© Universiti Tun Hussein Onn Malaysia Publisher's Office

IJIE

Journal homepage: http://penerbit.uthm.edu.my/ojs/index.php/ijie

The International Journal of Integrated Engineering

Accelerating Extreme Learning Machine on FPGA by Hardware Implementation of Given Rotation - QRD

Chong Yeam Tan[1], Nordinah Ismail[1,*], Chia Yee Ooi[1], Jin Yong Hon[1]

[1] Embedded System Research Laboratory
MJIIT, Universiti Teknologi Malaysia, Jalan Sultan Yahya Petra, 54100 Kuala Lumpur, MALAYSIA

*Corresponding Author

DOI: https://doi.org/10.30880/ijie.2019.11.07.005
Received 30 May 2019; Accepted 30 July 2019; Available online 10 August 2019

Abstract: Currently, Extreme Learning Machine (ELM) is one of the research trends in the machine learning field due to its remarkable performances in terms of complexity and computational speed. However, the big data era and the limitations of general-purpose processor cause the increasing of interest in hardware implementation of ELM in order to reduce the computational time. Hence, this work presents the hardware-software co-design of ELM to improve the overall performances. In the co-design paradigm, one of the important components of ELM, namely Given Rotation-QRD (GR-QRD) is developed as a hardware core. Field Programmable Gate Array (FPGA) is chosen as the platform for ELM implementation due to its reconfigurable capability and high parallelism. Moreover, the learning accuracy and computational time would be used to evaluate the performances of the proposed ELM design. Our experiment has shown that GR-QRD accelerator helps to reduce the computational time of ELM training by 41.75% while maintaining the same training accuracy in comparison to pure software of ELM.

Keywords: Extreme Learning Machine (ELM), machine learning, field programmable gate array (FPGA), hardware implementation.

1. Introduction

Nowadays, rapid development of technology has caused an increased demand involving machine learning in various applications in order to improve the overall performance of the systems. Many of conventional machine learning techniques are based on feedforward network topologies and gradient-descent backpropagation (BP) algorithm. These algorithms present some drawbacks such as slow convergence and local minimum problems which requires massive iterative steps and consumes a lot of computational time, hence inadequate to meet today's computational extensive application demand [1].

A new machine learning technique which called Extreme Learning Machine (ELM) was proposed in 2006 and proved that it has higher computational speed and better generalization performance than other traditional machine learning techniques [2]. ELM is a Single hidden layer feedforward network (SLFN) which consists of only one hidden layer, one input layer and one output layer in its learning network. Besides that, the linear algebra-based algorithm of ELM and its randomly generated input weight and hidden neuron bias feature converting better performance compared to the steepest descent-based methods used in various machine learning techniques such as Support Vector Machine (SVM) and Artificial Neural Network (ANN) [3]. In the algebra-based algorithm of ELM, Moore-Penrose pseudo-inverse method is used to obtain the minimum norm least squares solution of system equation. Therefore, the computation of Moore-Penrose pseudo-inverse in ELM algorithm is the key to determine the performances of ELM. There are several techniques that can be used to compute the Moore-Penrose pseudo-inverse which include Singular Value Decomposition (SVD), QR decomposition (QRD), LU decomposition and Cholesky decomposition. In [4], The C-based implementation of an embedded Extreme Learning Machine (ELM) algorithm is proposed by using modified Gram-Schmidt QR decomposition (MGS-QRD) and the hardware implementation is then created for FPGA. In this paper, Givens rotation

*Corresponding author: nordinah.kl@utm.my

QR decomposition (GR-QRD) is chosen to be used due to its high stability, less complexity and suitability in parallelized architecture [5].

However, software-based implementation of Moore-Penrose pseudo-inverse consumes a lot of computational time due to the limitations of sequential-based processing of the general purpose processor which only able to perform one arithmetic operation per processing time. Therefore, it is desirable to have a dedicated hardware to accelerate the pseudo-inverse computation. Field Programmable Gate Array (FPGA) is a silicon chip that can be reprogrammed to perform specific functions or applications instead of running through software application [6]. In 2016, [7] proposed an FPGA implementation of a real-time extreme learning machine. Another variant of ELM namely Online Sequential Extreme Learning Machine (OS-ELM) is also implemented using FPGA [8-9]. The features of FPGA such as reconfigurability, high parallelism and low power consumption makes it as the best candidate to accelerate the highly computational algorithm such as pseudo-inverse hence optimizing the overall performance of ELM [10][11].

