HARDWARE IMPLEMENTATION OF AN ANN TRAINED BY GA AND PSO BASED ON FPGA[1]

Asst. Prof. Dr Hanan A. R. Akkar[2] **Ph.D. Student Sundus Dhamad Hasan[2]**

Abstract

In this paper, Particle Swarm Optimization-feedforward Neural Network (PSONN) and Genetic Algorithm-Neural Network (GANN) are proposed to enhance the learning process of ANN in term of convergence rate and classification accuracy. They have been tested and compared and the results applied in pattern classification. The experiments show that both algorithms produce feasible results in terms of convergence time and classification percentage. At the end of the evolutionary process of GANN for optimal structure, not only the best network structure for a particular application but also the trained network with few numbers of epochs is provided. A Hardware Design of ANN platform (HDANN) is proposed to evolve the architecture of ANN circuits using FPGA-spartan3 board (XSA-3S1000 Board). The HDANN design platform creates ANN design files using WebPACK^{TM} ISE 9.2i, and converted into device-dependent programming files for eventual downloading into an FPGA device by using GXSLOAD program from the XSTOOLS programs.

Keywords: Neural Network, Genetic Algorithm, Particle Swarm Optimization and FPGA.

التنفيذ العملي للشبكات العصبية الاصطناعية المدربة بواسطة الخوارزمية الجينية وامثلية الحشد الجزيئي بأستخدام مصفوفة البوابات المنطقية المبرمجة

الخلاصة

تم في هذا البحث التنفيذ العملي للشبكات العصبية الاصطناعية(ANN) المدربة باستخدام الخوارزمية الجينية(GA) وامثلية الحشد الجزيئي(PSO) باستخدام مصفوفة البوابات المنطقية المبرمجة حيث تم إقتراح برنامجين هما خوارزمية امثلية الحشد الجزيئي التي تدرب الشبكات العصبية (PSONN)، والخوارزمية الجينية التي تستخدم في تدريب الشبكات العصبية (GANN) لزيادة عملية التعلم من حيث معدل التقارب ودقة التصنيف. حيث تم اختبارهما ومقارنة النتائج في تطبيق تصنيف الأشكال (Pattern Classification). بينت النتائج بأن الـ GANN أسرع من PSONN مع أقل دقة، تبعا لنتائج كل مثال. وأن كلا الخوارزميتين ذواتا تأثير متساوي ولكن يمتلك PSONN كفاءة جيدة على GANN. وبشكل عام ، بينت التجارب أن كلا الخوارزميات تحقق نتائج جيدة من حيث الوقت والنسبة المئوية للتقارب. تظهر النتائج أن ANN التي صممت بإستخدام الـ GA ذات قدرة أفضل في التعميم وعدد أقل من التكرار لتدريب الشبكة العصبية الاصطناعية المصممة. تم اقتراح (HDANN) لتمثيل ANN بإستخدام (FPGA-spartan3 board (XSA-3S1000 Board. بواسطة HDANN يتم إنشاء الملفات الخاصة بتصميم ANN بإستخدام برنامج WebPACK^{TM} ISE 9.2i، التي يتم تحويلها الى ملفات البرمجة التي يعتمد عليها لتحميلها الى FPGA بإستخدام برنامج GXSLOAD من مجموعة برامج XSTOOLS.

[1] This paper was presented in the Engineering Conference of Control, Computers and Mechatronics On Jan. 30- 31/2011, University of Technology.

[2] Control and Systems Eng. Dept./University of Technology

1-Introduction

Artificial Neural Networks (ANNs) exhibite remarkable properties, such as: adaptability, capability of learning by examples, and ability to generalize. One of the most used ANN models is the well-known Multi-Layer Perceptrons (MLPs). The training process of MLPs for pattern classification problems consists of two tasks, the first one is the selection of an appropriate architecture for the problem, and the second is the adjustment of the network connection weights. Extensive research work has been conducted to tackle this issue. Global search techniques, with the ability to broaden the search space in attempt to avoid local minima, has been used for connection weights adjustment or architecture optimization of MLPs, namely Particle Swarm Optimization (PSO), Evolutionary Algorithms (EA), and Genetic Algorithm (GA) [1].

Reprogrammable systems have provided significant performance improvements for many types of applications. Many modern FPGAs have the ability to be reprogrammed in-system, in whole or in part. This has led some researchers to create dynamically reconfigurable computing applications within one or more FPGAs in order to create extremely high-performance computing systems. The technology of reconfigurable computing is still in its infancy, however, due in large part to the high cost, in terms of power and configuration time, of dynamically reprogramming an FPGA [2].

Thus, there is great interest in implementing neural networks in reprogrammable systems, both because of the speed benefits, as well as because the reprogrammability of the FPGAs can support the reconfiguration necessary to program a neural network [3].

