperbolic sigmoid based stochastic logic based Lab View random.

In Figure (6) the ripples are a result of the closely linked design with the problem of random fluctuations as the number of onesin the stream does not match the required number that represent input. The problem can be solved by low pass filter or increase the number of stream bits length.

Figure (7) illustrates the results of applying 64535 iteration. The fluctuation effect is abolished approximately, also it is necessary to note that the error has a small difference in each time the simulation process occurs due to that the random generator has a function that changes its initial seed and thus the stream of number and eventually all the numbers generated sequence changes and changes.

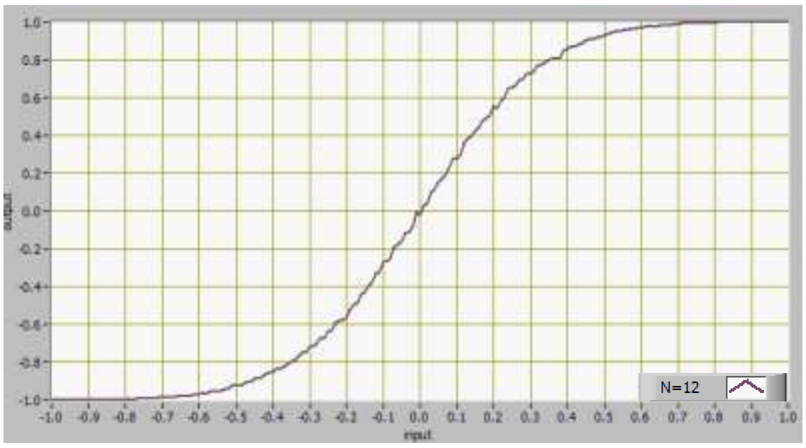


Figure (7): Hyperbolic sigmoid with increased bits stream length.

The generator of random numbers from a NI cannot be performed directly on a physical entity. So, after conducting some experiments, a predominate generator was developed. The generator is a combination of two 16 bit LFSR, where the bits derived from the generators in specific sites are combined together through an exclusive-OR gate to produce a single random bit and all exerted bits are then combined together to produce 14 bits that represent the random number.

The resulting functions after applying the proposed engine are shown in Figure (8). Figure (9) shows both traditional hyperbolic sigmoid activation function (tanh) and hyperbolic sigmoid based stochastic logic (stanh) with 8 states. There is a scalar between them [4], the scalar value is N/2, and this Figure indicates that the error is very small (with a high accuracy)

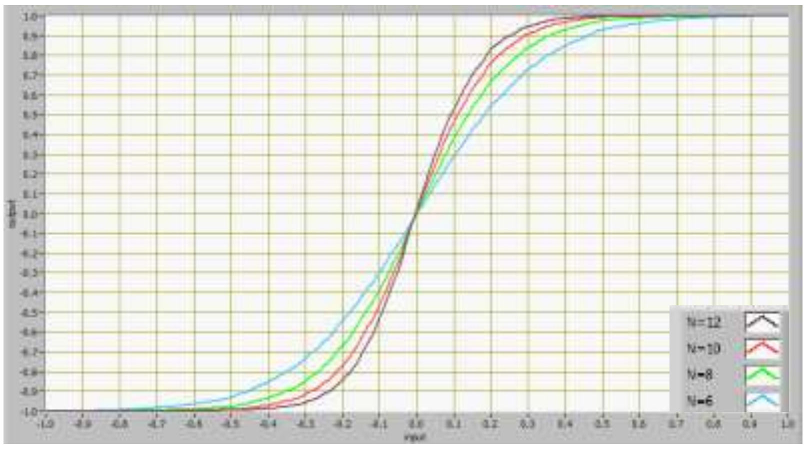


Figure (8): Hyperbolic sigmoid based stochastic logic with proposed random engine.

