RAM-Based Neural Network Parallel Implementation on a Reconfigurable Platform and Its Application for Handwritten Digits Recognition

Shefa A. Dawwd, **Ali Al-Saegh**
shefadawwd@gmail.com, ali.alsaegh@uomosul.edu.iq
Computer Engineering Department, University of Mosul

Abstract:

Artificial neural networks (ANNs) are widely used in different areas of nowadays applications. Many challenges are imposed on the practical implementation of ANNs. Some of them are: the number of samples required to train the network; the number of adders, multipliers, nonlinear transfer functions, storage elements; and the speed of calculations in either training phase or recall phase. In this paper, the RAM-based neural network is investigated. No weights, adders, multipliers, transfer functions are required to implement it neither in hardware nor in software, but at a cost of large RAM utilization. In addition, a small number of samples are required for training. However, in hardware implementation, a large size of memory is required to train it. The network is implemented on the FPGA platform. The Stratix IV GX FPGA development board, which is provided on large on board RAM, is used. A considerable speedup of 237 is achieved in either training or recalling phases. A comparable error rate of 7.6 is achieved when MNIST (Mixed National Institute of Standards and Technology) database are used to train the network on handwritten digit recognition.

Keywords: Field programmable gate arrays, handwritten digits recognition, RAM-based neural network.

تصميم شبكة عصبية مبنية على ذاكرة الوصول العشوائي وتنفيذها على الوحدات القابلة لإعادة التشكيل وتطبيقها في تمييز الأرقام المكتوبة باليد

شفاء عبدالرحمن داؤود **علي الصائغ**
قسم هندسة الحاسوب، جامعة الموصل

الملخص: تستخدم الشبكات العصبية الإصطناعية في العديد من التطبيقات الموجودة في الحياة اليومية. ولكن هناك الكثير من التحديات التي تواجه تطبيق هكذا شبكات في الواقع العملي. ويمكن تلخيص تلك التحديات بالتالي: عدد النماذج المطلوبة لتدريب الشبكة، عدد عمليات الجمع والضرب وعدد الدوال غير الخطية وكذلك عدد وحدات الخزن المطلوب التعامل معها وتوفيرها، وسرعة الحسابات في طوري التدريب والفحص. تم في هذا البحث إستخدام الشبكة العصبية المبنية على ذاكرة الوصول العشوائي، حيث أن تطبيقها عمليا لا يتطلب أوزان أوعمليات جمع وضرب أو دوال تحويل ولكنه يتطلب توفير ذاكرة الوصول العشوائي. تتميز هذه الشبكة بالعدد القليل من نماذج التدريب المطلوبة ولكنها تكون على حساب حجم ذاكرة الوصول العشوائي. تم في هذا البحث تصميم شبكة عصبية من هذا النوع وتنفيذها بإستخدام مصفوفة البوابات المنطقية القابلة للبرمجة في الحقل. تم فحص النظام المقترح في تمييز الأرقام المكتوبة باليد وكانت نسبة الخطأ في التمييز هي 7,6% مع تحقيق تسريع جدير بالإعتبار ومقداره 237.

الكلمات المفتاحية: مصفوفة البوابات المنطقية القابلة للبرمجة في الحقل، تمييز الأرقام المكتوبة باليد، شبكة عصبية مبنية على ذاكرة الوصول العشوائي.

Received: 12 – 1 - 2014 **Accepted: 14 – 5 - 4**

1. INTRODUCTION

Conventional artificial neural networks' (ANNs) [1-3] constructions are built from the well-known weighted-sum-and-threshold artificial neurons called McCullogh and Pits which are comparatively simple processing units. These artificial neurons communicate with each other through a big set of weighted connections. The artificial neuron is depicted via two equations (1) and (2), the first one specifies a linear weighted sum of the inputs to the neuron and followed by the other one which represents a nonlinear activation function.

$$u = \sum_{j=1}^{n} w_j x_j \quad \text{-------- (1)}$$

$$y = \frac{1}{1+e^{-au}} \quad \text{--------- (2)}$$

Where u is the activation function, w are the weights, x are the inputs, a specifies the shape of the output sigmoid, and y is the output.

Numerous works proved that this kind of neural networks has a skillful generalization ability [2]. Nevertheless, the complexity of huge parallel nonlinearity calculations inherent in such models is a cost for that generalization. In the same context, hardware implementation of conventional ANNs involves the use of large numbers of adders and multipliers by the artificial neurons [3] which in turn challenging for entirely parallel implementation of the networks.