2-Artificial Intelligent Systems

The Artificial Neural Network (ANN), Genetic Algorithm (GA) and Particle Swarm Optimization (PSO) are examples of AI techniques [4]. Each of these techniques has demonstrated some success at solving simple AI problems [3].

An Artificial Neural Network (ANN) is an information processing paradigm that is inspired by the way biological nervous systems process information. The Backpropagation (Bp) algorithm is commonly used learning algorithm for training ANN. Bp algorithm is used in ANN learning process for supervised or associative learning. During training, the network tries to match the outputs with the desired target values [5].

The intelligent techniques, such as Genetic algorithm (GA) and Particle Swarm Optimization (PSO), have been developed rapidly [4]. Genetic algorithm was introduced by John H. Holland in 1960's where GA was a probabilistic optimization algorithm. GA is a family of computational models inspired by evolution. The original idea came from the biological evolution process in chromosomes. GA exploits the idea of the survival of the fittest where the best solutions are recombined with each other to form new better solutions. There are three processes in GA which are selection, crossover and mutation [6].

PSO is a simple concept adapted from nature decentralized and self-organized systems such as bird flock and fish schooling [7]. The PSO was introduced by Kennedy and Eberhart 1995 as a population based stochastic search and optimization process. It's a population-based algorithm in which individual particles work together to solve a given problem.

The population (or swarm) and the member called particle is initialized by assigning random positions and velocities then the

potential solutions are flown through the hyperspace [8].

3- Programmable Logic Devices (PLDs)

Programmable Logic Devices PLDs (Programmable Modules) are modules that have been developed to be customizable for particular functions at the last stage of fabrication. PLD can be programmed, erased, and reprogrammed many times, allowing easier prototyping and design modification [2]. In the mid 1980s a new technology for implementing digital logic was introduced, the Field Programmable Gate Array (FPGA). FPGAs were capable of implementing significantly more logic than PLDs, especially because they could implement multi-level logic, while most PLDs were optimized for two-level logic. One common use of the FPGA is the prototyping of a piece of hardware that will eventually be implemented later into an ASIC. FPGAs have been increasingly used as the final product platforms. Their use depends, for a given project, on the relative weights of desired performances, development, and production costs [9].

4- Design and Implementation of ANN using GA and PSO

One of the more intriguing possibilities is that of combining a neural network with other AI systems, like GA and PSO, to enhance the learning process in terms of convergence rate and classification accuracy.

Genetic Algorithm (GA) has been used to determine optimal value for BP parameters such as learning rate and momentum rate and also for weight optimization. Particle Swarm Optimization (PSO) is chosen and applied in Feedforward neural networks to enhance the learning process in terms of convergence rate and classification accuracy. Two programs called Particle Swarm Optimization feedforward Neural Network (PSONN) and Genetic Algorithm Neural Network (GANN) have been proposed.

5- The Proposed Genetic Algorithm-Neural Network (GANN)

To guide ANN learning, GA is employed to determine the best number of hidden layers and nodes, learning rate, momentum rate and weight optimization. With GA, it is proven that the learning becomes faster and effective.

The flowchart of GANN for weight optimization is shown in Figure (1). In the first step, weights are encoded into chromosome format and the second step is to define a fitness function for evaluating the chromosome's performance. This function must estimate the performance of a given neural network. The function usually used is the Mean Squared Errors (MSE). The error can be transformed by using one of the two equations below as fitness value.

$$Fitness = \frac{1}{MSE} \quad \ldots(1)$$

or

$$Fitness = \frac{1}{1+MSE} \quad \ldots(2)$$

In GANN for optimum topology, the neural network is defined by a "genetic encoding" in which the genotype is the encoding of the different characteristics of the MLP and the phenotype is the MLP itself. Therefore, the genotype contains the parameters related to the network architecture, i.e. number of hidden layers (**H**), number of neurons in each hidden layer ($\mathbf{N_H}$), and other genes representing the Bp parameters. The most common parameters to be optimized are the learning rate (η) and the momentum (α). They are encoded as binary numbers. The parameter, which seems to best describe the goodness of a network configuration, is the number of epochs (ep) needed for the learning. The goal is to minimize the ep. The fitness function is:

$$Fitness = \frac{1}{ep} \quad \ldots(3)$$

or

$$Fitness = \frac{1}{1+ep} \quad \ldots(4)$$

The flowchart of GANN for optimum network architecture is shown in Figure (2).

