Volume 5, Issue 8, August 2015 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

FPGA Implementation of A Neural Network for a Real-Time Velocity Selective Recording Using Multi-Electrode Cuffs System

Assad I. K. Al-Shueli
Biomedical Engineering Department, College of Engineering,
Thi-Qar University, Iraq

Abstract— This This paper validates and exams the improvements of the system for obtaining velocity spectral information from electroneurogram (ENG) recordings using multi-electrode cuffs (MECs) in hardware term. The present study adopts a fundamentally non-linear velocity classification approach based on a type of artificial neural network (ANN). The present paper deals with the detection and validation of velocity selective recording (VSR) for sorting of APs and the able to distinguish the activity from single fibre from the signals recorded by MEC. This study tackles two considerable issues: the hardware ability for function in real-time for long term implantation; and high performance characteristics such as unsupervised adaptive (automatic classification and detection), fast detection and accurate selectivity. Hence, these challenges have been dealt with ANN signal processing and the acceleration achievable with FPGAs.

This work describes the implementation time delay neural network(TDNN) using FPGA board which is established for VSR method. The design procedure includes investigation different types of implementation strategies for the transfer function, each neuron, each layer in the TDNN and the complete system. Two different approaches for implementation the transfer function have been chosen and evaluated for this system, which are Piecewise linear(PWL) and nonlinear approximation. The results demonstrate the nonlinear transfer function approximation can save about 55% from the silicon area compare with linear approach. Also, the mean squared error (MSE) in nonlinear approach is about ten times less than PWL method. As a result, the nonlinear implementation strategy has been employed in TDNN implementation. For the whole system implementation also two different procedures are used, a sequential and parallel implementation algorithms have been selected for this task. The results demonstrate that the sequential implementation method requires less silicon area compare with parallel approach by 38% and the power consumption is only 25.56 mWatt. This is a significant improvement have been achieved to reduce the consumed power and area which are very important factors for implanted integrated circuits applications.

Keywords— Biomedical signal processing, Biomedical transducers, Microelectronic implants, Neural prosthesis, Artificial neural networks, FPGA

I. INTRODUCTION

One component of neuroprosthetic systems is the neural input through which information is transformed from the physiological to the artificial. Tripolar recording from peripheral nerves cannot generally extract much of the information in the neural traffic, so we and others have been investigating velocity selective recording (VSR) since it allows the possibility of increasing the information extracted from peripheral nerves (electroneurogram-ENG) by carrying out a spectral analysis in the velocity domain [1-13]. The resulting spectrum shows not only the direction of action potential (AP) propagation (afferent or efferent) but also provides a measure of the differential level of excitation of the fibre populations in the nerve. Since there is a well-established relationship between AP propagation velocity and fibre diameter that is approximately linear for myelinated fibres, VSR provides a method for assessing the level of activity in nerve fibre populations of different diameter as well as establishing the direction of propagation. This information potentially allows more information to be extracted from an intact nerve for use in applications requiring sensory feedback such as neuroprostheses [5-9].

During the last few decades, the ANNs have become most interest area of research and have influenced many sectors of industry. The notion of the ANNs and their types and electronics applications for commercial usage are highlighted in [5], [6]. These applications show the ability of ANNs to solve the challenges due to the nonlinear nature of ANNs; they have become an important part of classification and detection of neural recording applications [7–9].

This work presents the implementation for conventional and ANN approach using a field programmable gate array (FPGA), the ANN is used to improve the selectivity and separate neural activity based on their conduction velocity for online application. In addition, the hardware implementation challenges in this approach have been tackled in this paper such as time consuming problem hardware area and power consumption.

A simulation and an experimental comparison results have been presented in this study to show the differences between these approaches. These results display that the ANN method stills efficient and active in selectivity term even with 2 % accuracy reduction in hardware results compare with the simulation results. Also, the sequentially computation implementation method for hardware implementation improves the system performance by reduction the number of logic

gates required, which is achieved by minimizing the number of inner multiplications used for this strategy. Consequently, this process led for further enhancement in hardware size and power consumption by 38% compare with parallel computation implementation approach.

II. IMPLEMENTED THE ANN APPROACH ON FPGA.

In order to study and validate VSR recording for detection and spike sorting performance in real time environment a hardware implementation has been presented in this section. A comparison study has been used for examination the hardware performances for the study cases. The hardware implementations have been configured using FPGA board for both approaches.

Many studies have been reported on implementation the ANNs on FPGAs [10], [11] for neural recording applications. In fact, the FPGA have been selected for hardware implementation and system reconfigurable due to their advantages [12], [13] such as:

- Fast prototyping implementation fore different type of ANN.
- Ability to reconfigure and generate multi-number of ANN designs which can selected later for testing.
- Capability of current improved FPGAs to reconfigure at run time; this dynamic reconfiguration lead for improvement in the density term.

However, the implemented designs on FPGAs have more computing power and hardware size than alternative options such as the ASICS [14]. But, this large amount of power is not necessary for implementation the ANNs prototype purposes because that can be turned on CMOS technologies such as in [15].

A. ANN Implementation Components

In general, the ANNs are consisting from several components; which are multiplication, summation and transfer function as shown in Fig. 1. Typically, the ANNs are containing from one or more layers and each layer has one or more neurons connected in different arrangement type. The inputs elements within individual neuron are multiplied with weights and added together with bias and then supplied to the transfer function. There are several types of transfer function such as Hard-Limit, Linear, Log-Sigmoid and Tan-Sigmoid. Basically, The ANN respond to interested input patterns by choosing these weights and transfer function. As a result, the training process is used to select the weights to produce the required output with specified elements of inputs. Indeed, the nonlinear properties of the ANN cause by the nonlinear gain from their transfer function, the output of the active function presents the single neuron output[16], [17].