Moreover, training a multilayer neurons demands presenting the training samples in a repetitive manner in order to achieve suitable neuron weights and this leads to very long learning time [2, 4]. Implementation of training in hardware systems can be implemented either off-chip or on-chip [5, 6]. In off-chip learning, the values of weights are calculated externally by software and are then downloaded to the neural network which is then used only for recall. This is the easiest but least favored method, since training times can be long. It may be suitable for networks, which do not need to adapt to new data once trained such as the problem of pattern recognition or classification, but it is not suited for some other problems such as data compression. Off-chip learning does have the advantage in that it is easy to change the learning algorithm simply by modification of software. It also allows the use of floating point arithmetic for the algorithms which may not be feasible on a neural network chip. On-chip learning must be seen as the most desirable method, since it may open the way to stand-alone neural network chips. The main advantage of running the learning algorithm in hardware is the gain in speed. However, there is a trade-off in flexibility, where "programming" of the learning algorithm is difficult. Other obstacles to the development of on-chip learning are the extra chip area used, and the fact that many of the current or popular algorithms (e.g. back propagation) require global data.

To this end, the aim of this paper is to implement a NN on a reconfigurable FPGA platform for handwritten digits recognition, and hence looking for a robust and speeding hardware design is of great importance. However, accomplishing this through ordinary ANNs sounds to be not a very suitable idea due to the lack of necessary circuit density in FPGA while this type of ANNs, as it has been discussed earlier, consumes a lot of hardware resources.

Relying on the previous discussion, it is very important to look for another type of ANN to obtain a design with a simpler architecture and still has a generalization ability. Therefore, RAM-based NN has been chosen for implementing the design, whereas it has been proved to offer fast training and simplicity of implementation [2, 4, 7]. Besides that, training can wholly

be implemented on-chip in low area cost with low control complexity, high precision, and high performance.

The rest of this paper is structured as follows: Section 2 explains the RAM-based NN and its advantages over ordinary ANNs, Section 3 shows the designed NN, Section 4 describes the hardware architecture of the designed NN, Section 65 demonstrates the FPGA implementation, the results are presented in Section 66, then finally Section 7 concludes this paper.

2. RAM-BASED NEURAL NETWORK

Also known as n-tuple classifiers, offers keys to those presented problems although it was not initially aimed for considering that. RAM-based NN is easy to learn where it has the capability to learn with presenting the training set once only; besides the simplicity of its hardware which needs only memory slices and counters [2] while the ordinary ANNs needs a large number of adders and multipliers.

RAM-based NN employs the neurons that deal with binary input patterns. This network does not carry any information on its connections but rather it saves data in Random Access Memories (RAMs) i.e. inside the network's nodes or neurons, therefore, it is a weightless neural network. In order to learn this sort of NNs for recognizing different classes of patterns, it is necessary to construct a number of logic functions that describe the problem. In the testing stage, these functions assess true in case of a pattern belongs to the class that the function represents and false for all other classes [1, 2].

The 1-bit word RAM node used for constructing the RAM-based NN is shown in Fig. 1; the N-address lines are connected to a logical decoder for accessing only one memory location of 2^N available memory locations. It has a control input as well in order to toggle the RAM mode between "Write" in the training phase and "Read" in the testing phase.

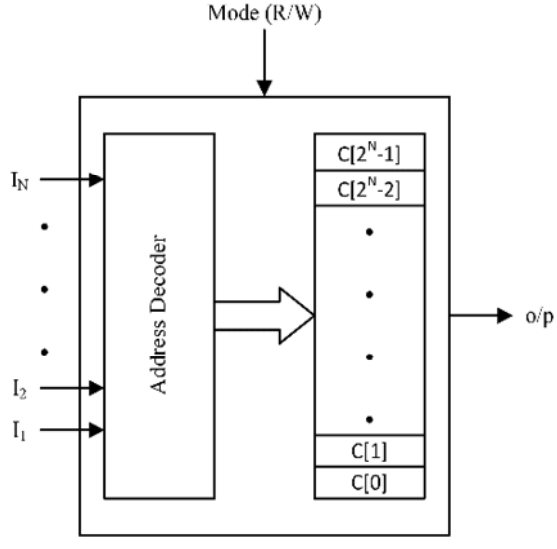

Fig. 1. 1-Bit RAM Node. The lines $I_1 - I_N$ for addressing the memory locations $C[0] - C[2^N - 1]$. The mode line is for toggling between the write and read modes.

The RAM node itself does not perform generalization (delivering the correct output for unseen patterns in the training phase) while it does produce a correct output for patterns stored throughout the training phase. Whilst the generalization is introduced by a discriminator device which is composed of a grid of RAM nodes, refer to Fig. 2. The address lines of each RAM (a tuple accesses only one of the memory locations in the concerned RAM) in the discriminator are randomly connected to N bits from a binary input pattern (a binary image in our study), so that a discriminator will consists of M RAMs each of N bits and the binary pattern is of size $M \times N$bits.