3-layer ANN is used to do the classification of three examples as shown in Figure (3). The parameters of GANN training algorithm are a one-point crossover with probability Pc=0.8 and mutation operator with probability Pm=0.05. The MSE is equal to 10^{-6} and the maximum number of generations (epochs) equal to 50. The fitness function is defined by equation (2). After several runs the genetic search returns approximately the same result each time as the best solution despite the use of different random generated populations and a different population size (for each example). Figures (4), (5), and (6) show the curves of MSE against the number of iterations of training for both GANN and Bp algorithms for each example respectively. These Figures show the training of the network by using GANN training algorithm for each example, reaching the lowest value of MSE with a very few number of generations as compared with the results from using Bp algorithm. The learning by using GANN algorithm is independent of the value of η and α. Conversely, Bp algorithm fails if unsuitable values of η and α are chosen. GANN training algorithm avoids local minima by searching in several regions. It has no restrictions on the network structure because it doesn't require backward propagation of an error signal.

In GANN for optimal structure, the initial population is evaluated, which includes a certain number of Bp training cycles. The maximum number of training cycles may be set relative to the size of the network. The fitness of an individual is defined by equation (4). The number of individuals in the population has been fixed equal to 10. The operators are the crossover with Pc=0.8 and mutation with Pm=0.05. The value of the minimum error (MSE) has been set to 10^{-6}. It has been necessary to train the neural networks for 100, 100, 1000 epoch for each example respectively. The results of this algorithm for the three examples are summarized in Table (1). It can be seen that the best network for each example can be obtained in a small number of generations with low number of epoch for training with Bp as compared with the old networks. At the end of the evolutionary process, not only the optimal network architecture for a particular application but also the trained network is provided.

6- The Proposed Particle Swarm Optimization - Feedforward Neural Network (PSONN)

PSO is one of the latest techniques that can be fitted into ANN. A swarm is made up of particles where each particle has a position and a velocity. The idea of PSO in ANN is to get the best set of weight (or particle position) where several particles (problem solution) are trying to move or fly to get the best solution [10].

In PSONN, the position of each particle in swarm represents a set of weights for the current epoch or iteration. The dimensionality of each particle is the number of weights associated with the network. The particle moves within the weight space attempting to minimize learning error (or Mean Squared Error-MSE or Sum of Squared Error-SSE). Changing the position means updating the weight of the network in order to reduce the error of the current epoch. In each epoch, all the particles update their position by calculating the new velocity, which they use to move to the new position. The flowchart of PSONN algorithm is shown in Figure (7).

The acceleration constants C_1 and C_2 are set equal to 2, the number of particles is equal to 15, and the MSE is equal to 10^{-6}. r_1=0.8, r_2= o.2, Φ =0.9 and the maximum number of iterations (epochs) equal to 6000. Figures (8), (9), and (10) show the curves of MSE against the number of iterations for PSONN training algorithm for each example respectively.

It has been noticed that the training of the network by using PSONN algorithm reaches the lowest value of MSE with less number of iteration (epoch) as compared with the results from using Bp algorithm. The PSO can avoid getting into the local optimal solutions because PSO has the probabilistic mechanism and multi-starting points (it's a global optimizer).

On the other hand, GANN is faster than PSONN with less accuracy for each example as shown in Table (2). Both algorithms are converged using the minimum error criteria.

For the correct classification percentage, it shows that PSONN result is better than GANN. GANN significantly reduces the error at a small number of iteration compared to PSONN. For overall performance, the experiments show that both algorithms produce feasible results in terms of convergence time and classification percentage.

7-The Proposed Hardware Design of ANN platform (HDANN)

Hardware implemented ANNs have an important advantage over computer simulation ANNs because they fully exploit the parallel operation of the neurons, thereby achieving a very high speed of information processing. The proposed Hardware Design of ANN platform (HDANN) is a circuit design platform built to evolve the architecture ANN circuits using FPGA hardware. Therefore, the system must have software that can be downloaded into an FPGA.

The basic hardware and software components of the proposed HDANN are shown in Figure (11). It consists of a computer and FPGA-spartan3 board (XSA-3S1000 Board).

After down-loading the design into the board, a Dc function generator has been applied to the input-pins of the ANN design while the output has been measured by an oscilloscope. Figure (12) shows these output data when the input to the network is applied.

It shows that the experimental result is the same as the simulation result shown in Figure (13) for example2.

8-Conclusions

Based on the experiments performed in this study, it can be concluded that GANN for training has proved to be superior to Bp algorithm in both accuracy and speed of learning because GAs avoid local minima by searching in several regions. GANN for training is independent of the parameters that the Bp depends on. Tests show that GAs obtain best weight vectors quickly for each example at the same efficiency.

The optimal structure, including the number of neuron in the hidden layer, learning rate, and momentum rate, is found by using GANN for designing an optimal ANN structure. The GA is used to find a network structure that is best able to classify data from a specific situation with a small number of epochs.