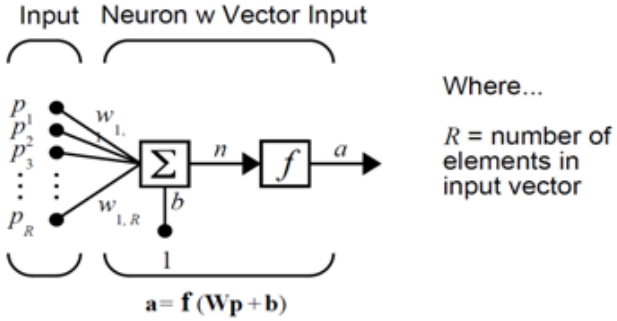

Fig. 1: A single neuron neural network with multiple inputs and one output(image adapted from [18]).

During the last three decades, many implementation strategies have been reported for implementation the individual neuron or/and the ANN[14], [17], [19], [20]. The literature studies show the general structure for a single neural implementation in hardware devices as shown in Fig. 2.

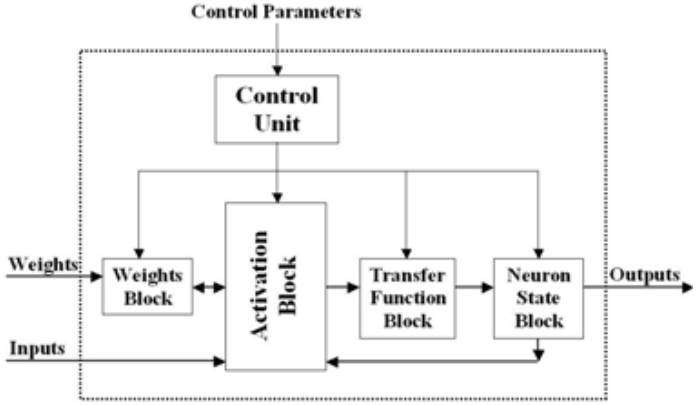

Fig. 2: Block diagram of single neuron hardware implementation (taken from [14]).

Practically, the weights are stored inside a ROM and the activation block computes the weighted and added of the inputs data. The transfer function is the important part within the ANN structure especially when this function is

nonlinear. The neuron state block provides the output data and necessary feedback data in some types of ANN configuration. All these blocks are controlled by the control unit which is managing the data stream from unit to unit, such as the weight number correspondence to the right input[14], [21], [22].

B. Tangent Hyperbolic Transfer Function Implementations

The key of the digital hardware implementation of ANNs is the approximation of a nonlinear transfer function; the most challenges on this implementation are the approximation accuracy, hardware size and the process time[23], [24].

The most common nonlinear transfer function is tangent hyperbolic (tanh) [23]which is used in the current design as defined in eqn. (1) and demonstrated in Fig. 3.

$$Tansh = \frac{e^{2x} - 1}{e^{2x} + 1} \qquad (1)$$

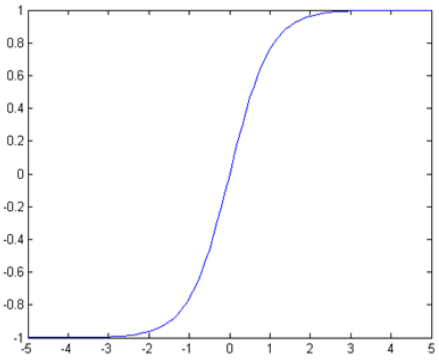

Fig. 3: Tangent Hyperbolic function used for ANN.

Many studies are reported on the optimizing the implementation of tangent hyperbolic function for ANNs uses [23–25]. However, all these approaches provided an approximation solution, the simplest approximations is called a linear approximation [26]which had been achieved using linear region with saturation levels as shown in Fig.4. Indeed, there are different types of linear approximation methods; however, all these approaches not accurate enough to give an effective approximation for the original tanh function[24], [26].

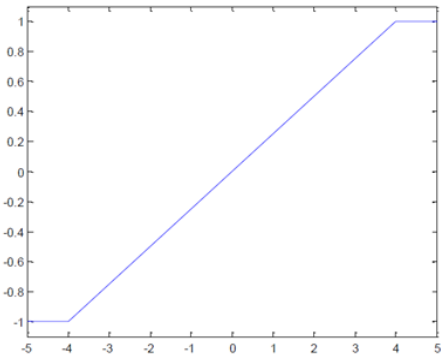

Fig. 4: Linear function with saturation for approximation Tangent Hyperbolic function.

On the other hand, nonlinear approximation methods have been presented for the same purposes, which are more precise than the linear methods. One of these approaches called The Elliott function used by[25] as seen in eqn.(2) and Fig.5. This function provides fast process time but it is not as the shape of the tanh function which is needed for the ANN implementation, in different words, it has high MSE.

$$y = \frac{1}{(1 + |x|)} \qquad (2)$$

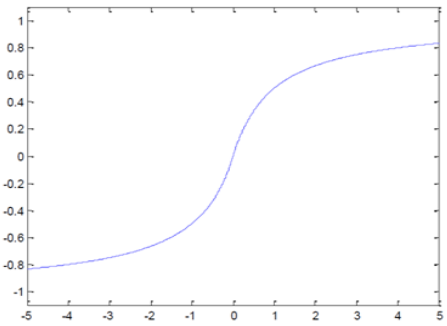

Fig. 5: Elliott function approximation

During the last three decades, different nonlinear approximations approaches is developed to improve the accuracy and hardware performance, some of them are accurate and the others not efficient enough in hardware term [14], [23]. As results, for the current study only two approaches are selected for evaluation and examination study to figure out the best performances for approximation and implementation the TDNN network.

The Piecewise linear (PWL) approximation [24], [26] and nonlinear approximation [23] have been selected for this task. Both strategies are implemented using FPGA board and then simulated and evaluated using the ModelSim SE 6.3a and Matlab 2011b. In addition, hardware performances for both approaches are examined in this section.

1) Piecewise Linear Approximation (PWL)

The most common linear approximation method for tangent hyperbolic function is called Piecewise linear (PWL) approximation which is most simple method compare with other. This algorithm use multiple condition regions of linear relationship between the input and the output [24], [26] as shown in Fig.6. The number of these linear segments and their location are selected carefully depend on some parameters such as MSE, processing time, and silicon area. In fact, the hardware size and process time for the transfer function are proportional with number of segments, while, the MSE is decreased once the number of theses segments is increased.