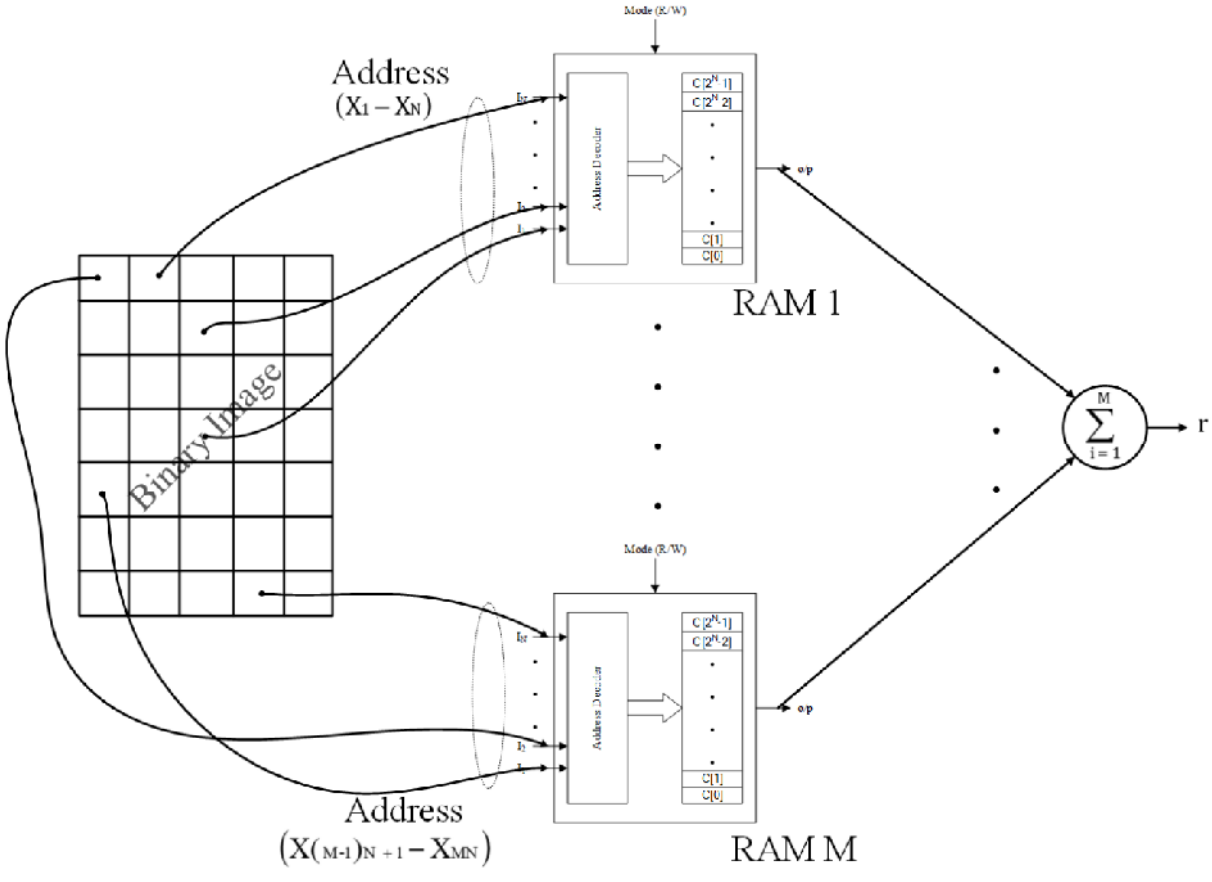

Fig. 2. A Discriminator. It consists of M RAMs each of N bits input connected to a binary input pattern in a random manner. The response (r) of the discriminator represents a counter for those RAMs with output of '1'.

Prior to the training phase, all RAM memory locations must be set to '0'. Then, during the training phase, a '1' is stored in a memory location addressed by an input vector $(X_1 - X_N)$. After training all of the patterns, memory contents will be either '0's or '1's. Normally, each bit in an input pattern is an input to one RAM only, however oversampling is possible for improving classification. Afterwards, the stored information is then used for the testing phase. Here it should be clear that a discriminator is trained on a specific class of patterns, hence a RAM-based NN with a number of discriminators equal to the number of classes must be assigned.

When unseen input patterns, which are composed of k classes are presented to the RAM-based NN that consists of k discriminators, Fig. 3, all of the addressed memory contents are read and summed to get a response (r) for each discriminator. So that r represents a counter for those RAMs with an output equal to '1'. The maximum r, which is equal to M, tells that the input pattern was presented in the training phase; on the other hand when none of the RAMs fires '1' $(r = 0)$ means that the input pattern did not appear in the training phase. In between values $(0 < r < M)$ represent the amount of likeness of the input pattern to the patterns in the training set [1, 7].