PSO is a simple optimization algorithm with less mathematical equations that can be effectively applied in ANN. The results of PSONN have shown that the PSO is an efficient alternative to ANN training as compared with Bp algorithm. On the other hand, GANN is faster than PSONN with less accuracy for each example.

PSO is similar to GA in the sense that they are both population-based search approaches and they both depend on information sharing among their population members to enhance their search processes using a combination of

deterministic and probabilistic rules. Both are Global optimizer algorithms. PSO is more computationally efficient (uses less number of function evaluations) than GA. The results show equal effectiveness but superior efficiency for PSO over GA.

HDANN platform has the ease of reimplementation due to the parameterized modules as well as the state of the art for the chosen FPGA platform. It possesses the speed of hardware while retaining the flexibility of the software implementation due to the reprogramming ability of FPGA. It has the potential to create ANN circuitry for AI applications.

References

1. Conforth M. and Meg Y., September 21-23, 2008, "Reinforcement Learning for Neural Networks using Swarm Inelligence", IEEE Swarm Intelligence Symposium, St. Louis Mo USA, pages(7).
2. Pellerin D. and Thibault S., 22 April, 2005, "Practical FPGA Programming in C", Prentice Hall PTR, pages (464).
3. Earl D. D., May, 2004, "Development of an FPGA-Based Hardware Evaluation System for Use with GA-Designed Artificial Neural Networks", PhD thesis, University of Tennessee, Knoxville, pages (147).
4. Engelbrecht A. P., 2007, "Computational Intelligence, an Introduction", J. Wiley & Sons, Ltd, USA, pages (288).
5. Picton P., 2000, "Neural Networks", Antony Rowe Ltd, Chippenham, Wilts, Great Britain.
6. Goldberg D. E., 1989, "Genetic Algorithms in Search Optimization and Machine Learning", Addison-Wesley Publishing Company.
7. Lazinica A., 2009, "Particle Swarm Optimization", Published by In-Tech, Vienna, pages (476).
8. Kiranyaz S., Ince T., Yildirim A., and Gabbouj M., 31 May, 2009, "Evolutionary Artificial Neural Networks by Multi-Dimensional Particle Swarm Optimization", Journal Neural Network, pages (15).
9. Deschamps J. P., Bioul G. J. and Sutter G. D., 2006, "Synthesis of Arithmetic Circuits FPGA, ASIC", and Embedded Systems, J. Wiley & Sons, pages (556) .
10. Kuok K. K., Harun S., and Shamsuddin S. M., 2010, "Particle Swarm Optimization feedforword Neural Network for Modeling Runoff", International Journal of Environmental Science of Technology, Volume 7, No. 1, pp. (67-78).

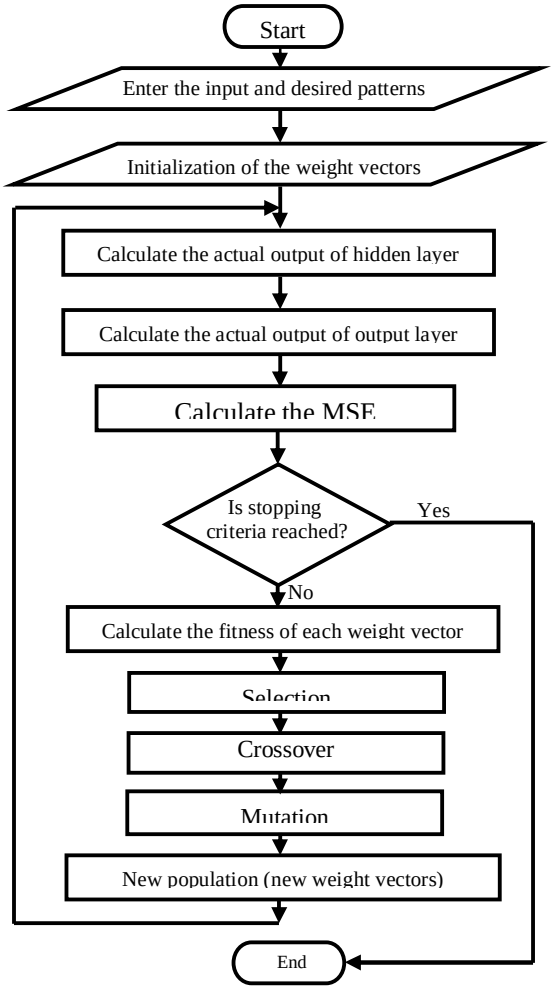


Figure (1) Flowchart of GANN weight optimization.

