Global Journal of Engineering and Technology Review

Journal homepage: www.gjetr.org

Global J. Eng. Tec. Review 3 (2) 30 - 38 (2018)

Hardware and Software Implementation of Artificial Neural Network in Hybrid CPU-FPGA Platform

Lim Chun Ming, Zaini Abdul Halim, Tan Earn Tzeh

School of Electrical & Electronic Engineering Universiti Sains Malaysia Penang, Malaysia

ABSTRACT

Artificial neural network (ANN) has been widely used in many applications and has been started to be implemented in embedded system. Recently hybrid platform like Altera DE1-SOC that contains both processor and FPGA had been introduced. When using this type of platform, artificial neural network can be either implemented in processor using software implementation or in FPGA using hardware implementation. Analysis should be done to see whether processor or FPGA is a better choice for the ANN. This paper presented the framework for implementation of ANN in processor and FPGA of Altera DE1-SOC and analyzed the efficiency of implementation of ANN in processor and in FPGA in terms of accuracy, execution time and resources utilization. Several multilayer perceptron (MLP) models with different number of inputs, number of hidden neurons and types of activation function had first been trained in MATLAB and after that, these trained models had been implemented in both processor and FPGA of Altera DE1-SOC. Experiments had been carried out to test and measure the performance of these MLP models in processor and FPGA. After comparing output result with ANN that run in MATLAB and computing the mean squared error (MSE), results showed that the ANN in processor has 100% accuracy and ANN in FPGA has minimum MSE of 7.3 x 10-6. Meanwhile ANN in FPGA is 20 times faster than ANN in processor. Therefore, if accuracy is main priority and execution time is not so important in a system, ANN is suggested to be implemented in processor. However, if execution time of ANN must be fast like less than microsecond in a system, ANN is suggested to be implemented in FPGA.

Type of Paper: Empirical

Keywords: Artificial neural network (ANN), Multilayer Perceptron (MLP), Altera DE1-SOC, Processor, FPGA

1. Introduction

Artificial neural network (ANN) is a computational model that is based on the interconnection of the neuron in the nervous system of human brain. ANN can be trained using learning algorithm and data set to solve problems. ANN can be used for pattern recognition, classification, prediction etc.

* Paper Info: Revised: February 18 2018

Accepted: April 4, 2018

* Corresponding author:

E-mail:zaini@usm.my

Affiliation: School of Electrical & Electronic Engineering Universiti Sains Malaysia

Today, artificial neural network has been widely used in various field of application like photovoltaic system [1], [2], medical diagnosis system [3], stock market prediction [4] etc. Most of the ANN implementations are PC based simulation software model [5], [6] which is relatively slow compared to hardware implementation type of ANN [7]. Besides, many of the ANN implementation in medical diagnosis system is not portable [8], [9]. Thus, more research should be done to study the hardware implementation of ANN model to increase the portable neural network solution in the market.

When it comes to portable embedded platform, CPU, ASIC, FPGA are available and recently, hybrid CPU-FPGA platform which integrate processor together with FPGA has been introduced in the market. In October 11, 2011 Altera Corporation introduces a new SOC FPGAs device that integrating a dual core ARM Cortex-A9 MPCore Processor with 28-nm Cyclone V and Arria V FPGA fabric [10]. High-level management functionality of processor and real-time, high flexibility, extreme data processing FPGA becomes one of the powerful embedded platforms.

Since hybrid CPU-FPGA platform has both processor and FPGA, sometimes one may have to consider which one to use for the application. FPGA with parallel feature has higher data processing rate, but design with high performance that use up a lot of resources may have a higher cost. [11]

It is important to have a detail analysis on the performance of application like artificial neural network on processor and FPGA based on several aspects like accuracy, execution time, resource utilization or cost to justify whether implementation on processor or FPGA is better for the artificial neural network.

This paper presented the implementation of multilayer perceptron neural network (MLP) in Altera DE1-SOC platform. Five MLP models with different structure, for example different number of input feature, different number of hidden neuron, and different type of activation function had been implemented on FPGA and Arm Cortex A9 of DE1-SOC separately. Besides, this paper also analyzed the performance of each model on FPGA and processor based on accuracy, execution time and resource utilization. This study has given a clear comparison of performance on the implementation of ANN models on processor and FPGA in hybrid CPU-FPGA platform like Altera DE1-SOC. The rest of paper is organized as follow: section II discusses the related works, section III is methodology, section IV is results and discussion, section V is conclusion and last section is future works.