2. Theoretical Background

2.1 Extreme Learning Machine

Extreme Learning Machine (ELM) is a machine learning algorithm using Single Hidden Layer Feedforward Networks (SLFNs) as the learning network [2]. The learning network of ELM is shown in Fig 1. The number of nodes of input and output layer depends on the number of attribute of the dataset and the number of output class respectively. As shown in Fig.1, ELM consists of only one hidden layer in its learning network. The main concern of the learning network of ELM is the number of hidden neurons we need to choose correctly to optimize the performance of learning algorithm [12].

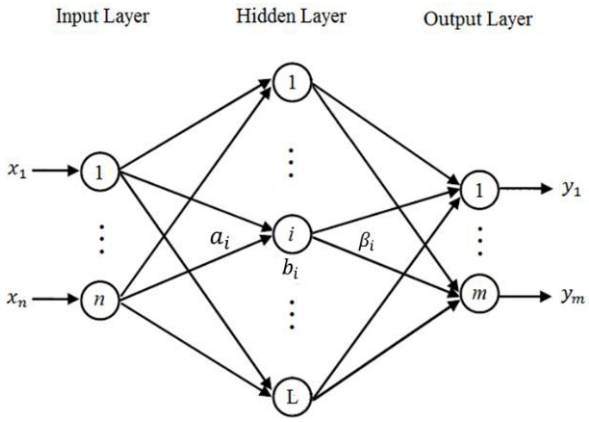

Fig. 1 – Topology of ELM

For ELM with n input nodes, L hidden nodes, m output nodes and a set of N arbitrary distinct samples (,); j = 1,…, N, where $= [\ _1, \ _2, \ldots, \ ]^T \in$ are the inputs vectors and $= [\ _1, \ _2, \ldots, \ ]^T \in$ are the outputs vectors of the training set. The output vectors function of ELM, $= [\ _1, \ _2, \ldots\ ]$, is

$$= \ = \sum_{=1}^{L} \ h( ) = \ \ (\ , \ , \ ) \quad = \quad 1, \ldots, \quad (1)$$

Where $= [\ _1, \ _2, \ldots, \ ]^T$ is the weight vector connecting the ith hidden node and the input nodes, $= [\ _1, \ _2, \ldots, \ ]$ is the weight vector connecting the ith hidden node and the output nodes, and is the threshold or the bias of the ith hidden node. The $(\boldsymbol{a}, \ , \ )$ is an activation function that satisfies ELM universal approximation capability theorem:

$$\mathrm{H} = \begin{bmatrix} h(\ _1) \\ \vdots \\ h(\ _N) \end{bmatrix} = \begin{bmatrix} \ (\ _1, \ _1, \ _1) & \cdots & \ (\ _L, \ _L, \ _1) \\ \vdots & \ddots & \vdots \\ \ (\ _1, \ _1, \ _N) & \cdots & \ (\ _L, \ _L, \ _N) \end{bmatrix}_{\times L} \quad (2)$$

For ELM implementation, there are four main training steps in ELM algorithm [11]:

i. Randomly assign input weight, a_i and hidden node bias, b_i.
ii. Calculate hidden layer output matrix, H.

iii. Compute Moore-Penrose Pseudo-Inverse of matrix H: H^{+}.
iv. Calculate output weight matrix, β: $\beta = H^{+}T$.

where

$$T = \begin{bmatrix} \boldsymbol{t}_1^{\boldsymbol{T}} \\ \vdots \\ \boldsymbol{t}_{\boldsymbol{N}}^{\boldsymbol{T}} \end{bmatrix} = \begin{bmatrix} t_{11} & \cdots & t_{1m} \\ \vdots & \ddots & \vdots \\ t_{N1} & \cdots & t_{Nm} \end{bmatrix}_{N \times m} \tag{3}$$