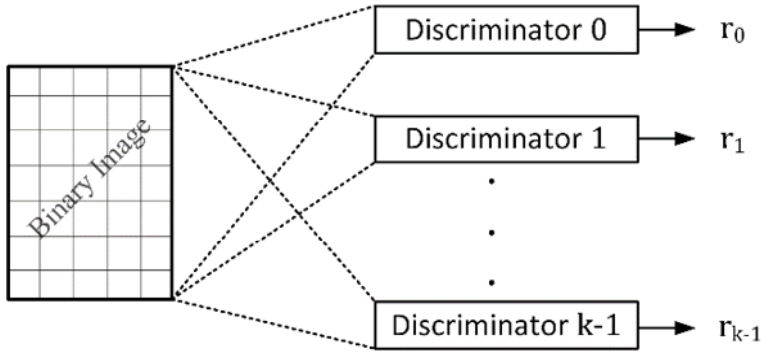

Fig. 3. A k-discriminator RAM-based NN. Each of the discriminators can be trained for recognizing one class of data. Therefore, this NN is able to recognize a k classes.

3. DESIGNED NEURAL NETWORK

The MINST data set of handwritten digits has been used. It consists of 60,000 training images and 10,000 testing images, each image is 28×28 pixel.

The designed RAM-based NN is composed of 10 discriminators as there are 10 classes (digit 0 to digit 9) in the data set. Oversampling has been used by a factor of three in order to achieve better classification at the end, such that each image is presented three times during the training phase to the corresponding discriminator. Hence, each of the discriminators is composed of 147 (3×49) RAMs each with 16 address lines (tuple size is 16 bits). Since each image is composed of 784 pixels, therefore, the number of RAMs needed for each discriminator is 49 $(784/16 = 49)$, but with the use of oversampling the number becomes 147. All training and testing images were converted to the binary format to be possible handling them throughout the designed NN. The training and testing algorithms for the designed NN are given below:

Training algorithm:

1. *Read train images and labels.*
2. *Change the format of images to binary.*
3. *Set tuple size (16), number of RAMs (147), and number of discriminators (10).*
4. *Select an image.*
5. *Specify the appropriate discriminator from the label of the image.*
6. *Start sampling the image.*
7. *Select a tuple randomly.*
8. *Store '1' in the RAM location determined by the tuple.*
9. *Repeat steps 7-8 until all image's bits are selected.*
10. *If the image is sampled for 3 times then continue, else go back to 6.*
11. *If all images are trained then continue, else go back to 4.*
12. *End.*

Testing algorithm:

1. *Read test images and labels.*
2. *Change the format of images to binary.*
3. *Tuple size, number of RAMs, and number of discriminators are the same as in the training stage.*

4. *Select an image.*
5. *Start sampling the image.*
6. *Select a tuple randomly.*
7. *Increment the response of all discriminators containing '1' in the RAM location determined by the tuple.*
8. *Repeat steps 6-7 until all image's bits are selected.*
9. *If the image is sampled for 3 times then continue, else go back to 5.*
10. *The maximum response represents the class of the image.*
11. *If the class is the same as given in the labels then the image is recognized, else it is not recognized.*
12. *If all images are tested then continue, else go back to 4.*
13. *End.*

4. H/W ARCHITECTURE

Unlike the traditional neural network node, each pattern that is to be classified using RAM-based NN requires M storage elements that distributed in M tuples. To improve the performance and achieve fast response time, each tuple should be accessed individually. This can be realized only by using separated memory banks (since that each memory can only be accessed once every clock cycle). However, to classify k patterns, a huge number of memory banks ($k \times M$) are required. That may seem impractical.

One memory location traditionally is composed of multiple bits (width of RAM). To cope with the problem mentioned above, in these paper M memory banks of k-width are used. Then, M tuples (RAM banks) of k-width can be used to classify k classes of patterns. Each of the RAM columns is now connected to a discriminator. A k-bit RAM representation can be seen in Fig. 4.

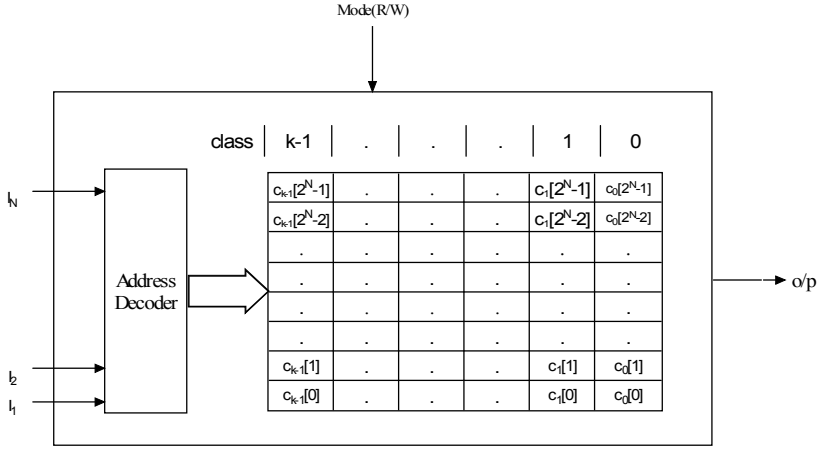

Fig. 4. k-bit RAM. Each column is reserved for one class of data. Each column is connected to one discriminator.