II. Related Works

OZDEMIR, A.T. and DANISMAN, K. [12] had done a comparative study on two artificial neural network architectures. First model was a 32-bit floating point FPGA based model. Second model was a 16-bit fixed point FPGA based model. Both architecture had a sized of 8 input neurons, 2 hidden neurons and 1 output neuron. First model achieved accuracy of 97.66% and second model achieved accuracy of 96.54%. However, resource utilization of second model was lower than the first model. Classification time for the first model is 1.07us and 1.01us for second model.

K.V.Ramanaiah, et al.[7] had proposed a parallel multilayer perceptron neural network architecture that implemented on Xilinx Virtex family FPGA device. The proposed hardware architecture was being implemented in two phases. First phase includes the training of neural network in MATLAB program.

Second phase was hardware implementation of the trained neural network on the Xilinx FPGA device. The size of the neural network was 16-4-16. After experimental analysis, the architecture on FPGA was found to be operated at the maximum speed of 138MHz, occupying 3% of the resources and consuming a total power of 55.285mW.

Andre L.S Braga, et al. [13] had done an interesting research on analyzing the implementation of ANN models in both FPGA by hardware description language (HDL) and in Xilinx MicroBlaze

embedded soft microprocessor by software approach in terms of resources utilization, accuracy, speed and power consumption. Author had created 3 ANN models for FPGA with VHDL and 4 models for MicroBlaze soft microprocessor. Results showed that MicroBlaze design had better accuracy than FPGA designs but FPGA designs were way faster than MicroBlaze designs.

In overall, there were quite some of the researchers did research on the implementation of artificial neural network on FPGA or embedded soft microprocessor. Most of the findings showed that many of the researchers only focus on the FPGA implementation of ANN and they did not analyze and compare the performance between implementation in processor and FPGA. Although researcher Andre L.S Braga [13] did have some analysis and compare the performance of implementation of ANN model on FPGA and implementation of ANN model using software approach on embedded soft microprocessor in terms of resources utilization, execution time and accuracy, however there is still some improvement that can be made. The improvement is a more detailed analysis that include some other important factors like number of input, number of hidden neurons and type of activation function in testing of performance. Besides, there is still lack of detailed analysis of ANN on hybrid platform like Altera DE1-SOC that consists of hard processor and FPGA. It would be better if several ANN models that includes previously mentioned factors can be tested on processor and FPGA of hybrid platform like Altera DE1-SOC.

III. Methodology

A. Implementation Platform

Altera DE1-SOC board had been used for ANN implementation. Altera DE1-SOC is a hybrid platform that integrates ARM based hard processor system that consists of processor, memory interfaces and peripherals with industry leading FPGA fabric using high bandwidth interconnects. Processor of hard processor system (HPS) is 800Mhz dual-core ARM Cortex-A9 MPCore processor with 1GB DDR3 SDRAM using 32bit data bus. Processor executes program or software in a sequential manner. Linux was used as operating system for ARM processor. ANN had been implemented in processor using C Programming language. While FPGA is Altera Cyclone 5 SE 5CSEMA5F31C6N device with 64MB SDRAM using 16bit data bus. FPGA is able to process data in parallel. ANN in FPGA had been implemented using Verilog Hardware Description Language.

B. Implementation of Neuron

The Artificial neural network used in this project was a multilayer perceptron with 3 layers: input layer, hidden layer, output layer. Multilayer perceptron is a type of feedforward neural network where information flows through layers in one direction only. Unit in each layer is called neuron. Neuron in layer is connected with certain weights to all neurons in next layer. All neurons except the input layer neuron are considered as processing elements. Figure 1 shows the structure of a multilayer perceptron neural network.

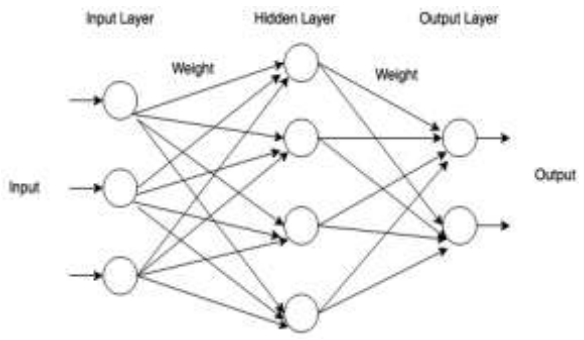

Figure 1: Multilayer Perceptron Model

Figure 2 shows the structure of artificial neuron where inputs of the neuron will be multiplied by respective weights. After that all products will be summed together with bias and go through the activation function. Activation function will then define the output based on the inputs.