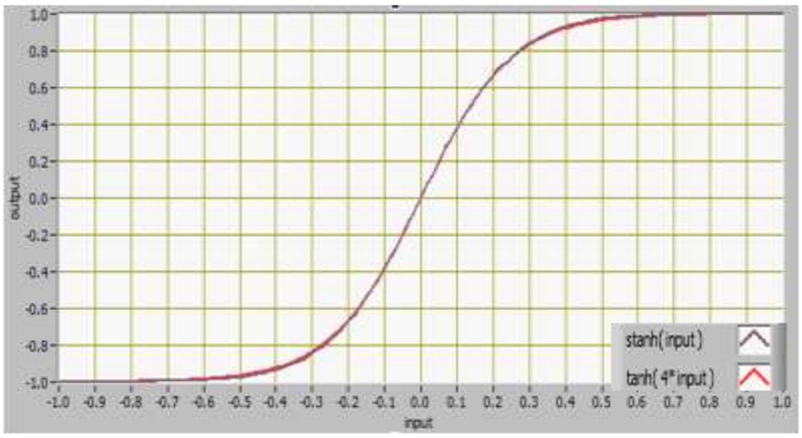


Figure (9): stanh (input) versus tanh (4*input).

For count-comparator method, the graph of the stochastic hyperbolic function with typical hyperbolic is illustrated in Figure (10).The stanh(x) function has charted under the Lab View random generator with 16383 cycle. Also, this activation function is evaluated when the gain is 41 and scalar value for input is equal to6.

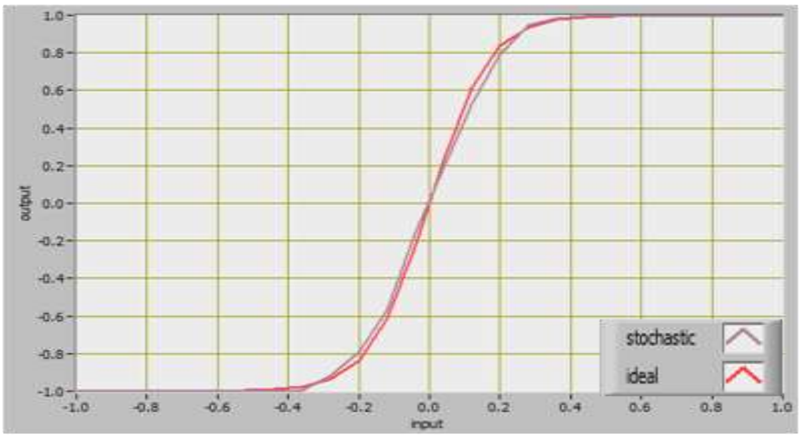


Figure (10): stanh(x) versus tanh (6*input).

Motivated by obtained results, and back to the main target: how to synthesize feed forward fully connected neural networks models based on stochastic logic or rather how to build a network on a silicon chip that consumes less space with a certain percentage compared with the conventional approach having the same size and type with the survival of performance within acceptable limits, two scenario can be can be taken into consideration as follows:

Scenario one:

The neural network consist of three main components multiplication, addition and activation function. These components can be applied by stochastic approach. Thegeneral characteristics of this model can be specified as follows:

1-High fault tolerance.
2- Resolution can be changed without re-designing the system,
3-The simple hardware implementation allows high clock rate.

The disadvantage includes that the correlation effect is very high, multiple random generator can be used for each or group of weight for solving this issue, but this is very expensive in terms of hardware as neural may have thousands of weights. This model can be used in simple applications with low accuracy.

Scenario two:

In this scenario hybrid model is used. That is a model that consist of some elements of stochastic logic and other elements based on fixed point logic. Two sub-models can be built. The block diagram illustrated in Figure (11), where this model uses stochastic multiplication with the stanh(x) activation. The addition operation is done in a usual way. This sub model is used to build a 1-6-1 network, the results obtained are shown in Figure (12).The graph was reassuring,their 0.4 scaling factor for the multiplier.