2.2 Moore-Penrose Pseudo-inverse

Moore-Penrose Pseudo-Inverse computation is a method which can provide a solution to a system of linear equations that lacks a unique solution. This method focuses on finding the minimum (Euclidean) norm solution to a system of linear equations with multiple solutions. For $A \in (, ;)$, the pseudo inverse of matrix A: A^{+} exists if it satisfies the four criteria [14]:

$$A A^{+} A = A \tag{4}$$

$$A^{+} A A^{+} = A^{+} \tag{5}$$

$$(A A^{+})^{*} = A A^{+} \tag{6}$$

$$(A^{+} A)^{*} = A^{+} A \tag{7}$$

where K donated the field of real or complex numbers while A^{*} denoted the Hermitian transpose of matrix A. For full columns rank (linearly independent columns), the pseudo-inverse, A^{+} can be expressed as:

$$A^{+} = (A^{*}A)^{-1} A^{*} \tag{8}$$

While for full rows rank (linearly independent rows), the equation of A^{+} is shown in below:

$$A^{+} = A^{*}(AA^{*})^{-1} \tag{9}$$

2.3 Givens Rotation QR Decomposition (GR-QRD)

Givens rotation is one of the QR decomposition methods that can decompose a $m \times n$ matrix A into [15]:

$$A = \tag{10}$$

Where Q is a $m \times m$ orthogonal matrix while R is a $m \times n$ upper triangular matrix. By using GR-QRD for Moore-Penrose pseudo-inverse computation, each nonzero element of matrix H below the main diagonal is first eliminated into 0 to form an upper triangular matrix R. For each iterative step, only two rows of matrix H are used to compute the zero and all the elements of these rows would change after the zero computation. The equations used to produce matrix R are shown below.

$$\begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} \cdots\cdots & \cdots\cdots \\ \cdots\cdots & \cdots\cdots \end{bmatrix} = \begin{bmatrix} \cdots\cdots \\ 0 \ \cdots\cdots \end{bmatrix} \tag{11}$$

where

$$= \sqrt{^{2} + ^{2}} \tag{12}$$

$$\cos\theta = \frac{1}{\sqrt{{}_1{}^2 + {}_1{}^2}} \tag{13}$$

$$\sin\theta = \frac{1}{\sqrt{{}_1{}^2 + {}_1{}^2}} \tag{14}$$

Besides that, the orthogonal matrix, Q can be obtained by using the inverse of matrix R as shown in following equation:

$$= {}^{-1}\mathrm{H} \tag{15}$$

By using R^{-1} and Q obtained, the Moore-Penrose pseudo-inverse can be computed by using the following equation.

$$\mathrm{H}^{+} = {}^{-1\,T} \tag{16}$$

Where ^{T} is the transpose of matrix Q.

3. Methodology

3.1 Hardware Partition

For the Moore-Penrose pseudo-inverse method used in ELM algorithm, the computation of GR-QRD greatly affects the performance of the ELM in terms of accuracy and computational speed. By implementing the GR-QRD using hardware logic on FPGA, the computational time can be significantly reduced since the tasks can be processed in parallel, thus it leads to much higher speed. Therefore, different aspects and related constraints should be considered in order to design and construct a parallel architecture for GR-QRD implementation [15]. In this paper, Linear Systolic Array (LSA) [17] was proposed as the parallel architecture for the implementation of GR-QRD on FPGA because of the following reason:

1) 100% utilization of processing units can be achieved;
2) Huge circuit resources can be preserved and much larger data can be processed. By using LSA, the number of processing elements (PEs) required to compute GR-QRD can be determined by the following equation:

$$E = \frac{(+ 1)}{2} \tag{17}$$

where n is the number of hidden neurons used in the ELM algorithm. There are two types of PE used in the LSA architecture for implementation of GR-QRD which are boundary node and internal node. Boundary node is the processing unit that computes the sine and cosine function from the hidden layer output matrix H by equation (13), (14) and (15) while internal node is the processing unit that performs the effective rotations to each element of the input matrix by using cosine and sine function computed from boundary node. There is only one boundary node required to compute GR-QRD while $\frac{-1}{2}$ internal nodes are required to perform the effective rotation in LSA architecture.

For designing the PEs in LSA architecture, boundary node requires three multipliers, one adder, one square root and one divider to compute the Givens rotation parameters while four multipliers, one adder and one subtractor are required for internal node to perform the effective rotations. All the mathematic calculations will be computed by using fixed-point arithmetic operation and each arithmetic operation requires 1 clock cycle to compute the output. Fig.2 and Fig.3 illustrate the block diagrams of the boundary node and internal node respectively and also the corresponding clock cycles for each arithmetic operation. The delay blocks in these PEs are the registers that temporarily store the value of r and will then become one of the inputs in the next rotation after a few cycles of delay, T_D where T_D is arbitrated by the number of hidden neuron used in the ELM algorithm.