Eqn. (3) demonstrates the segments location and the number of linear equations which have been employed to achieve the digital hardware implementation with optimum performances.

$$Tanh\_App\_PWL = \begin{cases} a_1 \times x + b_1 & -0.75 \le x \le 0.75 \\ a_2 \times x + b_2 & 0.75 < x < 1.8 \\ a_3 \times x + b_3 & 1.8 \le x < 2.5 \\ a_2 \times x - b_2 & -0.75 > x > -1.8 \\ a_3 \times x - b_3 & -1.8 \ge x > -2.5 \\ sign(x) & otherwise. \end{cases} \quad (3)$$

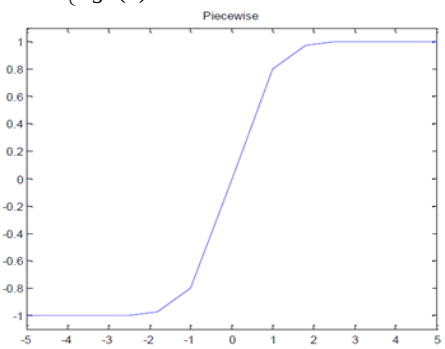

Fig. 6: Piecewise linear function

Fig.6 shows seven linear segments have been chosen to implement the concern active function for the target design in this research in order to find out the best and effective design.

Verilog HDL has been used to syntheses the FPGA board (XC3S1200E-5FG320C) for both approaches under Xilinx – ISE 9.2i platform. The input data length is 16-bit which include 1-bit for sign, 4-bit for integer number and 11-bit for the fraction, while the output data have 1-bit for sign and 15-bit for the fraction. As it seen, the input and output fraction lengths have been selected to reach the optimum performance. The ModelSim SE 6.3a has been used for the simulation the results for both approaches, also, the output data from the FPGA board have been captured and stored as Matlab data for further Matlab simulation and analysis.

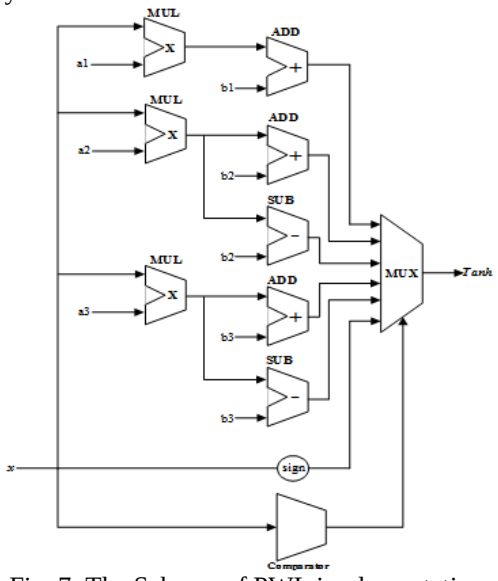

Fig. 7: The Scheme of PWL implementation

Fig.7 demonstrates the hardware implementation structure for PWL approach, meanwhile, Fig. 8 shows the ModelSim simulation results for the PWL output the. These data have been extracted from the FPGA board and plotted in Matlab as shown in Fig. 9. Typically, this algorithm needs several clock periods for the process time and more hardware area. Moreover, the MES error in this approach is quite high compare with original tanh function as shown in Fig. 9.

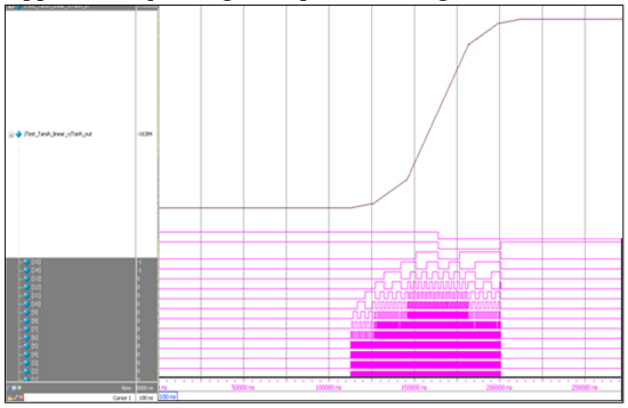

Fig. 8: The tanh approximation using PWL approach in ModelSim

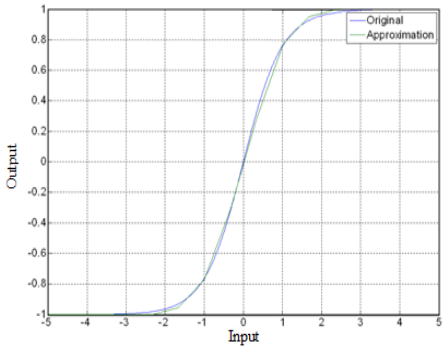

Fig. 9: The outputs of tanh and its PWL approximation in Matlab.

This system has been synthesized and placed and routed at 50 MHz system clock frequency. A summary of device utilization is shown in Table 1; it shows the total logic gates 1448 gates which are consumed for the active function implementation only. From the results, this design required high silicon area and power consumption from the FPGA recourses.

TABLE 1 DEVICE UTILIZATION SUMMARY FOR PWL METHOD

Device Utilization Summary			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of 4 input LUTs	134	17,344	1%
Logic Distribution			
Number of occupied Slices	95	8,672	1%
Number of Slices containing only related logic	95	95	100%
Number of Slices containing unrelated logic	0	95	0%
Total Number of 4 input LUTs	182	17,344	1%
Number used as logic	134		
Number used as a route-thru	48		
Number of bonded IOBs	34	250	13%
IOB Flip Flops	16		
Number of GCLKs	1	24	4%
Number of MULT18X18SIOs	3	28	10%
Total equivalent gate count for design	1,448		
Additional JTAG gate count for IOBs	1,632		

For the above reasons this study examined alternative algorithm procedure for target implementation using nonlinear approximation which has high performance in area consumption term and perform (see section (B)).