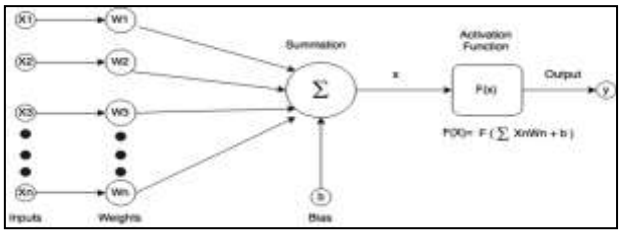

Figure 2: Artificial Neuron Model

For the implementation of neuron in processor, program like using a loop to keep multiplication of inputs with weights until all inputs have been multiplied had been used. After that the sum will be added by bias and applied in activation function. 64 bits double precision floating point had been used as data type for inputs, weights and bias in the C program of ANN. Figure 3 shows the total program flow chart of the ANN. Processing of neurons will be carried out one by one. After all neurons in hidden layer have been processed, processing of the neurons in output layer will be carried out one by one until getting the final output.

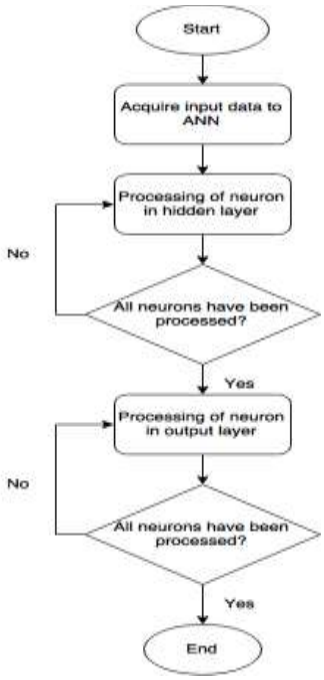

Figure 3: Overall program flow chart for ANN.

For the implementation of neuron in FPGA, multiplier was used to multiply inputs and weights one by one at each clock. The results were then feed to accumulator to be summed up together. At the end, accumulated result will be added with bias. Added result was then sent to activation function to obtain the output. Figure 4 shows the block diagram for implementation of neuron in FPGA.

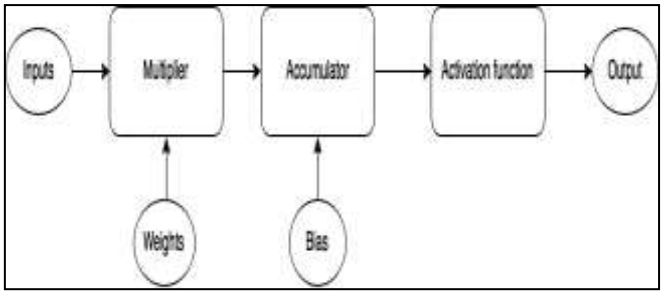

Figure 4: Block diagram for implementation of neuron.

Figure 5 shows the overall block diagram for FPGA implementation of ANN. Each neuron in layer will operate in parallel and simultaneously during each clock. After neurons in hidden layer finish processing, enable signal will be sent to neurons in output layer for operation.

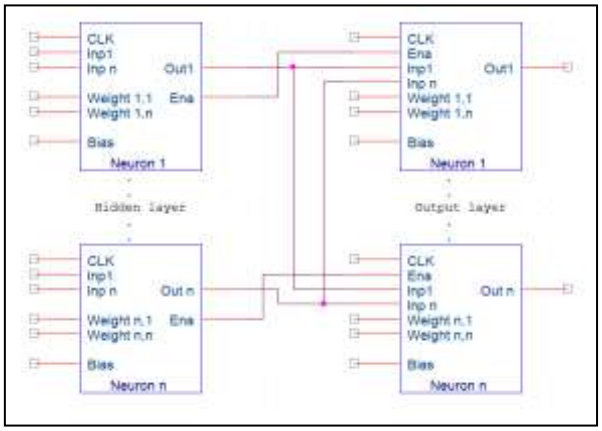

Figure 5: Overall block diagram for ANN in FPGA

Data format used in hardware implementation was 16-bits fixed point where 10 bit for fractional part, 5 bit for integer part and 1 bit for sign part as shown in Figure 6.