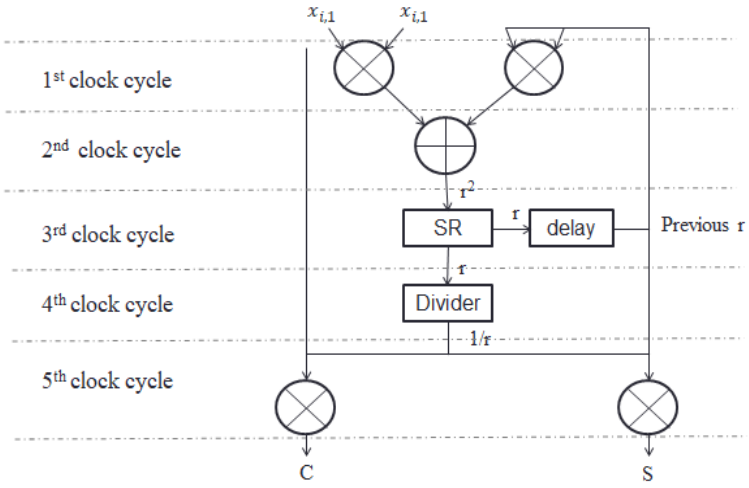

Fig. 2 – Block Diagram of Boundary Node

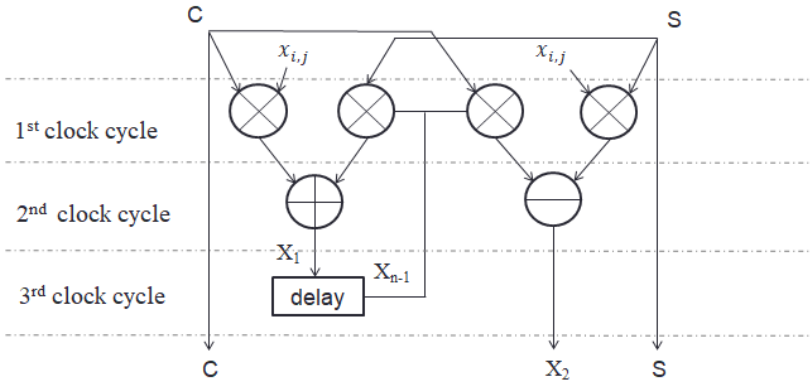

Fig. 3 – Block Diagram of Internal Node

The total circuit resources required for implementing a single boundary node and internal node on the DE2-115 FPGA Development Board were listed in Table 1 and Table 2. The results were obtained through simulation by using Quartus Prime software while the module was written in Verilog HDL.

Table 1 - Total Circuit Resources Requred By Boundary Node

Circuit Resource	Value
Total Logic Elements	2,901
Total Registers	384
Total Pins	264
Embedded Multiplier 9-bit Elements	32
Total PLLs	0

Table 2 - Total Circuit Resources Required By Internal Node

Circuit Resource	Value
Total Logic Elements	2,130
Total Registers	342
Total Pins	194
Embedded Multiplier 9-bit Elements	32
Total PLLs	0

3.2 Software Partition

In this paper, the ELM modules other than GR-QRD were described in C++ to be run on a soft-core processor. Soft processor exists as synthesized netlists incorporated in the FPGA using logic block resources of a particular FPGA and it provides the advantage of flexibility in system designing. The soft-core processor used for this study is NIOS II processor on Altera DE2-115 FPGA Development Board. Fig.4 illustrates the block diagram of software based design of ELM modules on NIOS II processor.