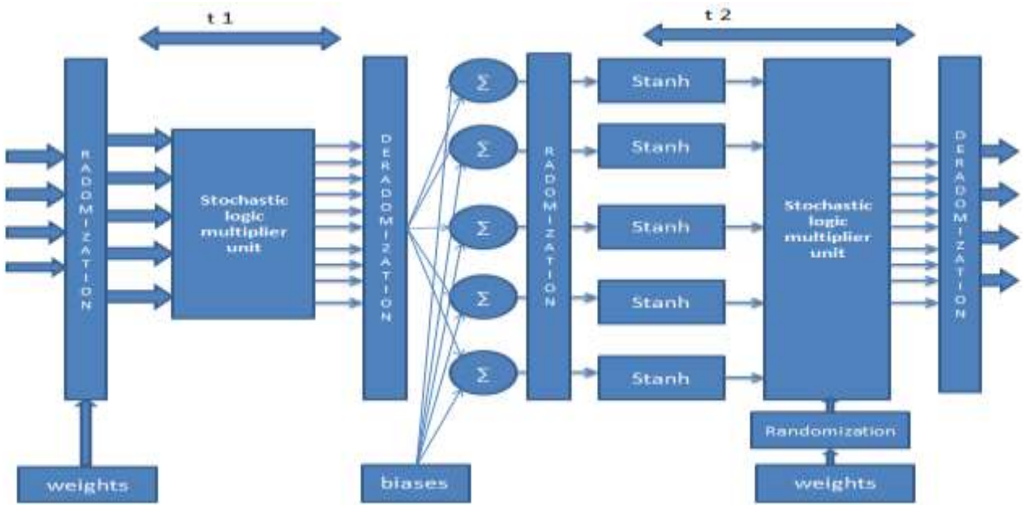


Figure (11): First sub-model of scenario two.

The most undesirable property of this sub-model is thatit is high time consuming model. From Figure (12),it can be seen that the first multiplication process needs t1 Stochastic Delay (SD) and after the addition operation, it isnoted that the stanh(x) function and the multiplication also need approximately t2 stochastic delay. Thus,the total delay in this case will be t1+t2+€,

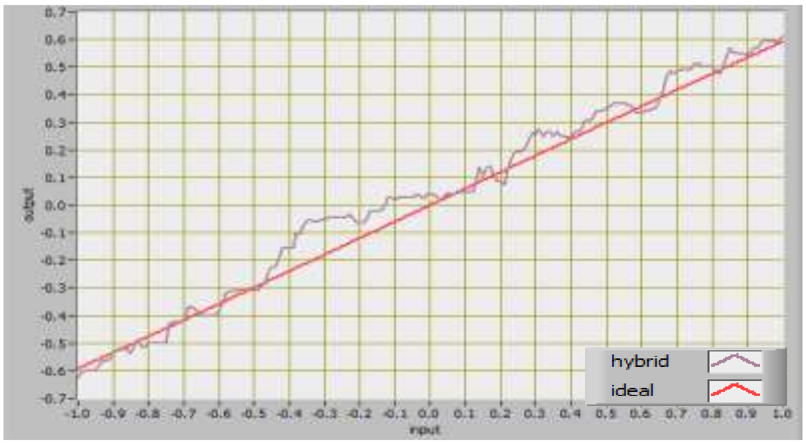


Figure (12): Output of first sub-model of scenario two.

where € is a small conventional delay related to the addition operation and can be neglected.Another issue related to this model is appeared during simulation that make the increase in the correlation effect that is caused by the state machine element where the output stream bits may not consist many ones or zeros that are not randomly distributed along the bit stream. This will affect,to a certain extent,the next stage which is the multiplication process. It is very important to note that the error curve cannot calculated in this model due to that the error will change every time the weights and biases is changed.

The second sub-model utilizes the multiplication process in stochastic manner only in hidden network layers since this area of the network isthe space physical entity is highly consumed. Other elements are also built in the conventional way. To validate the comparison response, a fully connected network 1-6-6-1 that contains four layer is built, the multiplications between second and third layer contain the most percentage of multiplier where 36 multipliers representing 75% of the total number are utilized. The weights coefficients are adjustment to have values equal to 1 without any biases with. tanh (x) is as activation function. The results obtained are shown in Figure (13) and the average error for this case=0.000948163 for 40 evaluated points, this indicate that a small errors has been happened.