Training traditional neural network requires updating weights in each training iteration. As mentioned earlier, floating-point computations, which are difficult to be implemented in hardware, is required to update weights in high resolutions. Since, the RAM-based neural network is weightless, then the training is very simple to be wholly implemented in hardware.

This can be achieved simply by bitwise writing a '1' to specific tuple locations ("writing" mode is selected).

After training, each column of RAM preserves a one-class bits representation of input patterns. M-RAM banks each of k-column preserve the entire bits representation of k classes.

The RAM-based NN are now ready to be used as a general classifier. The complete architecture of the system proposed in our work is shown in Fig. 5. In this figure, one can see that instead of using k-adders, each of $\log_2 M$ bit width, a set of k-counters $(C_0, C_1, \dots, C_{k-1})$ are used. Each counter can count up to M value. The bit width of each counter should be approximated to the nearest power of two number to the value of $\log_2 M$. All counters work in parallel.

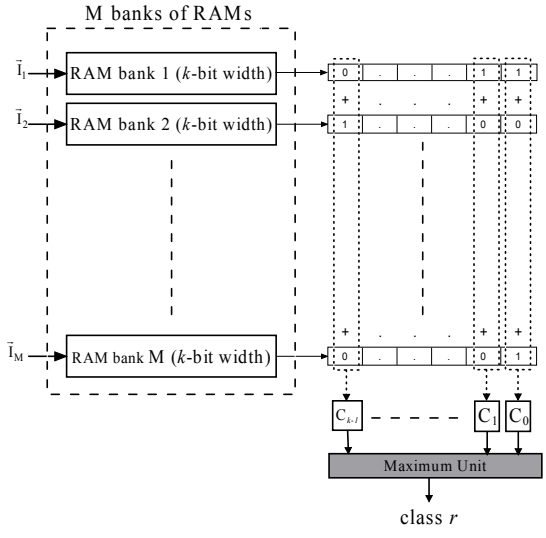

Fig. 5. The complete architecture of RAM-based neural network classifier. M banks of RAMs are reserved for M tuples and k counters are used each for one class. $\overrightarrow{I_1}, \overrightarrow{I_2}, \dots, \overrightarrow{I_M}$ are M addresses each comprises pixel values of one specific region of the input image.

5. FPGA IMPLEMENTATION

As it is noticed in the last section, memory RAMs are the main components of the neural architecture. All referenced trained patterns that are to be classified are stored in RAMs and a huge amount of RAM banks is required to store them.

Present FPGAs are usually made of reconfigurable logic blocks combined with fast access memories (RAM blocks) and high-speed arithmetic circuits. FPGAs have very limited amounts of on-chip RAM. For implementing large RAM, three options are available. These depend on the amount of memory required. The first option is the logic blocks themselves, which can be configured to act like RAMs, but this is usually an inefficient use of the logic blocks which should be left to be used for arithmetic calculations. The second option is the use of the embedded RAM blocks of tens of kilobytes. If the required size is in megabytes, then the external RAM is a better solution. Most of FPGA boards are provided with such huge size of external RAM, which can be operated at a very high bit rate.

If the embedded RAM block size is enough to accommodate the training patterns, then they can be used as the main storage device of the proposed network. Although that M number of RAM banks exceeds the available RAM blocks provided by most of the FPGA models, therefore, in this paper, single address space is shared with all available RAM Blocks (BRAM). Then, and according to the size of BRAM, multiple tuples may share the same BRAM. An interleaving technique is used to address the available BRAMs with single address space (see Fig. 6).

In Fig. 6, it can be noticed that instead of using M tuple each of one separated BRAM, in the above scheme all BRAMs are shared for all tuples.

Now, when the image that is to be trained or tested is applied, M RAM locations are provided by M tuples in parallel. Then, each tuple is applied to the address decoder serially using parallel to serial convertor. The address pattern is now available to determine the location that is to be updated (in training session) or that is to be read (in testing session). This address comprises $(N + \log_2 M)$ bits. It is partitioned into two parts (according to the type of interleaving either low or high order). A number of bits are applied to the multiplexer to determine the BRAM model, and the rest of bits determine the location within the model. One RAM access cycle is required to access each tuple location. For M tuples, M cycles are required.