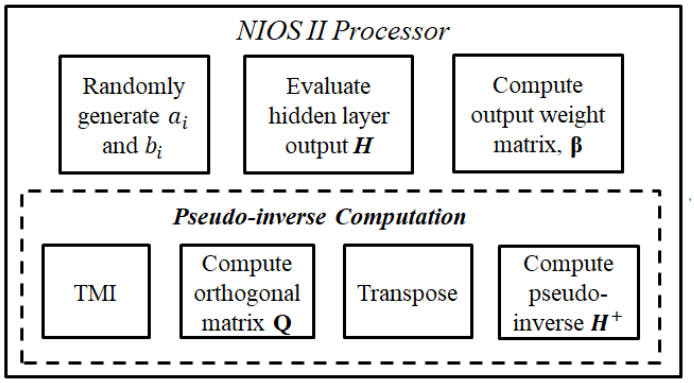

Fig. 4 – Block Diagram of Soft-core Based of ELM Modules

3.3 Hardware Software Co-Design

The hardware partition of GR-QRD accelerator and software partition of ELM modules need to be combined to accelerate the overall performance of ELM algorithm. In order to accomplish the task, Qsys System Integration Tools from Quartus Prime software was used to build the system interconnection between GR-QRD accelerator, NIOS II processor and other modules. Besides that, the data ports required by the GR-QRD accelerator are implemented in the form of Parallel Input/Output (PIO) components for data flowing between block to block.

Fig.5 represents the block diagram of the proposed ELM architecture. Inside the FPGA fabric, it contains all the System-on-A Programmable Chip (SOPC) components including NIOS II processor and the GR-QRD accelerator. The Avalon Bus Interface is used to create the connection between blocks for data transferring and controlling off-chip devices.

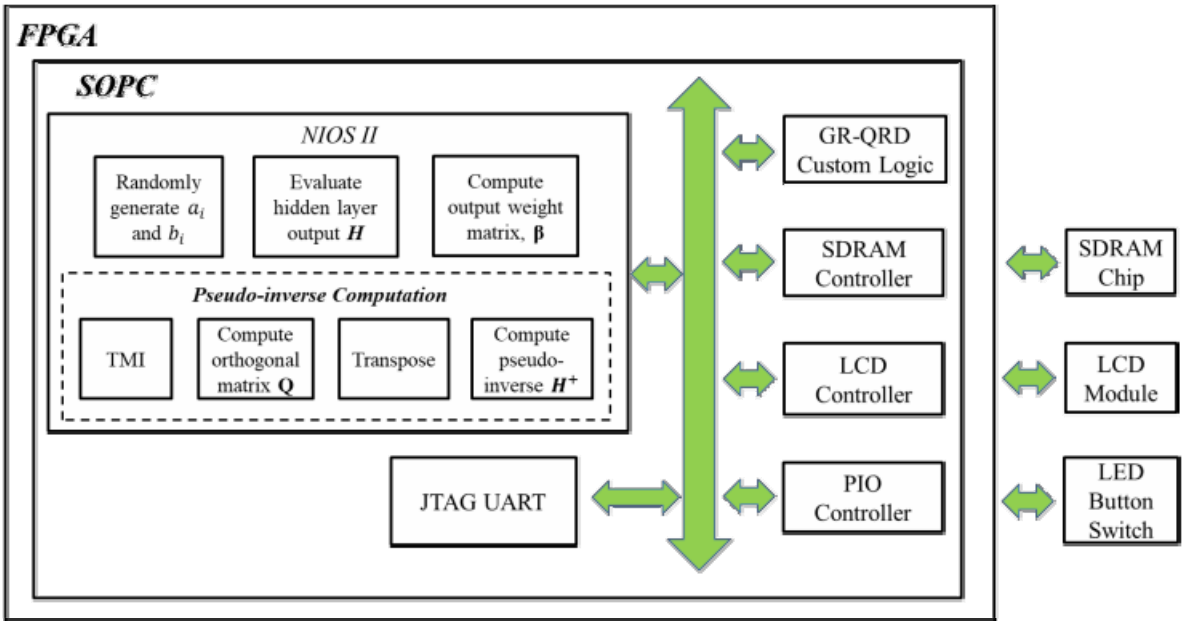

Fig. 5 – Block Diagram of Proposed ELM Architecture

4. Result and discussion

4.1 Performance Evaluation

Pima Indians Diabetes database has been used to appraise the performance of the proposed ELM algorithm. This database is a medical analysis emanated from the Applied Physics Laboratory, Johns Hopkins University, which was established in 1988 [18]. It consists of 768 female residents who are over 21-years old Phoenix, Arizona. The diagnostic, binary-valued variable investigated is whether the resident or patient shows signs of diabetes according to World Health Organization criteria. It is a binary classification dataset where the positive class will be represented by 1 while negative class will be represented by 0. There are 505 and 263 samples for diabetic and non-diabetic patient respectively. This dataset consists of 8 attributes, including pregnant frequency, concentration of plasma glucose, diastolic blood pressure, tickness of triceps skin fold, serum insulin in two hours, body mass index (bmi), diabetes pedigree function and age

The performance of ELM algorithm is affected by the number of the hidden neurons used. In this paper, an analysis was conducted to obtain the optimum number of hidden neurons in order to optimize the performances of the proposed ELM system. Table 3 shows the ELM network parameters used for the analysis and a graph of accuracy against number of hidden neurons has been drawn as shown in Fig 6.