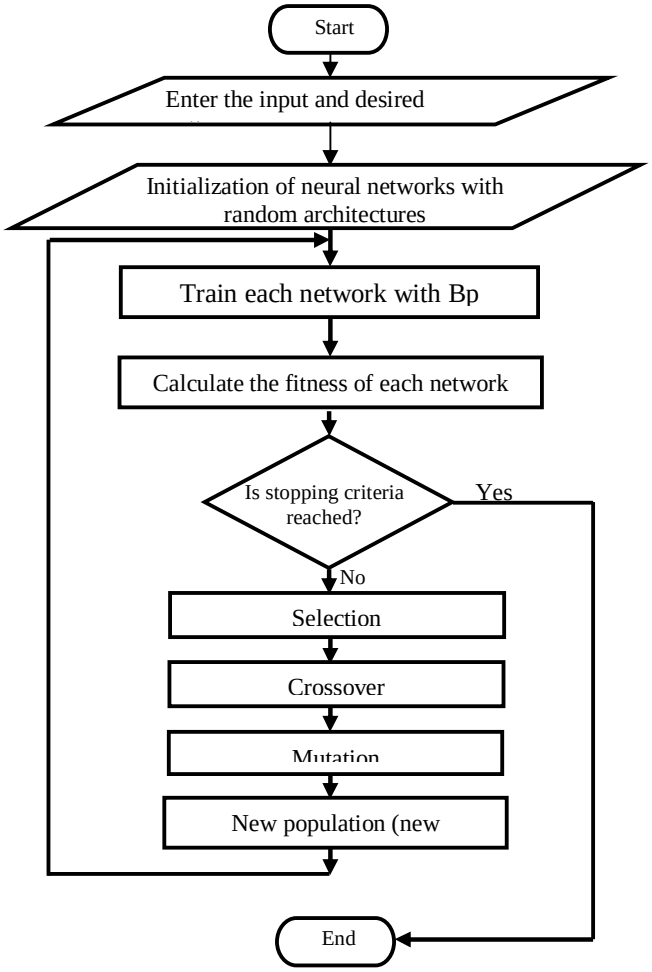


Figure (2) Flowchart of GANN for optimum network architecture.

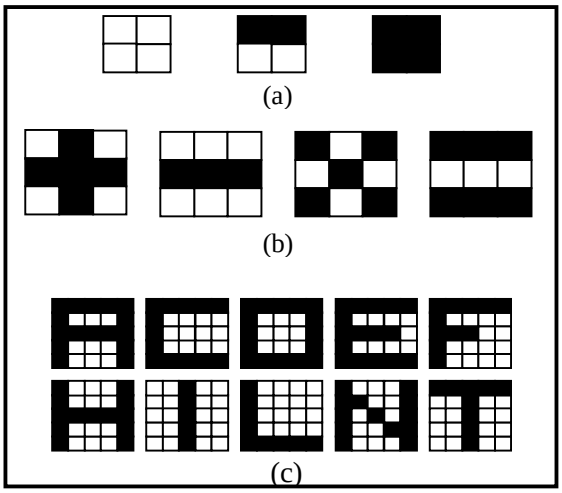
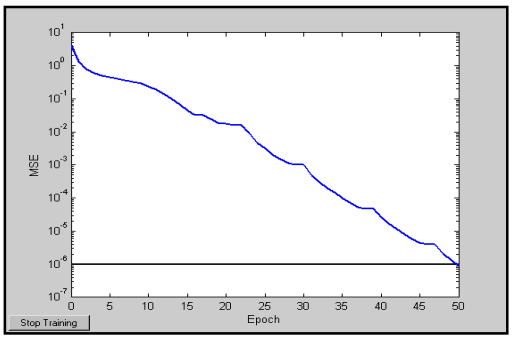
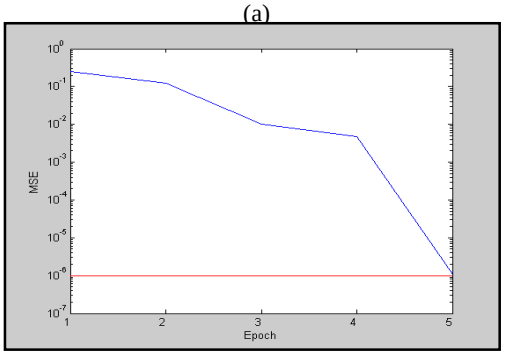


Figure (3) The tested examples: (a)example1(3patterns of (2x2 pixels). (b) example2: 4-patterns of (3x3 pixels). (c) example3: 10-patterns of (5x5 pixels.)