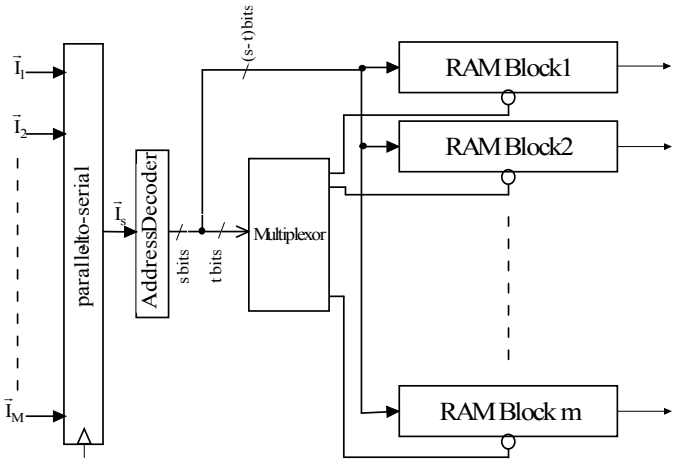

Fig. 6. Serial mapping of an interleaved RAM Blocks. According to the interleaving scheme, the output of the address decoder is divided into two groups of bits: t & $s - t$ depending on the available RAM Blocks.

If the training patterns require to be stored in a RAM of size out of the available RAM blocks, then the on board external RAM can be used as an alternative option.

During the testing phase, the k bits width output of each tuple is distributed among the k available counters. Then each counter updates its value according to the state of the bit that specified to it.

After M clock cycles, the results of all used counters are concurrently buffered onto the input of the maximum unit.

The essential components of the maximum calculation unit that is shown in Fig. 7 are: storage registers, counters, and comparators. These components are controlled to give the final output after a number of clock cycles depending on the number of input registers. The maximum unit searches for the maximum value among all its inputs. The compare unit is based on a sequential search algorithm. A k bits register is provided in the output. The detected class (r) is determined by the position of the bit that set to one among the other zero

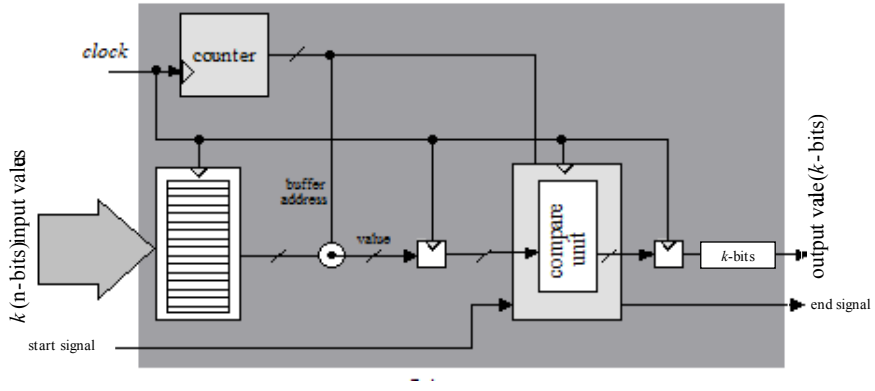

Fig. 7. The maximum unit. k values input, one value output of k bits: one bit is set among other all zeros.

bits.

6. EXPERIMENTAL RESULTS

Some of the MNIST samples are shown in Fig. 8, where ten examples are presented for each handwritten digit. It is clear that each digit occurs in different shapes in terms of rotation, scaling, transformation, and even the font style. Which may lead to ambiguity in some cases during the recognition process?

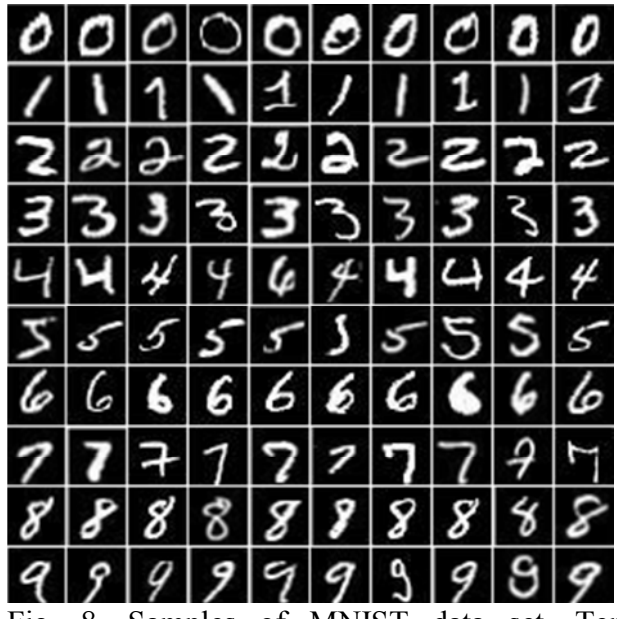

Fig. 8. Samples of MNIST data set. Ten examples are presented for each handwritten digit. It can be realized that each digit may occurs in different shapes in terms of rotation, scaling, transformation, and even the font style.