2) Nonlinear Approximation

Fig. 10 shows the nonlinear hardware structure which has been adopted for the final implementation for the tanh active function in the ANN system implementation. Due to the lower area consumption, MSE, and process time of this

approach compare with PWL method. The algorithms required just two function regions as shown in eqn.(4) [23], one of them is nonlinear and the second one is only sign of the input data, and that make this approach not just has a good approximation but also it is simple.

$$Tanh\_App\_Nonlinear = \begin{cases} x - 0.25 \times sign(x) \times x^2 & ,0 \le |x| \le 2; \\ sign(x) & ,otherwise. \end{cases} \quad (4)$$

The hardware implementation algorithm is presented in Fig. 9, which shows the main logic computation devices required to build this approach. This approach is only use two multipliers, one multiplexer, one subtractor, one shift register and one comparator. Indeed, there are many approaches have been used nonlinear approach with accurate approximation, however, in this study only one transfer function required and the hard ware size and power consumption is important because this system need to be implantable and acceptable size for the surgical procedure. Consequently, these limitations have been taken in account during the design and evaluation stages to figure out the best performance for concern application only.

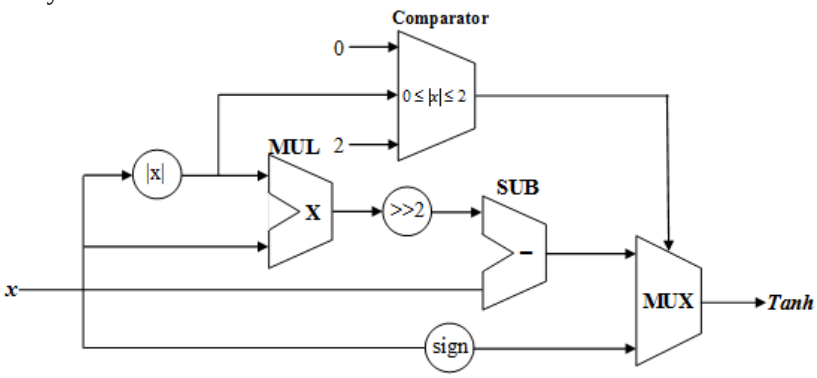

Fig. 10: The implementation scheme of nonlinear approximation method

Figs. 11 & 12 present the simulation results with ModelSim and Matlab respectively. Specifically, Fig.11 illustrates the simulation result in ModelSim, while Fig.12 demonstrates the comparison between the original tanh function with nonlinear approximation output which have been captured from the FPGA board.

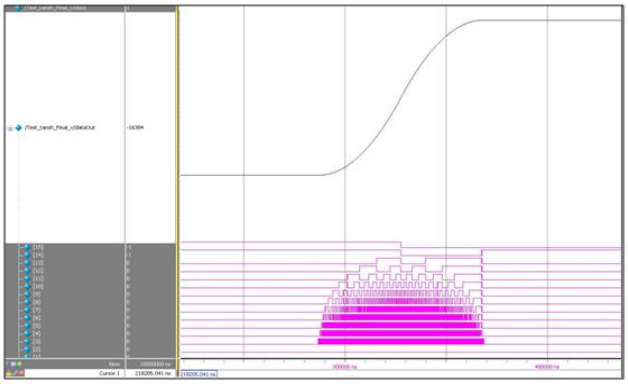

Fig. 11: The tanh approximation using nonlinear approach in ModelSim

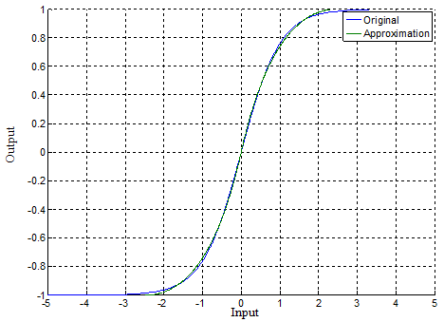

Fig. 12: The outputs of tanh and its nonlinear approximation in Matlab.

This system implementation has been configured and placed and routed at 50 MHz built-in clock frequency on the FPGA board. The summary of the final synthesis report is shown in Table 2; it shows the total logic gates required to implement the design is only 642 gates for the transfer function. In fact, this approach consumed only %45 from silicon area which has been used for the PWL implementation. This results show significant saving has been achieved on both silicon area and power consumption.

TABLE 2 DEVICE UTILIZATION SUMMARY FOR NONLINEAR METHOD

Device Utilization Summary				
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of 4 input LUTs	65	17,344	1%	
Logic Distribution				
Number of occupied Slices	37	8,672	1%	
Number of Slices containing only related logic	37	37	100%	
Number of Slices containing unrelated logic	0	37	0%	
Total Number of 4 input LUTs	67	17,344	1%	
Number used as logic	65			
Number used as a route-thru	2			
Number of bonded IOBs	32	250	12%	
Number of MULT18X18SIOs	1	28	3%	
Total equivalent gate count for design	642			
Additional JTAG gate count for IOBs	1,536			

The synthesis configuration report for the PWL and nonlinear implementation methods for the hyperbolic tangent sigmoid function are shown in Table 3 and Table 4 respectively. Meanwhile, Table 5 shows the time proceed, MSE and the total memory usage for both strategies.

TABLE 3 SYNTHESIS REPORT FOR PWL METHOD

```
Device utilization summary:
---------------------------

Selected Device : 3s1200efg320-5

 Number of Slices:                       98  out of   8672     1%
 Number of 4 input LUTs:                182  out of  17344     1%
 Number of IOs:                          34
 Number of bonded IOBs:                  34  out of    250    13%
    IOB Flip Flops:                      16
 Number of MULT18X18SIOs:                 3  out of     28    10%
 Number of GCLKs:                         1  out of     24     4%
```

TABLE 4 SYNTHESIS REPORT FOR NONLINEAR METHOD

```
Device utilization summary:
---------------------------

Selected Device : 3s1200efg320-5

 Number of Slices:                       37  out of   8672     0%
 Number of 4 input LUTs:                 67  out of  17344     0%
 Number of IOs:                          32
 Number of bonded IOBs:                  32  out of    250    12%
 Number of MULT18X18SIOs:                 1  out of     28     3%
```

TABLE 5 THE IMPLEMENTATION CHARACTERISTICS SUMMARY FOR BOTH TRANSFER FUNCTIONS APPROXIMATION.