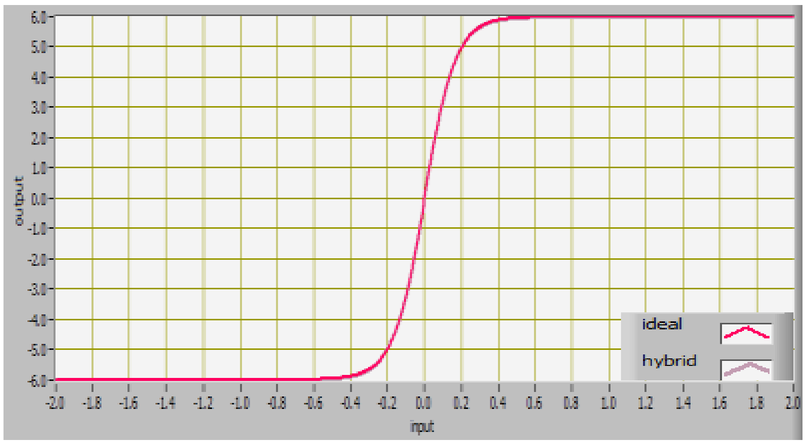
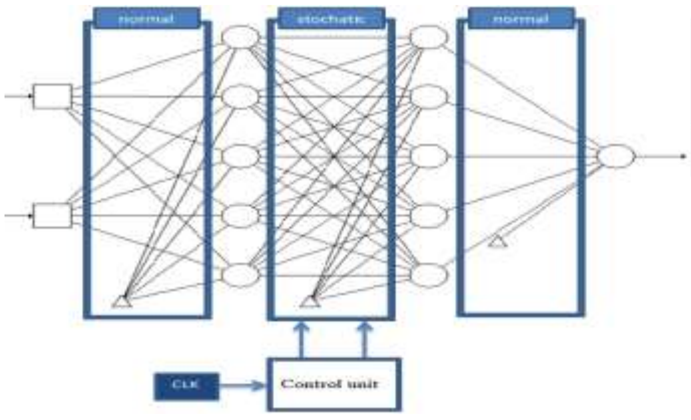


Figure (13): Output of second sub-model of scenario two.

4- Hardware results of the proposed neural network

This section presents a prototype reconfigurable architecture for implementing hybrid NNs on FPGA. From previous simulating analysis, it can be concluded that second sub-model of scenario two gives the best result compared with the other model. So this sub-model adopted. Figure (14) shows the general design. The complete structure of neuron shown in Figure(15).



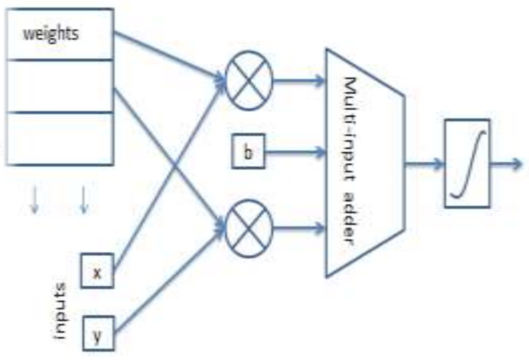


Figure (14): Structural diagram of the proposed network.

Figure (15): Structural diagram of the single neuron.

All neurons have log-sigmoid activations except the output neuron which only a linear activation. Second order approximation has been used for log-sigmoid. It structural block diagram is shown in Figure (16).Its comparison with ideal case is shown in Figure (17). The neural multipliers appear with less complexity. The control unit consists of counter and comparators. It is responsible for maintaining the overall timing synchronization as each random generator engine and bit stream counter need both a reset and enable signal that must operate at specific time. These signals has been chosen during the synthesis stage as base on the fact that each layer needs number of cycles to propagate data through it. The synthesis report from ISE 14.1 is shown in Table 1.