Table 3 - ELM Network Parameters Used

Parameter	Detail
Number of Hidden Layers	1
Number of Input Neurons	8
Number of Output Neurons	2
Activation Function	Sigmoid

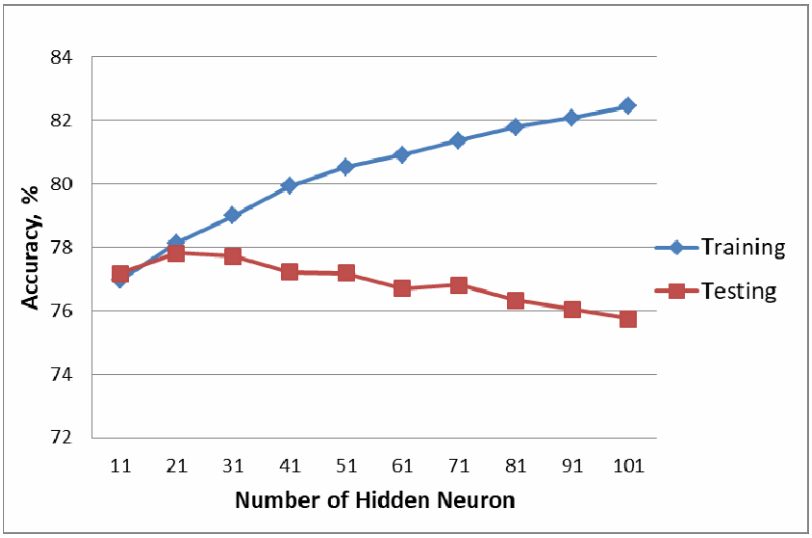

Fig. 5 – Graph of Accuracy against Number of Hidden Neurons

For ELM algorithm, the network size which is too small in hidden neurons will have low training and testing accuracy due to underfitting while the network size which is too large in hidden neurons will also result in low testing accuracy due to overfitting. In our case study, the optimal number of hidden neurons used in the proposed ELM system for diabetes binary classification is 21 at which both training and testing accuracy achieve its highest value.

.

4.2 Comparison of Software Implementation and Hardware-Software Co-Implementation of ELM System

In order to evaluate the time reduction of this hw-sw co-implementation, a fully software-based ELM algorithm was implemented running on NIOS II processor on the FPGA device itself. The source code of this soft-core based ELM design was written in C programming language and implemented on Altera DE2-115 FPGA Development Board. The training accuracy, testing accuracy, training time and testing time for both designs were recorded as shown in Table IV.

Table IV - Performances of Soft-core Based and Combined Hard-core and Soft-core Based ELM Implementation

Type of Implementation	Accuracy (%)		Computational time (s)	
	Training	*Testing*	*Training*	*Testing*
Software-based	77.78	76.04	276	29
Hardware-Software Based	78.12	76.04	158	29

The hw-sw co-design of ELM algorithm in this project has less significant effect on the testing accuracy and testing time because GR-QRD computation is not required in the ELM testing phase. Most importantly, the result shows that the hardware-based implementation of GR-QRD in ELM algorithm helped to reduce the training time from 276 seconds to 158 seconds which accelerates the ELM training phase by 42.75 % while maintaining the same training accuracy.

5. Conclusion

In this study, embedded system architecture for hardware-software implementation of ELM is presented. In ELM algorithm, Givens rotation QR decomposition (GR-QRD) method was proposed to perform Moore-Penrose pseudo-inverse computation. In order to evaluate the performance of the proposed ELM system, a real world dataset, diabetes dataset, was tested. The results show that the proposed ELM system can produce a relatively high accuracy in both training and testing phase.

Besides, the GR-QRD was implemented as fully hardware logic to speed up the computation of ELM algorithm and Linear Systolic Array (LSA) was used as the parallel architecture to implement the GR-QRD accelerator. By comparing the hw-sw co-implementation approach with the fully software-based ELM design, the results show that the hardware-software implementation of ELM with GR-QRD accelerator can reduce the computational time of ELM training phase by 41.75% while maintaining the same training accuracy.