Method	Mean Squared Error(MSE)	Total memory usage (Kb)	Time Consumption (sec.)
Piecewise linear (PWL)	0.355	159224	5.86
Nonlinear	0.0309	157176	4.53

The results show that the nonlinear approach offers MSE less ten times against the PWL method approximation, also, it consumed less silicon area and process time. Consequently, the nonlinear approximation has been selected for the final design implementation of the TDNN network.

C. Implementation the TDNN system for VSR

The adopted structure of TDNN for VSR is shown in Fig.13. The two main challenges that require to be tackled in hardware implementation are the system configuration structure and process time.

In general, the TDNN structure for interested system consist of two layers which are presented in multilayer networks type, each layer contains from individual neuron, the inputs data (nine tripolar channels) in the first layer delayed with

tap delay line (TDL) and multiplied with the weights and then added together with bias, the adder output is fed to the transfer function, the output of the active function presents the first layer output and connected with the second layer. The second layer is only having single weight and single bias with linear transfer function.

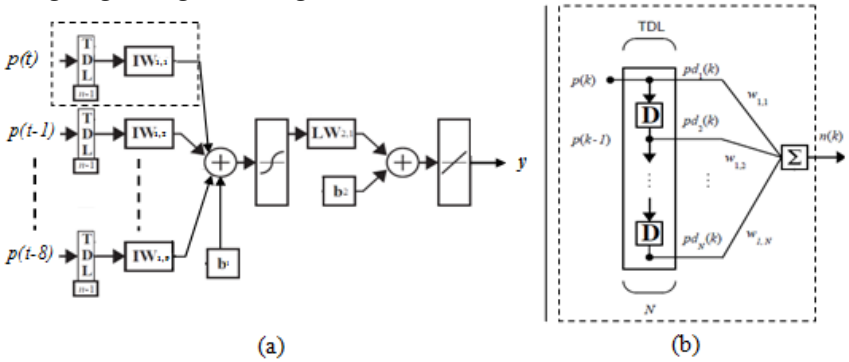

Fig. 13: Where; (a) The block diagram of TDNN for VSR and (b) The Time delay line with weights in Fig. 13(a) (Image adapted from [18]).

There are several key points of arithmetic process that need to be concerned during the ANNs implementation process on FPGAs, such as data representation, inner arithmetic computation, implementation of transfer functions, process time and consumption area [14].

The most effective aspect is the data representation within the digital environment process because it can have an important effect on FPGA area consumption. The digital data can be formatted as fixed-point or floating-point as numbers. Indeed, the Floating-point format provides a wider range of numbers than fixed-point with the same length of bits. Nevertheless, fixed-point format is more efficient and better performance than the Floating-point format for ANN implementation on FPGA [14], [19], also it needs less silicon area compare with the another format [14], [22].

As a result, this study adopted the fixed-point number representation for the case study to avoid the limitations of using floating-point format with the FPGA.

Practically, the required inner multipliers dominate on the computation within the TDNN implementation because it used high number of weights. The parallel implemented provides an option for implementation process, however it can increase the cost of the system due to the large amounts of silicon area required on FPGA. In addition, parallel implementation needs all the input data to be stored before any arithmetic process, which mean further cost in consumption area term. As a result, the sequential implementation is adopted here to minimize the number of the internal multipliers required because it process sequentially by multiply-accumulate unit. Consequently, it presents a good saving in the area consumption [13], [19], [21], [27], [28].Practically, weights array are stored within a ROM in the internal or external memory of the FPGA. The index control is used to provide the right index for both delayed inputs and weights.

The inputs amplitude (nine tripolar channels) are remapped between -1and 1 at the input stage and then formatted as fixed-point with one bit signed, one bit integer and 14 bits fractions as an optimum format.

The hardware implementation of full system shown in Fig. 14, which is consist of two layers, the first layer split into three units. First unit is the delay line tap with weights which is implemented using register and ROM, basically, the ROM unit contains 40 weights for each channel and the register contain the value of tap delay line (TDL) which is contain 39 unit delays as shown in Fig.13, both of them are control using counter for index selection. The second unit is multiply accumulator unit to process the computation sequentially. The First and the second units are responsible for receiving the inputs data, delay inputs and multiplying them with correspondence weights correctly. The last unit is the bias additive and the tangent hyperbolic function which is implemented as nonlinear approximation in previous section. The main reason for splitting the nonlinear transfer function from the other units is to gain saving in the silicone area on FPGA. This sequentially procedure of neuron computes it only used one activation function per layer instead of using one per neuron or channel.

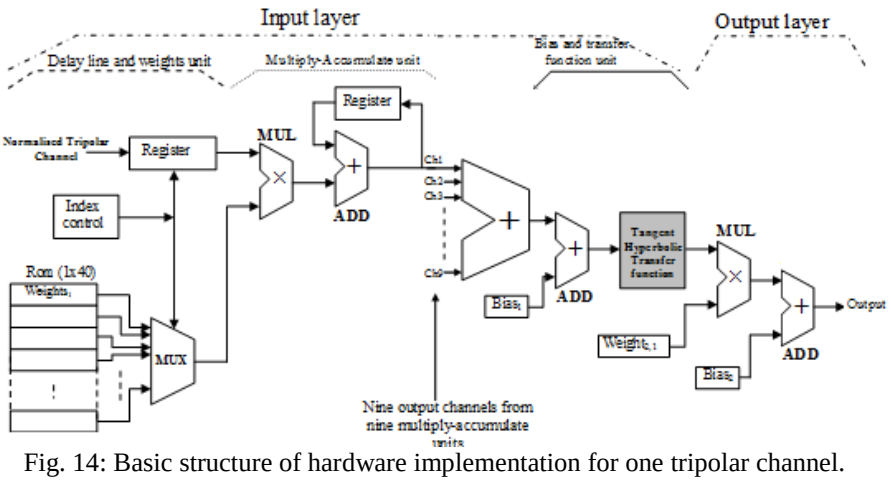

Fig. 14: Basic structure of hardware implementation for one tripolar channel.