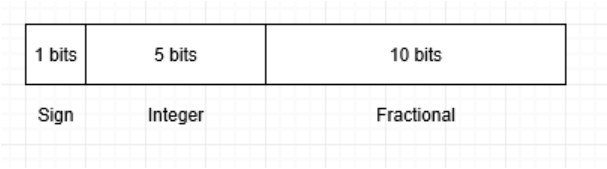

Figure 6: Data format for hardward implementation

For the negative numbers, signed magnitude representation was used where if sign bit is 1, it is a negative number and if the sign bit is 0, it is a positive number. Magnitude bit remains the same or unchanged for positive and negative number. Sign magnitude format is used because it is a simple concept and ease of understanding and implementation.

Implementation of Activation function

The Activation function of hidden layer was tangent sigmoid function and the activation function of output layer was either tangent sigmoid function or pure linear function. Pure linear function with the equation f(n)= n can be easily implemented in software by just multiply input by 1. And in hardware this function block will just feed input into output without operation. For implementation of

hyperbolic tangent sigmoid function with equation of 2/(1+e-2n)-1, in software, this mathematical equation can easily be described using C/C++ programming language. However, for hardware implementation of this non-linear function will be difficult due to the exponential nature. Several methods can be used to realize this non-linear activation function. Method used was Piecewise linear approximation method.

Piecewise linear approximation is a method to construct a function to fit the non-linear function using several segments of linear function for certain interval. After using piecewise linear approximation method in MATLAB, non-linear tangent sigmoid function will become several linear functions or piecewise linear function. Figure 7 shows the piecewise linear function derived from

$$f(x) = \begin{cases} -1 & , x < -3.0 \\ 0.01688x - 0.94441 & , -3.0 \le x < -2.5 \\ 0.04517x - 0.87368 & , -2.5 \le x < -2.0 \\ 0.11775x - 0.72851 & , -2.0 \le x < -1.5 \\ 0.28710x - 0.47448 & , -1.5 \le x < -1.0 \\ 0.59895x - 0.16264 & , -1.0 \le x < -0.5 \\ 0.92423x & , -0.5 \le x < 0.5 \\ 0.59895x + 0.16264 & , 0.5 \le x < 1.0 \\ 0.28710x + 0.47448 & , 1.0 \le x < 1.5 \\ 0.11775x + 0.72851 & , 1.5 \le x < 2.0 \\ 0.04517x + 0.87368 & , 2.0 \le x < 2.5 \\ 0.01688x + 0.94441 & , 2.5 \le x < 3.0 \\ 1 & , x \ge 3.0 \end{cases}$$

Figure 7: Piecewise linear function for tangent sigmoid

D. Experimental Setup

Five MLP models with different number of inputs, number of hidden neurons and type of activation function had first been trained in MATLAB using data set provided by MATLAB, after that trained network had been built and tested in processor and FPGA. Testing had been carried out to measure the performance of several MLP models in processor and FPGA in terms of accuracy, execution time and resources utilization. MLP model 1 has the size of 3-3-2 with tangent sigmoid in both hidden and output layer. MLP model 2 has the size of 3-10-2 with also tangent sigmoid in both hidden and output layer. MLP model 3 has the size of 3-3-2 with tangent sigmoid in hidden layer and pure linear function in output layer. MLP model 4 has the size of 10-3-2 with tangent sigmoid in both layer and MLP model 5 has the size of 3-6-2 with also tangent sigmoid in both hidden and output layer.

For measuring the accuracy of ANN in processor, program had been uploaded to processor and in the Linux terminal, data had been inserted and result had been shown in the terminal. 20 data had been used to test the accuracy. The result from processor had been compared with result from MATLAB and MSE with the equation (1) will be computed. The lower the MSE, the better the accuracy.

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (y - \hat{y})^2 \quad (1)$$

where n = total data, y= output from MATLAB and ŷ= output from ARM processor

For measuring the total time used by ARM processor for MLP model, a test program had been uploaded to the ARM processor. In the test program a library called ‘time.h’ of C programming was utilized. In the beginning of the test program, input data will be entered in the Linux terminal. After

that a clock will be initialized, timer start counting every 1 micro second and ANN program start. Since the ANN program is very small, and it is difficult to measure and show the time of this small program, the ANN program will be executed 100000 times by using for loop. After the ANN program has been executed, timer will stop the counting. The total time used by the ANN program is computed by dividing total clock with clock per second (1 Mega clocks). Measured time will be shown in terminal. This time is the total execution time for 100000 ANN process.