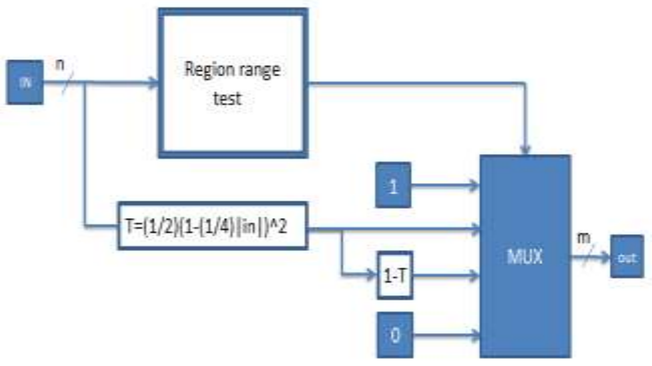


Figure (16): Structural diagram of the log-sigmoid.

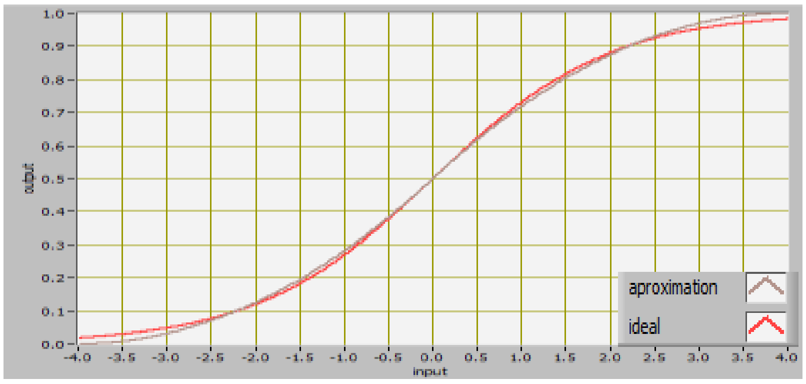


Figure (17): log sigmoid ideal and approximation with error.

Table 1: ISE Synthesizes report.

Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	2221	4656	47%
Number of Slice Flip Flops	1048	9312	11%
Number of 4 input LUTs	4185	9312	44%
Number of bonded IOBs	44	232	18%
Number of MULT18X18SIOs	20	20	100%

The network contains 50 multipliers, 20 of them are the embedded multipliers that available on the chip. The reduction ratio in the Slices about 23%. To get exact hardware utilization with stochastic logic multipliers vs. fixed point multipliers, single neuron has been used, then we increase the number of inputs from 8 up to 512 and at each increase we calculate chip utilization and occupied slice of Spartan 3E. The result shown in Figure (18) and Figure (19) respectively. The result indicate that stochastic give best utilization when large numbers of multipliers used.