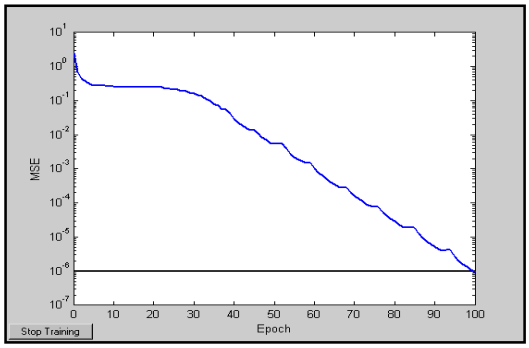


(a)

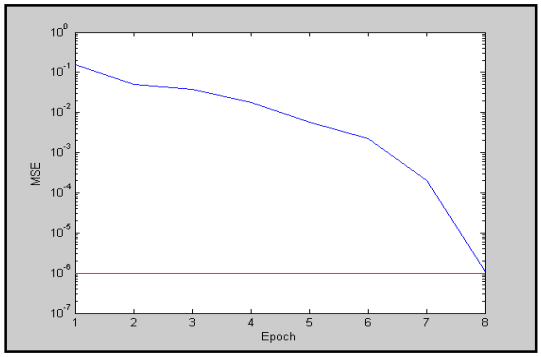


(b)

Figure (4) Training MSE with epoch for example1: (a) with Bp algorithm. (b) with GANN.

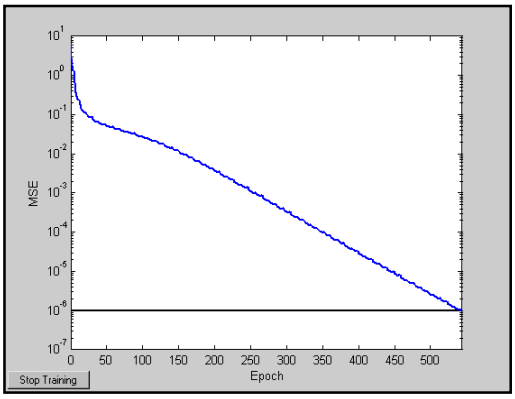


(a)

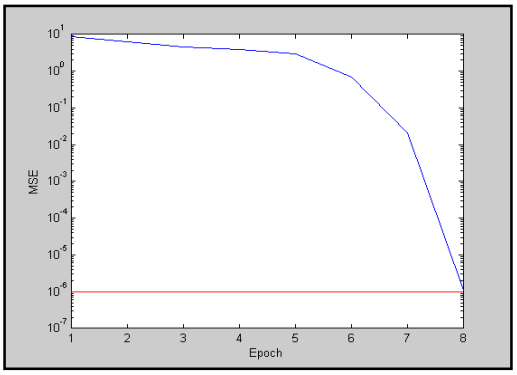


(b)

Figure (5) Training MSE with epoch for example2: (a) with Bp algorithm. (b) with GANN.



(a)



(b)

Figure (6) Training MSE with epoch for example3: (a) with Bp algorithm. (b) with GANN.

Table (1) The results of GANN for optimum structure for each example.

Examples	Example 1	Example 2	Example 3
No. of generation	5	5	8
No. of epoch for old network	50	100	544
No. of epoch for new network	20	52	50
η	0.11	0.59	0.74
α	0.63	0.62	0.83
No. of neurons in the hidden layer (old)	3	4	20
No. of neurons in the hidden layer (new)	5	8	16

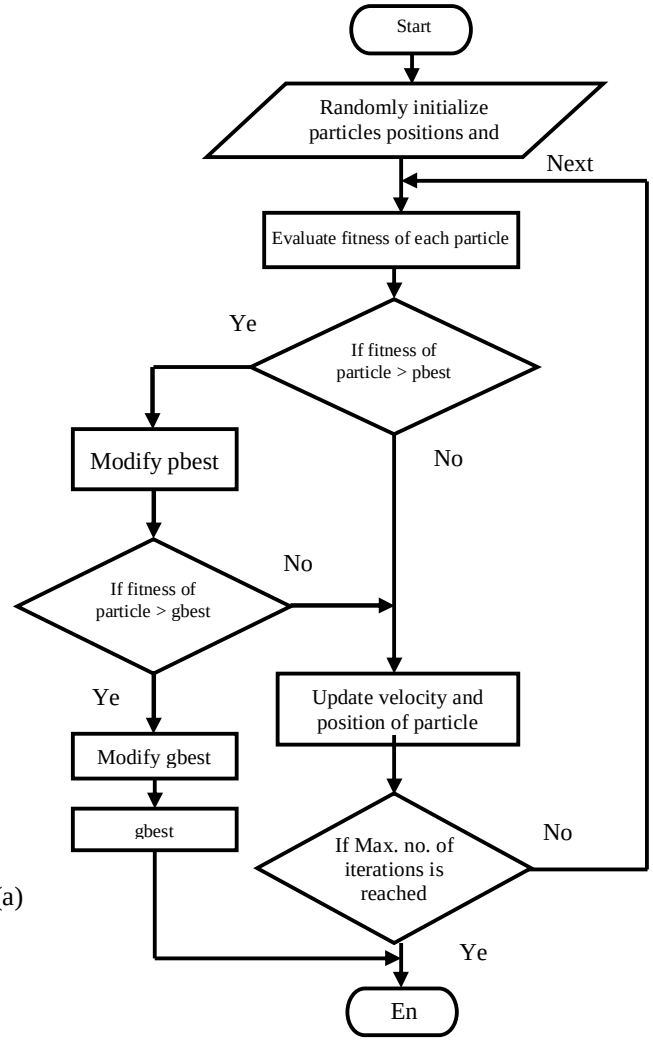


Figure (7) The flowchart of PSONN algorithm.