The best results on MNIST were obtained by training the designed NN with only 6,000 images with test error rate of 7.63%. Training the NN with less number of samples returns more test error rate, which can be interpreted as still not adequate training samples are presented to the NN. Also training the NN with more number of samples returns more test error rate, which can be interpreted as the locations of RAMs became saturated, and hence the NN cannot discriminate between patterns. Results of test error rate as a function of training samples are shown in Fig. 9.

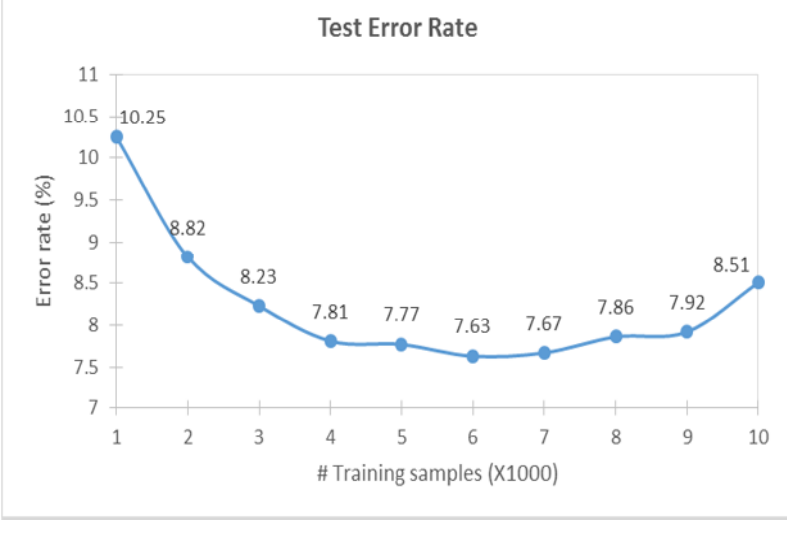

Fig. 9. Test error rate as a function of training samples. The best recognition is obtained when the NN is trained with only 6,000 samples. Whilst training the network with more or less than 6,000 samples yields less recognition rate.

The experiential results have been chosen to describe some features of the hardware model of RAM-based neural network using Stratix IV GX FPGA board.

All the following results are based on the network parameters that gives the maximum recognition accuracy (# tuples= 147).

Table (1) summaries the recourses utilization for the main components of the designed network that is shown earlier in Fig. 5. From the table, one can see that the slices consumption is very small in comparison to the memory required, which comprises the major block of the proposed architecture.

Table 1. Resources Usage

Block	LE	RAM (MB)
k-Counter	70	0
Max unit	61	0
Interleaving components	45	0
Tuples RAM	0	12

Moreover, one can see that the logic elements consumption is very small in comparison to the memory required, which comprises the major block of the proposed architecture. Since that the size of the required RAM is in megabyte, then the third option mentioned in Section 5 is followed. The Stratix IV GX FPGA development board, which is provided on large on board memory, is used as a target FPGA design platform.

The device is provided with a about of 14 M bits of on chip memory that distributed among 1235 and 22 block RAMs of two models, 9 Kbits and 144 Kbits respectively. The size of the on chip memory is small enough to accommodate the required tuples RAM presented in Table (1). Fortunately, the device is provided with a total of 650 M byte of external RAM which can be accessed by FPGA. Only 12 M byte of it is consumed for tuple RAM.

The time required to train the network as a function of training sample in either software based implementation-using MATLAB or hardware based implementation using FPGA is shown in Fig. 10. The computer system that is employed for this work is equipped with an Intel Core2 Due CPU of 2 GHz, 3 MB L2 Cache, 3 GB of DDR2 400MHz physical memory, and running Windows 7 Ultimate 64-bit and Matlab 2009a.

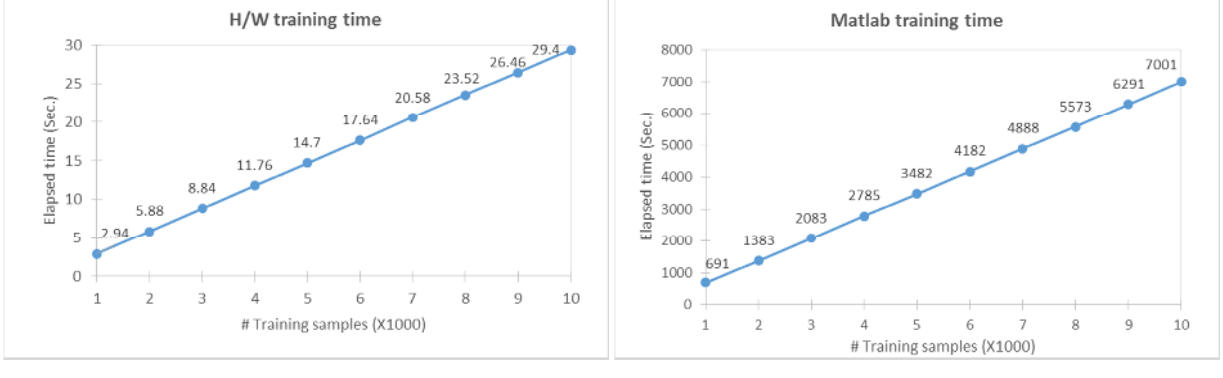

Fig. 10. Elapsed Training Time. Left: the required training time as a function of training samples using FPGA. Right: the required training time as a function of training samples using Matlab.