For measuring the accuracy of ANN in FPGA, test bench was run in Altera ModelSim to simulate the output and the rest of steps is the same as measuring accuracy of ANN in processor. To measure the execution time of ANN in FPGA, total number of clock used by the ANN had been measured in ModelSim Altera using simulation. After that, timing simulation or time quest analyzer had been run in Quartus software to simulate the total propagation delay of the ANN design and compute the maximum operating frequency (Fmax) that the ANN design can use. After that the total execution time of ANN model had been calculated by multiplying total number of clocks with time per clock (maximum frequency). To measure the total FPGA hardware resource used by ANN design, after functional verification, the Verilog design had been compiled in Quartus software and logic synthesis had been carried out. In the end, compilation report had been generated, and information regarding the total hardware resources usage had been shown in the report.

Iv. Results & Discussion

After conducting the experiments and compared the result with MALAB, all MLP models in ARM processor have zero MSE or 100% of accuracy. On the other hand, models in FPGA have minimum MSE of 7.3 x 10-6 and maximum MSE of 3.9 x 10-4. Table I shows the comparison of accuracy in MSE. Use of 64 bits double precision floating point as data type in implementation might be the reason of good accuracy for ARM processor. While FPGA has lower accuracy might because the 16 bits fixed point data type. Besides the piecewise approximation method for tangent sigmoid might also lower the accuracy of the ANN in FPGA. In overall, ARM processor is better than FPGA in terms of accuracy for implementation of ANN.

TABLE I: Comparison of Accuracy in MSE

MLP models	MSE in ARM processor	MSE in FPGA
Model 1	0	7.3×10^{-6}
Model 2	0	3.9×10^{-4}
Model 3	0	8.0×10^{-5}
Model 4	0	2.1×10^{-4}
Model 5	0	1.5×10^{-4}

Table II shows the comparison of execution time of all models between ARM processor and FPGA. The execution time of MLP models in ARM processor is longer than in FPGA. The average execution time in ARM processor is around 20 times longer than in FPGA. It might because of the parallel processing in the FPGA that allows faster processing for ANN. The maximum time for model in ARM processor is 5.7 microseconds and minimum time is 1.7 microseconds. While the maximum time for model in FPGA is 0.15 microseconds and minimum time is 0.079 microseconds. In overall, implementation of ANN in FPGA has faster execution time than in ARM processor.

TABLE II: Comparison of Execution Time

MLP models	Execution time in ARM processor	Execution time in FPGA

	(microseconds)	(microseconds)
Model 1	2.2	0.091
Model 2	5.7	0.150
Model 3	1.6	0.079
Model 4	3.0	0.140
Model 5	3.7	0.110

Table III shows the hardware resources usage of all MLP models in FPGA. The maximum usage of resources for model is 13 % and minimum usage of resources for model is 4 %. Small size of ANN like model 1 and model 3 uses only a little of hardware resources. As size of ANN increases, like model 2, the hardware resources usage also increases. Each model consumes less than 15% of total hardware resources and there is still enough of hardware resources for implementation of other system or function.

TABLE III: Comparison of Hardware Resources Usage

MLP Models	Hardware Resources Usage (%)
Model 1	5
Model 2	13
Model 3	4
Model 4	7
Model 5	11

Thus, the implementation of the ANN in FPGA is better than the implementation of ANN in ARM processor in terms of execution time. Lastly, each MLP model in FPGA uses less than 15% of total resources in FPGA. There are still enough resources for implementation of other system and function. In a conclusion, if accuracy of ANN is main priority and execution time is not so important in a system, ANN is suggested to be implemented in ARM processor as it has 100% of accuracy and execution time of ANN in ARM processor is still around microsecond. However, if accuracy is not so important and execution time is main priority or must be less than microsecond, ANN is suggested to be implemented in FPGA as it has extremely faster execution time due to parallel processing.

V. Conclusion

This paper has presented the framework for implementation of ANN in ARM processor and FPGA of Altera DE1-SOC and has analyzed the efficiency of implementation of artificial neural network in processor and FPGA in terms of accuracy, execution time and resources utilization has also been studied. From the conducting experiments and testing, after comparing the result of ANN in ARM processor and FPGA with result of ANN in MATLAB and computing the mean squared error (MSE), implementation of ANN in ARM processor has the same result with MATLAB or 100% of accuracy, while the ANN in FPGA has minimum mean squared error of 7.3 x 10-6. From the execution time testing, results show that the average time for ANN in ARM processor is in microsecond, while the average time for ANN in FPGA is less than microsecond. The execution time of ANN in FPGA is 20 times faster than ANN in ARM processor.