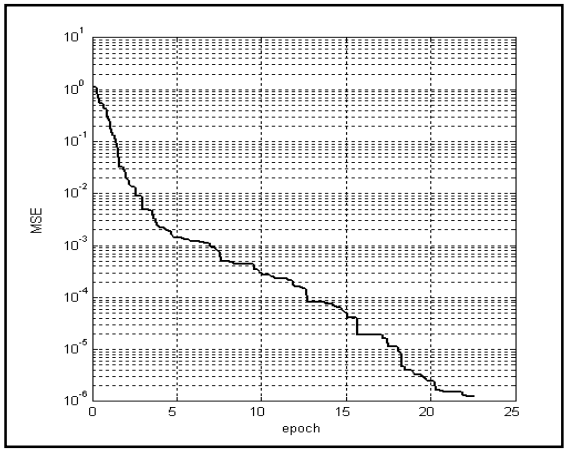


Figure (8) Training MSE with epoch for example1 by using PSONN algorithm.

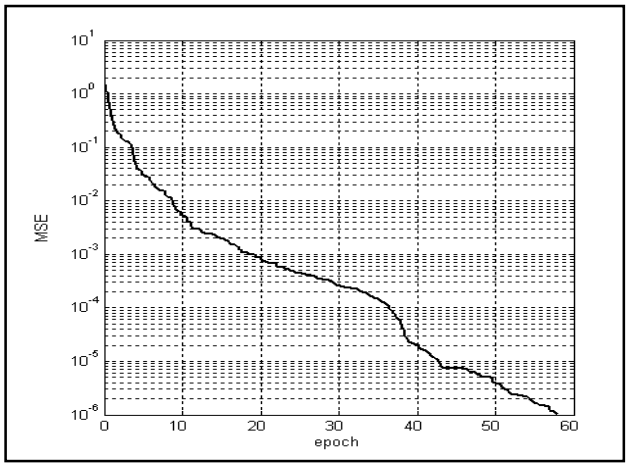


Figure (9) Training MSE with epoch for example2 by using PSONN algorithm.

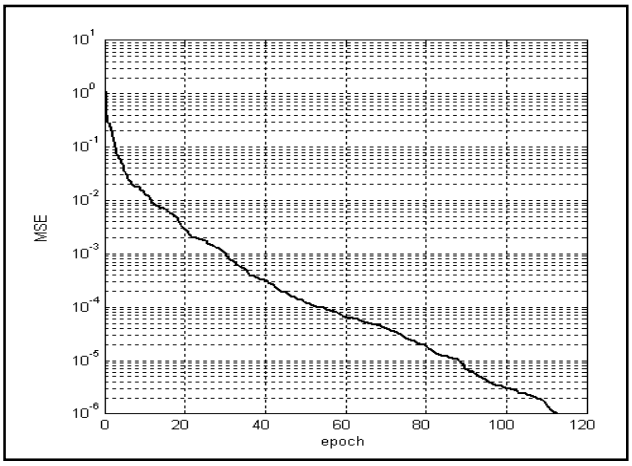
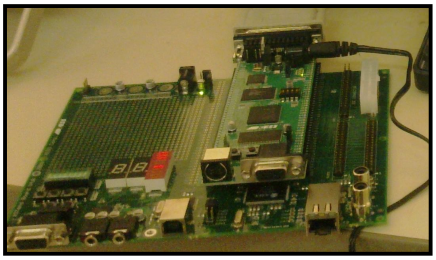


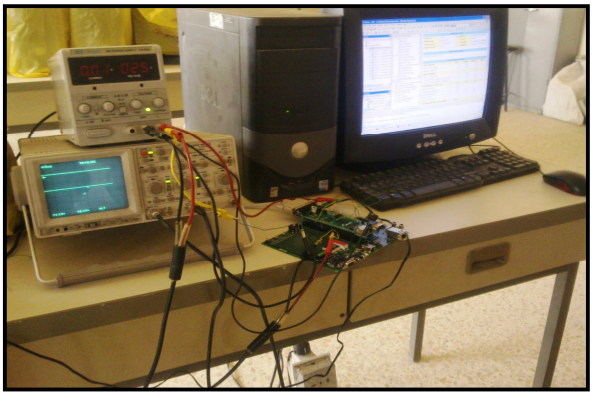
Figure (10) Training MSE with epoch for example3 by using PSONN algorithm.