For the current and future work, MGS-QRD can be used to compute the Moore-Penrose pseudo-inverse and parallel computing features can be explored on FPGA to further increase the computational speed. The proper scheduling and pipelining between the modules can significantly speed-up the ELM computation without affecting training and testing accuracy.

Acknowledgement

This work is supported by Universiti Teknologi Malaysia (UTM) Research University Grant with vote number Q.K130000.2543.17H55.

References

[1] JCM Than et al. Lung Disease Classification using GLCM and Deep Features from Different Deep Learning Architectures with Principal Component Analysis, Int. J. of Integrated Engineering Vol. 10 No. 7 (2018) p. 76-89 [2] Huang, G., Huang, G. B., Song, S., & You, K. (2015). Trends in Extreme Learning Machines. Neural Network, 61, 32-48.

[3] Huang, G. B., Zhu, Q. Y., & Siew, C. K. (2006). Extreme Learning Machine: Theory and Applications. Neurocomputing, 70, 489-501.

[4] Yeam, T. C., Ismail, N., Mashiko. K., & Matsuzaki, T. FPGA implementation of extreme learning machine system for classification, IEEE Region 10 Annual. Int. Conf. TENCON, pp. 1868-1873, Dec. 2017.

[5] Golub, G.H., & Van Loan, C. F. (1996). Matrix Computations. London: The Johns Hopkins University Press.

[6] Vazhoth Kanhiroth, Vivek J. (2017). Embedded processors on FPGA: Hard-core vs Soft-core. Masters Theses. 845.

[7] Frances-Villora, J. V., Rosado-Munoz, A., Martinez-Villena, J. M., Bataller-Mompean, M., Fco. Guerroro, J., & Marek, W. (2016). Hardware implementation of real-time extreme learning machine in FPGA: Analysis of precision resource occupation and performance. Computer & Electrical Engineering, 51, 139-156,

[8] Tsukada, M., Kondo, M., & Matsutani, H. (2018) OS-ELM-FPGA: An FPGA-Based Online Sequential UnsupervisedAnomaly Detector. In Proceedings of the International European Conference on Parallel and Distributed Computing (Euro-Par'18) Workshops, pages 518–529, Aug 2018.

[9] Guerrero-Martinez, J.F., Frances-Villora, J.V., Bataller-Mompean, M., Barrios-Aviles, J., Rosado-Muñoz, A. (2018). Moving Learning Machine towards Fast Real-Time Applications: A High-Speed FPGA-Based Implementation of the OS-ELM Training Algorithm. Electronics, 7, 308.
[10] Dubey, R. (2009*)*. Introduction To Embedded System Design Using Field Programmable Gate Arrays. Girona, Spain: eStudio Calamar S.L.
[11] Noor Huda Ja'afar, Afandi Ahmad, Abbes Amira. Rapid Prototyping of Three-dimensional (3-D) Daubechies with Transpose-based Method for Medical Image Compression, International Journal of Integrated Engineering, Vol. 4 No. 3 (2012) p. 26-34.
[12] Huang, Y. W. (2012). Hidden Node Optimization for Extreme Learning Machine. AASRI Procedia, 3, 375-380. [13] Huang, G. B., Zhu, Q. Y., & Siew, C. K. (2004). Extreme Learning Machine: A New Learning Scheme of Feedforward Neural Networks. 2004 IEEE International Joint Conference on Neural Networks. 25-29 July 2004. Budapest, Hungary.
[14] Serre, D. (2011). Matrices: Theory and Applications, Paris: Dunod.
[15] Erdos, L. (2000). QR factorization revisited. Linear algebra for MATH2601 Numerical methods, 30-44.
[16] Culler, D., Jaswinder, P. S., & Anoop, G. (2007). Parallel Computer Architecture: A Hardware/Software Approach. San Francisco, CA: Morgan Kaufmann Publishers.
[17] Lightbody, G., Walke, R., Wood, R., & Mccanny, J. (2000). Linear QR Architecture for a Single Chip Adaptive Beamformer. Journal of VLSI Signal Processing Systems, 24, 67-81.
[18] Blake, C. L., & Merz, C. J. (1998). UCI Repository of machine learning databases.