If accuracy is the main priority and execution time of artificial neural network is not so important in a system, artificial neural network is highly suggested to be implemented in processor as implementation of ANN in processor has 100% of accuracy and the time also still in around microsecond. However, if accuracy of ANN is not important, and the system requires faster execution time, for example less than microsecond, the ANN is suggested to be implemented in FPGA as ANN in FPGA is 20 times faster than in processor and the execution time is less than microsecond. In overall, this study is important as it can give a clear understanding of efficiency of implementation of artificial neural network in processor and FPGA of hybrid platform Altera DE1-SOC.

Vi. Future Works

The future work is to implement the ANN in both processor and FPGA using hardware software co-design method like hardware software partitioning. After that, this type of implementation will be compared with the implementation of ANN in processor only and the implementation of ANN in FPGA only.

Acknowledgment

The authors would like to thank the School of Electrical & Electronic Engineering, University of Science Malaysia for supporting this research under Grant No 1001/PELECT/814266.

References

[1] Vaz, A.G.R., Elsinga, B., van Sark, W.G.J.H.M. and Brito, M.C., 2016. An artificial neural network to assess the impact of neighbouring photovoltaic systems in power forecasting in Utrecht, the Netherlands. Renewable Energy, 85, pp.631-641.

[2] Chine, W., Mellit, A., Lughi, V., Malek, A., Sulligoi, G. and Pavan, A.M., 2016. A novel fault diagnosis technique for photovoltaic systems based on artificial neural networks. Renewable Energy, 90, pp.501-512.

[3] Antony, A., Ramesh, A., Sojan, A., Mathews, B. and Varghese, M.T.A., 2016. Skin Cancer Detection Using Artificial Neural Networking. Skin, 4(4).

[4] Sathe, S.S., Purandare, S.M., Pujari, P.D. and Sawant, S.D., 2016. Share Market Prediction using Artificial Neural Network. International Education and Research Journal, 2(3).

[5] Kaur, T., Kumar, S. and Segal, R., 2016, January. Application of artificial neural network for short term wind speed forecasting. In Power and Energy Systems: Towards Sustainable Energy (PESTSE), 2016 Biennial International Conference on (pp. 1-5). IEEE.

[6] Artrith, N. and Urban, A., 2016. An implementation of artificial neural-network potentials for atomistic materials simulations: Performance for TiO 2. Computational Materials Science, 114, pp.135-150.

[7] K. V. Ramanaiah and S. Sridhar, 2015. Hardware implementation of artificial neural networks. I-Manager's Journal on Embedded Systems 3(4), pp. 31-34.

[8] M. K. Gautam and V. K. Giri, "A Neural Network approach and Wavelet analysis for ECG classification," 2016 IEEE International Conference on Engineering and Technology (ICETECH), Coimbatore, 2016, pp. 1136-1141.

[9] Begum, R. and Ramesh, M., 2016. Detection of Cardiomyopathy using Support Vector Machine and Artificial Neural Network. International Journal of Computer Applications, 133(14), pp.29-34.

[10] S. Gabriel, "Altera Introduces SoC FPGAs: Integrating ARM Processor System and FPGA into 28-nm Single-Chip Solution," Altera Corporation, 2011. [Online]. Retrieved from https://www.altera.com/about/news_room/releases/_2011/products/nr-soc-fpga.html. [Accessed 1 3 2017].

[11] G.Li, J.Feng, C.Wang, J.Wang, "Hardware/Software Partitioning Algorithm Based On The Combination of Genetic Algorithm and Tabu Search," Engineering Review, vol. 34, no. 2, pp. 151-160, 2014.

[12] ÖZDEMİR, A.T. and DANIŞMAN, K., 2015. A comparative study of two different FPGA-based arrhythmia classifier architectures. Turkish Journal of Electrical Engineering & Computer Sciences, 23(Sup. 1), pp.2089-2016.

[13] A. L. S. Braga, C. H. Llanos, D. Göhringer, J. Obie, J. Becker and M. Hübner, 2010. Performance, accuracy, power consumption and resource utilization analysis for hardware / software realized Artificial Neural Networks, 2010 IEEE Fifth International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA), Changsha, pp. 1629-1636.