The output layer is basically a linear layer contains from only one weight and one bias, so it is not include any important computation functional unit.

The implementation of TDNN for VSR is a multilayer perceptron with one hidden and one output layers structure shown in Fig. 14, the inputs data for this network are nine tripolar channels which are received form DEMUX and converter unit in fig. 1 with sample frequency 32.5k Sa/s. Each segment is 400 samples in length 12ms. Because the system functions in real-time, the hardware implemented to meet of this timing restriction. In order to evaluate the implementation features, the whole system include the data acquisition unit, DEMUX and converter unit have been synthesized, and placed and routed.

The implementation object FPGA for TDNN is Xilinx Spartan 3E-1200(XC3S1200E-5FG320C). The input length used for arithmetic precisions for synthesis is 16 bits for each channels divided in one bit for sign and one bit integer and 14 bits for fraction, while, the weights word length have been chosen as 16 bits which includes one bit for sign and 15 bits for fraction, The output word length is sited to have 16 bits which consists of one bit for sign and 4-bit for integer and 11 bit for fraction.

The full hardware system after implementation and verification is shown in fig.15 which includes electrode simulation unit, amplifiers and ADC unit and the DSP unit. The electrode simulation unit is used to generate eleven unipolar channels [1-4] which are connected by BNC connecter cable with amplifiers &ADC unit. This unit is connected with the DSP unit by four channels, three of them are provided by DSP unit which are sampling frequency, sync input, and commend controller, and the last one is used to supply tripolar channels to the DSP unit after demultiplexing and converting them from monopolar to tripolar [1-4]. The DSP unit is used for processing the data in both conventional and TDNN approach which have been synthesis and implemented on the same FPGA board. Indeed, the flexibility of the FPGA allowed reconfiguring and generating multi- prototype implementations designs using only one FPGA board. This advantage of choosing FPGA for implementation target is saving the efforts, time and cost during prototype design. Also, generation multi- prototypes for both approaches provide a wide selection options to have fair comparison among these implementation methods without needing to redesign the system each time because it is easy to upload them and running at any time. In fact, this FPGA future can not be easy offer with alternative technologies. Moreover, the cost it self is very reasonable and efficient for producing a prototype task.

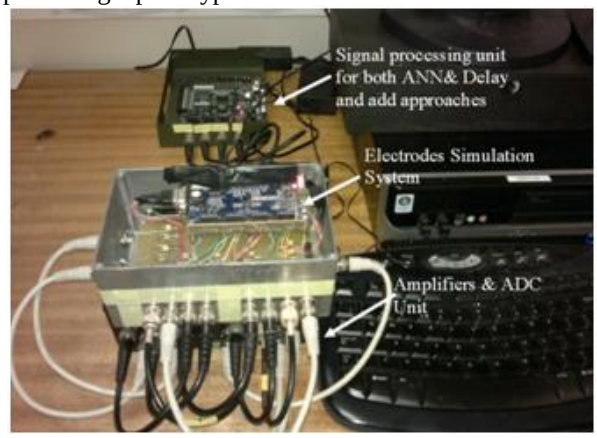

Fig. 15: The hardware system constructions for the complete design

III. RESULTS AND DISCUSSION

The system implementation has been correctly synthesized and placed and routed at 50 MHz system clock frequency. A summary of device utilization for the delay and add, TDNN parallel and sequential implementation are shown in Table 6, 7 and 8 respectively.

TABLE 6 THE DEVICE UTILIZATION SUMMARY FOR THE DELAY AND ADD IMPLEMENTATION.

Device Utilization Summary				
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	285	17,344	1%	
Number of 4 input LUTs	1,426	17,344	8%	
Logic Distribution				
Number of occupied Slices	952	8,672	10%	
Number of Slices containing only related logic	952	952	100%	
Number of Slices containing unrelated logic	0	952	0%	
Total Number of 4 input LUTs	1,628	17,344	9%	
Number used as logic	1,426			
Number used as a route-thru	202			
Number of bonded IOBs	38	250	15%	
IOB Flip Flops	12			
Number of Block RAMs	13	28	46%	
Number of GCLKs	3	24	12%	
Total equivalent gate count for design	867,187			
Additional JTAG gate count for IOBs	1,824			

TABLE 7 THE DEVICE UTILIZATION SUMMARY FOR THE TDNN SEQUENTIAL COMPUTATION IMPLEMENTATION.

Device Utilization Summary				
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	6,856	17,344	39%	
Number of 4 input LUTs	6,841	17,344	39%	
Logic Distribution				
Number of occupied Slices	7,063	8,672	81%	
Number of Slices containing only related logic	7,063	7,063	100%	
Number of Slices containing unrelated logic	0	7,063	0%	
Total Number of 4 input LUTs	7,324	17,344	42%	
Number used as logic	6,841			
Number used as a route-thru	483			
Number of bonded IOBs	38	250	15%	
IOB Flip Flops	12			
Number of Block RAMs	13	28	46%	
Number of GCLKs	3	24	12%	
Number of MULT18X18SIOs	28	28	100%	
Total equivalent gate count for design	969,519			
Additional JTAG gate count for IOBs	1,824			

TABLE 8 THE DEVICE UTILIZATION SUMMARY FOR THE TDNN PARALLEL COMPUTATION IMPLEMENTATION.