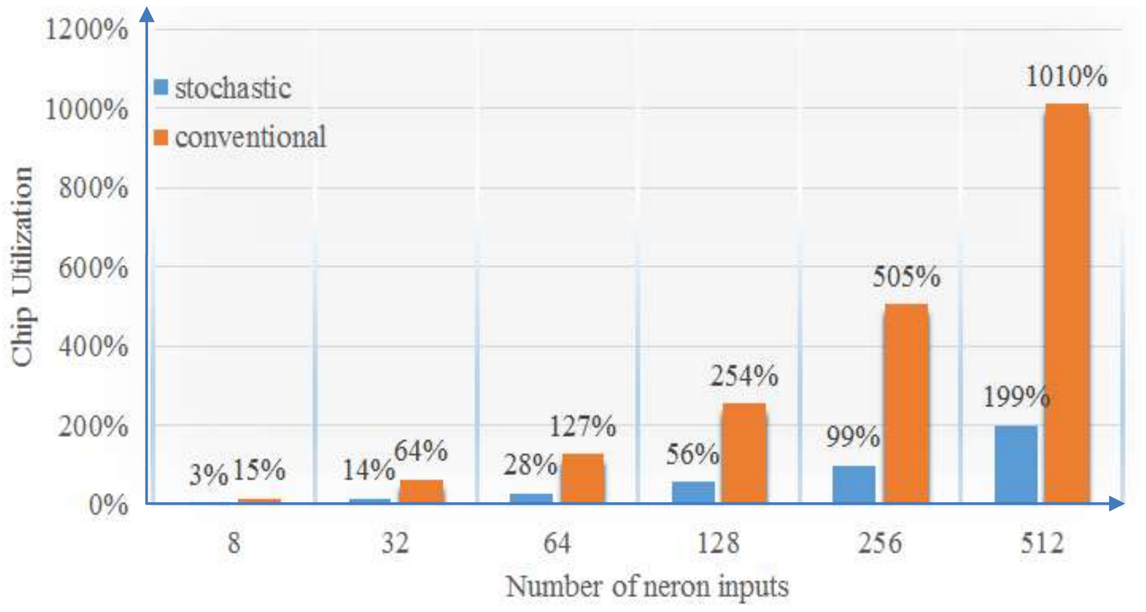


Figure (18): chip utilization.

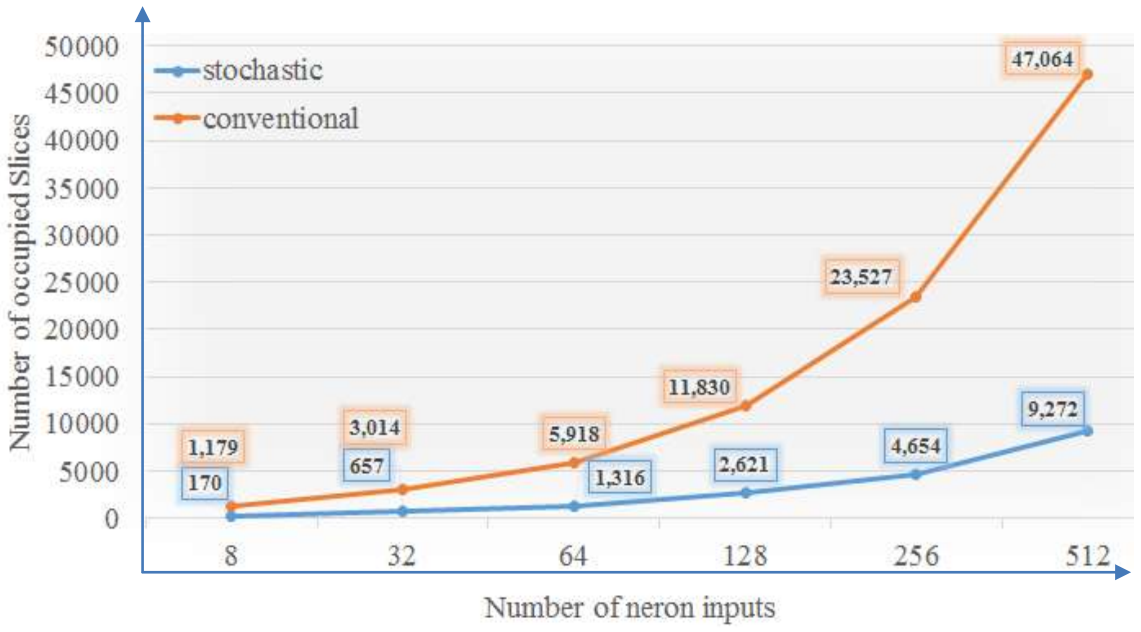


Figure (19): Occupied Slices curves.

After all designing parts of hybrid neural, the normal network is trained offline by BP algorithm using Matlab software and the weights and biases that are obtained are moved to RAM and the network is then tested. The result obtained from ISE simulation is shown in Table 2.

Table 2: Matlab and tested results.