Table (2) the comparison results between GANN and PSONN algorithms.

	GANN			PSONN		
Examples	a	b	c	a	b	c
Learning iteration	5	8	8	23	58	116
Error Convergence	4.008 e^{-6}	1.0001 e^{-6}	3.0009e^{-6}	9.75e^{-7}	7.86 e^{-7}	9.95e^{-7}
Classification (%)	93.2 %	96.24 %	92.83 %	99.98 %	99.99 %	98.42 %

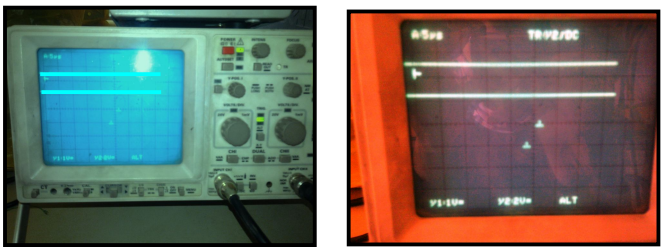


(a)



(b)

Figure (11) The Hardware Design of ANN platform (HDANN) system: (a) FPGA board (b) HDANN system.



(a) (b)

(c) (d)

Figure (12) the output data when the input to the ANN is applied: (a) Nu0 (b) Nu1 (c) Nu2 (d) Nu3.

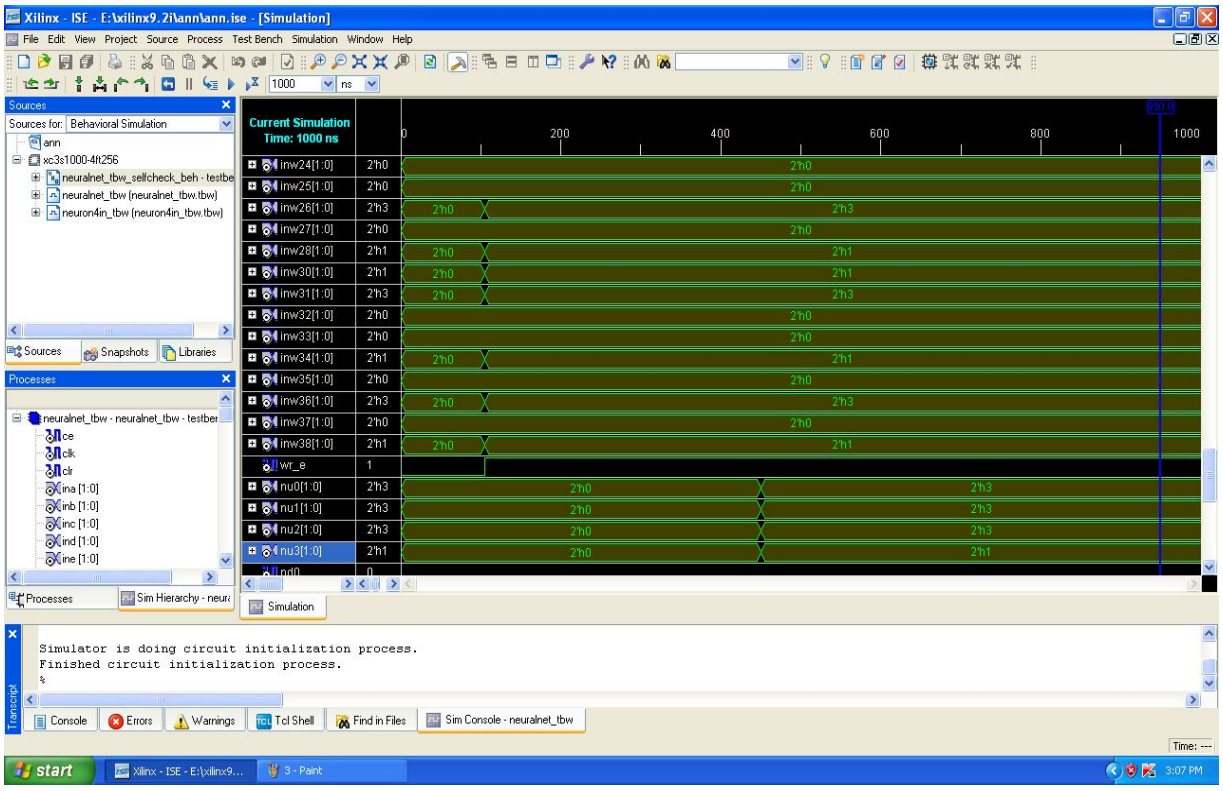


Figure (13) The simulation results of the ANN circuit of example2.