Device Utilization Summary				
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	6,964	17,344	40%	
Number of 4 input LUTs	7,256	17,344	41%	
Logic Distribution				
Number of occupied Slices	7,262	8,672	83%	
Number of Slices containing only related logic	7,262	7,262	100%	
Number of Slices containing unrelated logic	0	7,262	0%	
Total Number of 4 input LUTs	7,611	17,344	43%	
Number used as logic	7,256			
Number used as a route-thru	355			
Number of bonded IOBs	38	250	15%	
IOB Flip Flops	12			
Number of Block RAMs	22	28	78%	
Number of GCLKs	3	24	12%	
Number of MULT18X18SIOs	28	28	100%	
Total equivalent gate count for design	1,563,624			
Additional JTAG gate count for IOBs	1,824			

The synthesis reports for them show that the delay and add method has the lowest consumption silicon area which only uses 285 flip-flops and 1426 LUTs, while TDNN parallel implementation requires 6964 flip-flops and 7256 LUTs. On the other hand, the sequential computation implementation uses 6856 flip-flops and 6841 LUTs, which is less than parallel implementation but it is slightly more than delay and add method. Moreover, the power consumption has been investigated and measured experimentally to evaluate the circuit performance. The results of sequential computation implementation of the TDNN system have been compared with parallel computation implementation of the TDNN from one hand, and with the delay and add approach implementation on the other hand .The same FPGA board and input and output word length have been used for this comparison. Table 9 shows the device utilization summary and estimated power consumption for TDNN implementation options and delay and add approaches. The experimental results show that the total power consumption for the delay and add, TDNN parallel and sequential computation implementation are 22.8 , 45.6 and 25.56 mWatt respectively. Compared with the traditional delay and add technique, the sequential TDNN requires about 11% more than the traditional delay and add method in term number of gates and the power consumption. Meanwhile, the parallel computation implementation of TDNN requires approximately double the number of gates and double the power consumption in the delay and add method. However this increase should be considered in the context that this design is implemented using FPGA, which is well known to have significantly higher power consumption than a comparable custom device of the type that would be required in an implantable system. For example using 0.18µm CMOS for implementation process would be enough to counteract the impact of the additional processing in the ANN-based system.

TABLE 9 DEVICE UTILIZATION SUMMARY AND POWER CONSUMPTION FOR BOTH APPROACHES

Method	Device utilization summary and power consumption
Delay-and-add	Total equivalent gate count for design: 867,187 Current consumption:19 mA

	Total Power consumption(measured) : 22.8mWatt Total power consumption (estimated) =22.23mWatt
TDNN (Parallel computation implementation)	Total equivalent gate count for design: 1,563,624 Current consumption: 38 mA Total Power consumption (measured) : 45.6mWatt Total power consumption (estimated) =48.94mWatt
TDNN (sequential computation implementation)	Total equivalent gate count for design: 969,519 Current consumption: 21.3 mA Total Power consumption (measured) : 25.56 mWatt Total power consumption (estimated) =26.6 mWatt

In order to test and verify the FPGA-Based on TDNN, The hardware stricture is dealt by the generator written in Verilog HDL, and then the output file is generated automatically into MATLAB using Microsoft Visual Studio 2010 for the simulation processes.

The comparison between the hardware TDNN outputs against simulation TDNN output and training target have been examined. The results show that the MSE between the target and the hardware TDNN output is 0.0034, while the MSE for the hardware TDNN output compare with simulated output is about 0.0038 as shown in fig 16 and 17 respectively. Also, the velocity selectivity is decrease by 2% against simulation results along the velocity domain. Indeed, this impact comes as a result from the MSE increment, which occurs due to the fixed point limitation and nonlinear transfer faction approximation. On the other hand, the conventional method based on delay and add also suffer from the same issue. As a results, the TDNN stills more efficient and accurate than the linear approach even with hardware implimentation.

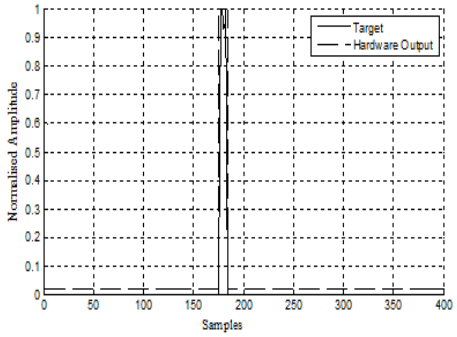

Fig. 16: Comparing the output of FPGA-based on TDNN with Matlab Target

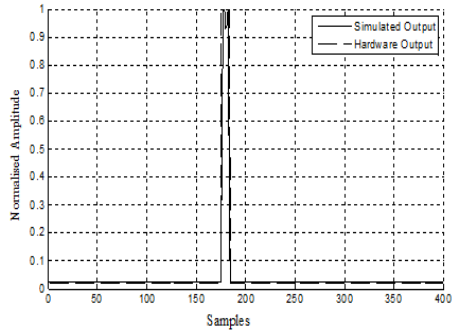

Fig. 17: Comparing the output of FPGA-based on TDNN with Matlab simulation output

IV. CONCLUSIONS

In this study use of TDNN has been examined on configured for VSR improvement for ENG neural activity classification system. The implementation has been synthesis on FPGA board and it is optimized the process computation using sequential type of TDNN algorithm to reduce the area and power consumption on the FPGA. The system successfully functions in real time recording application without losing the velocity selective resolution which is developed for it.

In this paper a structure for implementing TDNN on FPGAs has been presented. The main challenges in implementation of TDNN on FPGAs have been discussed and the layer and neuron configurations for TDNN implementation have been described. A full system for VSR is implemented with only two layers, one for input and one for output. A multiply-accumulate unit, weight ROM, single nonlinear transfer function is configured using nonlinear approximation method, and it is used by all channels in each layer.

The results show the nonlinear transfer function approximation could minimize hardware size by 55% compare with linear approach. Also, the mean squared error (MSE) in nonlinear approach is reduced by ten times compare with PWL

method. As a result, the nonlinear implementation method has been used in TDNN implementation. In addition, the system is implemented using two different methods, a sequential and parallel implementation approach. The results show that the sequential implementation approach requires a lesser amount of silicon area compare with parallel method by 38% and the power consumption is only 25.56 mWatt. This is a important improvement have been achieved to reduce the consumed power and hardware size which are very significant factors for implanted integrated circuits applications.

REFERENCES

[1] A.I. Al-Shueli, "Validation of Artificial Neural Network Method for Action Potential Detection and Classification Based on Velocity Selective Recording" International Journal of Computer Science and Mobile Computing(IJCSMC), vol.3, no. 10, pp. 10-19, 2014.