Although that the system can operate with a maximum frequency of 112.638 *MHz*, but the operating frequency of the FPGA device is $50MHz$. Thus, the processing time depends on

the actual frequency rather than the maximum one. However, from Fig. 10, it is noticed that a considerable speed up of about (237) is achieved even when the device is operated on 50*MHz*. The recognition time of the proposed model is the time of mapping one image into the hardware architecture presented in Fig. 5. For the FPGA model operating in $50MHz$, the recognition time equals to $3\ \mu sec$.

A comparison of the network proposed in this paper with other works in terms of recognition error rates for MNIST data set for handwritten digit recognition is presented in Table (2). In this table, one can see that the error rate of our proposed network is reasonable for the following reasons: the network is trained using few numbers of training samples (6,000 training samples out of 60,000) and are tested using 10,000 samples and gives a relatively moderate error rate; no preprocessing is used, each training sample is presented directly to the available tuples; The invariant property of rotation, scaling and translation are considered in most of handwritten digit recognition systems (such as the Convolutional Neural Networks), in our proposed network, neither of them is considered.

Table 2. Comparison of results with other studies

Classifier	Preprocessing	Training set	Testing set	Test Error Rate (%)
Linear classifier (1-layer NN) [8]	None	60,000	10,000	12.0
Linear classifier (1-layer NN) [8]	Deskewing	60,000	10,000	8.4
RAM-based logical NN	None	6,000	10,000	7.63
Pairwise linear classifier [8]	Deskewing	60,000	10,000	7.6
Convolutional net Boosted LeNet-4 [8]	None	60,000	10,000	0.7
Large convolutional net, unsupervised pretraining [9]	None	60,000	10,000	0.39

7. CONCLUSION

A RAM-based logical neural network with 10 discriminators has been designed and implemented in FPGA. It has been used for the recognition of MNIST handwritten digits. Taking into account the simplicity of the designed NN and the few number of training samples it requires in comparison to other types of NNs, RAM-based NN has achieved very good results. Moreover, it has been shown that the required time for both training and testing phases is relatively low.

However, the lack of the proposed NN can be represented in two things. The first thing is the saturation of memory locations which happens after training large number of samples and this may lead to ambiguity in some cases during the recognition process. The second thing is the relatively large size of required memory for implementing the proposed NN. These two issues may be considered in the future studies.

REFERENCES

[1] I. Aleksander, M. D. Gregorio, F. M. G. França, P. M. V. Lima, and H. Morton, "A brief introduction to Weightless Neural Systems," in *European Symposium on Artificial Neural Networks*, Belgium, 2009, pp. 22-24.

[2] J. Austin, "A review of RAM based neural networks," in *Microelectronics for Neural Networks and Fuzzy Systems*, 1994, pp. 58-66.

[3] N. Nedjah and L. Mourelle, "Reconfigurable hardware for neural networks: binary versus stochastic," *Neural Computing and Applications,* vol. 16, pp. 249-255, 2007.
[4] C. Badue, F. Pedroni, and A. F. D. Souza, "Multi-label Text Categorization Using VG-RAM Weightless Neural Networks," in *10th Brazilian Symposium on Neural Networks*, Brazil, 2008, pp. 105-110.
[5] K. P. Lakshmi and D. M. Subadra, "A Survey on FPGA based MLP Realization for On-chip Learning," *International Journal of Scientific & Engineering Research,* vol. 4, pp. 1-9, 2013.
[6] D. Maliuk and Y. Makris, "A dual-mode weight storage analog neural network platform for on-chip applications," in *IEEE International Symposium on Circuits and Systems*, Korea, 2012, pp. 2889-2892.
[7] T. B. Ludermir, A. d. Carvalho, A. P. Braga, and M. d. Souto, "Weightless neural models: a review of current and past works," *Neural Computing Surveys,* vol. 2, pp. 41-61, 1999.
[8] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE,* vol. 86, pp. 2278-2324, 1998.
[9] M. A. Ranzato, C. Poultney, S. Chopra, and Y. LeCun, "Efficient learning of sparse representations with an energy-based model," in *Advances in neural information processing systems*, 2007, pp. 1137-1144.

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