X	Y	Ө (target)	Ө (est.) Matlab	Ө (est.)ISE
0.984807753012	0.17364817766	0.1745329251	0.17825573725	0.1560974121
0.866025403784	0.5	0.5235987755	0.5256470166	0.541473388
0.642787609686	0.76604444311	0.8726646259	0.872707205171	0.92230224609
0.5	0.86602540378	1.0471975511	1.046590735573	1.0879211425
0	1	1.5707963267	1.569631077900	1.55978393554
-0.5	0.86602540378	2.0943951023	2.094719841118	2.013671875
-0.64278760968	0.76604444311	2.2689280275	2.269871720067	2.210083007
-0.76604444311	0.64278760968	2.4434609527	2.444925918100	2.4242248
-0.86602540378	0.5	2.6179938779	2.619754798236	2.64205932617
-0.9396926207	0.34202014332	2.7925268031	2.794359769703	2.84933471

In Table 2 x and y are the training data with Ө as the target. After the training, Ө is estimated in Matlab, Table 2 indicates comparable values to Ө that are obtained from ISE simulation. The error between Matlab and ISE values have many sources:
1-Limited resolution has been used, for inputs(x, y) 8 bits , 14 bits for hidden layersand 16 bits fixed point for output layer.
2- Correlation effect between streams of bit.
3- Quantization error has been removed by using LSFR that converts approximately all possible output values of activation function and weights during conversion process.
4- Error dueto limit bits stream length at the output of the multiplication, so there will be a round approximation for each value.
5-Errors due to use of approximation of log-sigmoid functions.

5- Conclusion

This paper has discussed the basic view of probability representation, while developing an efficient approach for exploiting this representation of numbers in ANNs to reduce area and circumvents the problem of correlation that leads to low accuracy results. An approach simple and high redundant (due to using LFSR random generator).Number conversion from binary radix domain to probabilistic domain has been applied and complex arithmetic adders and multipliers implementation become realized in hardware by simple gates. The stochastic implementations of complex functions as tanh has been obtained with low hardware resources. Using a state machine that is based on markov theory or count comparator, all previous features become stochastic logic suitable for FPGA. Future work can be directed to use state machine to generate other functions and exploit this approach in suitable applications. Certain algorithms such testing stochastic computing in noisy environment. Using it in image processing, measure the power consumed, all can also be adopted.

6- References

[1] J.L. Rosselló, V. Canals and A. Morro "Hardware implementation of stochastic-based Neural Networks", In Proc. International Join Conference on Neural Networks (IJCNN 2010), Barcelona, July. 18-23 2010, PP. 1 – 4.

[2] W. Qian, X. Li, M. D. Riedel, K. Bazargan and D. J. Lilja, "An Architecture for Fault-Tolerant Computation with Stochastic Logic", IEEE transactions on computers, Vol. 60, No. 1, Jan. 2011, PP. 93 – 105.

[3] B. D. Brown, "Soft Competitive Learning Using Stochastic Arithmetic", M.Sc. thesis, Dept. of Electrical and Computer Eng., University of Manitoba, 1998.

[4]P. Li, D. J. Lilja, W. Qian, K. Bazargany and M. D. Riedely,"The Synthesis of Complex Arithmetic Computation on Stochastic Bit Streams Using Sequential Logic", IEEE/ACM international conference on computer-aided design (ICCAD), San Jose, CA, Nov. 5-8 2012, PP. 480 – 487.

[5] D. and H. Li, "A Stochastic-Based FPGA Controller for an Induction Motor Drivewith Integrated Neural Network Algorithms", IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS, VOL. 55, NO. 2, FEB. 2008, PP. 551 – 561.

[6] B.R. Gaines, "Stochastic Computing Systems", Advances in Information Systems Science, Vol. 2, 1969, PP. 37-172.

[7] J.L. Rosselló, V. Canals, A. Morro," Probabilistic-based Neural Network Implementation", IEEE world congress on computational intelligence, Brisbane, QLD , June. 10-15 2012, PP. 1-7.

The work was carried out at the college of Engineering. University of Mosul