[2] A.I. Al-Shueli, C.T. Clarke, and J.T. Taylor, "Simulated Nerve Signal Generation for Multi-electrode Cuff System Testing," Biomedical Engineering and Biotechnology (iCBEB), 2012 International Conference on , vol., no., pp.892,896, 28-30 May 2012.

[3] A.I. Al-Shueli, C.T. Clarke, N. Donaldson, and J.T. Taylor, "Improved Signal Processing Methods for Velocity Selective Neural Recording Using Multi-Electrode Cuffs," Biomedical Circuits and Systems, IEEE Transactions on , vol.8, no.3, pp.401,410, June 2014.

[4] A.I. Al-Shueli, "Signal Processing for Advanced Neural Recording Systems," PhD Thesis, University of Bath, Bath, UK, 2013.

[5] C. T. Leondes, "Algorithms and Architectures Neural Network Systems Techniques and Applications", USA: Academic Press, 1998.

[6] M. A. El-Sharkawi, "Neural network application to high performance electric drives systems," in Proceedings of IECON '95 - 21st Annual Conference on IEEE Industrial Electronics, vol. 1, pp. 44–49, 1995.

[7] K. H. Kim and S. J. Kim, "Neural spike sorting under nearly 0-dB signal-to-noise ratio using nonlinear energy operator and artificial neural-network classifier.," IEEE transactions on bio-medical engineering, vol. 47, no. 10, pp. 1406–11, Oct. 2000.

[8] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and Lang K. J., "Phoneme recognition using time-delay neural networks," IEEE Trans. Acoustics, Speech, Signal Processing, vol. 37, pp. 328–339, 1989.

[9] R. Chandra and L. M. Optican, "Detection, classification, and superposition resolution of action potentials in multiunit single-channel recordings by an on-line real-time neural network.," IEEE transactions on bio-medical engineering, vol. 44, no. 5, pp. 403–12, May 1997.

[10] K. Imfeld, S. Neukom, A. Maccione, Y. Bornat, S. Martinoia, P.-A. Farine, M. Koudelka-Hep, and L. Berdondini, "Large-scale, high-resolution data acquisition system for extracellular recording of electrophysiological activity.," IEEE transactions on bio-medical engineering, vol. 55, no. 8, pp. 2064–73, Aug. 2008.

[11] U. Frey, U. Egert, F. Heer, S. Hafizovic, and A. Hierlemann, "Microelectronic System for High-Resolution Mapping of Extracellular Electric Fields Applied to Brain Slices," Biosensors & bioelectronics, vol. 24, no. 7, pp. 2191–8, Mar. 2009.

[12] A. R. Omondi, "FPGA Implementations of Neural Networks", Springer US, 2006.

[13] I. Kuon, R. Tessier, and J. Rose, "FPGA Architecture: Survey and Challenges," Foundations and Trends in Electronic Design Automation, vol. 2, no. 2, pp. 135–253, 2007.

[14] C. S. Rai and A. P. Singh, "A Review of Implementation Techniques For Artificial Neural Networks," University School of Information Technology,GGS Indraprastha

[15] C. T. Clarke, X. Xu, R. Rieger, J. Taylor, and N. Donaldson, "An implanted system for multi-site nerve cuff-based ENG recording using velocity selectivity," Analog Integrated Circuits and Signal Processing, vol. 58, no. 2, pp. 91–104, Nov. 2008.

[16] W. Matlab, "Artificial Neural Networks The Tutorial."

[17] N. Izeboudjen, a. Bouridane, a. Farah, and H. Bessalah, "Application of design reuse to artificial neural networks: case study of the back propagation algorithm," Neural Computing and Applications, vol. 21, no. 7, pp. 1531–1544, Dec. 2011.

[18] H. Demuth, Neural Network Toolbox User's Guide. The MathWork,Inc, 2000.

[19] P. Skoda, T. Lipic, A. Srp, B. M. Rogina, K. Skala, and F. Vajda, "Implementation Framework for Artificial Neural Networks on FPGA," in MIPRO, 2011 Proceedings of the 34th International Convention, 2011, pp. 274–278.

[20] M. P. Rothney, M. Neumann, A. Beziat, and K. Y. Chen, "An artificial neural network model of energy expenditure using nonintegrated acceleration signals," J Appl Physiol, vol. 20892, pp. 1419–1427, 2007.

[21] B. Van Liempd, D. Herrera, and M. Figueroa, "An FPGA-Based Accelerator for Analog VLSI Artificial Neural Network Emulation," 2010 13th Euromicro Conference on Digital System Design: Architectures, Methods and Tools, pp. 771–778, Sep. 2010.

[22] R. Raeisi and A. Kabir, "Implementation of Artificial Neural Network on FPGA," 2006.

[23] C. Lin and J. Wang, "A digital circuit design of hyperbolic tangent sigmoid function for neural networks," in 2008 IEEE International Symposium on Circuits and Systems, 2008, pp. 856–859.

[24] K. Basterretxea, J. M. Tarela, and I. Campo, "Approximation of sigmoid function and the derivative for hardware implementation of artificial neurons," in Circuits, Devices and Systems, IEEE Proceedings, 2004, vol. 151, no. 1, pp. 18– 24.

[25] D. L. Elliott, "A Better Activation Function for Artificial Neural Networks," 1993.

[26] D. Myers and R. Hutchinson, "Efficient implementation of piecewise linear activation function for digital VLSI neural networks," Electron. Lett., vol. 25, no. 24, pp. 1662–1663, 1989.

[27] M. Krips, T. Lammert, and A. Kummert, "FPGA implementation of a neural network for a real-time hand tracking system," in Electronic Design, Test and Applications, 2002. Proceedings. The First IEEE International Workshop on, 2002, pp. 313–317.

[28] S. R. D. Naoussi, N. K. Nguyen, H. Berviller, C. H. Kom, J. P. Blondé, M. Kom, F. Braun, M. Kom, and F. Braun, "FPGA resources reduction with multiplexing technique for implementation of ANN-based harmonics extraction by mp-q method," in IECON 2010 - 36th Annual Conference on IEEE Industrial Electronics Society, 2010, pp. 2043–2048.

