FPGA Implementation of a Probabilistic Neural Network for a Bioelectric Human Interface

Nan Bu, Taiji Hamamoto and Toshio Tsuji
Dep. of Artificial Complex Systems Engineering
Hiroshima Univ.
1-4-1, Kagamiyama
Higashi-Hiroshimashi, 739-8527 JAPAN

Osamu Fukuda
National Institute of Advanced
Industrial Science and Technology
1-2-1, Namiki
Tsukubashi, 305-8564 JAPAN

***Abstract*—Since a probabilistic neural network (PNN) provides a stochastic perspective of pattern discrimination, it has been proven to be efficient for complicated data such as bioelectric signals. As for practical implementation, however, a general-purpose computer is usually necessary, so that a compact design of an application system is difficult to be realized. This paper describes a field programmable gate array (FPGA) implementation of a PNN, with which system on chip (SoC) design of a bioelectric human interface device becomes possible. Its effectiveness is then verified with a practical application, and it is shown that the hardware implementation provides comparable performance with the software solution on a general-purpose computer.**

I. INTRODUCTION

In the context of pattern discrimination, probabilistic neural networks (PNNs) have been widely studied [1]-[3]. Due to the prominent nonlinear approximation capability of PNNs, it is expected that, by training the network architecture and the weights appropriately, PNNs can estimate the real posterior probability distribution of input patterns with arbitrary accuracy. Recently, PNNs have been used as an important tool for pattern discrimination, and have been proven to be efficient especially for complicated problems such as classification of bioelectric signals.

In [2], Tsuji et al. proposed a PNN, called a log-linearized Gaussian mixture network (LLGMN), which estimates the posterior probability based on a Gaussian mixture model (GMM) and the log-linear model. Although weights of the LLGMN correspond to a nonlinear combination of the GMM parameters, such as the mixture coefficients, mean vectors, and covariance matrices, constraints on the parameters in the statistical model are relieved in the LLGMN. Therefore, a simple learning algorithm can be derived, and the LLGMN is expected to have high performance in the case of statistical pattern discrimination. The LLGMN has been successfully applied to pattern discrimination of bioelectric signals, e.g., electromyogram (EMG) and electroencephalogram (EEG), and has been further used to develop various human interface applications like prosthetic device control, an EMG-based pointing device, and so on [4]-[6].

So far, software implementation of the LLGMN on a general-purpose computer is usually adopted. However, development of practical human interface applications is quite difficult, since this kind of applications requires the interface devices to be compact and portable, especially in the design of wearable and handheld devices like the EMG-controlled prosthetic limbs. Also, in cases where real-time processing is necessary (e.g., the EMG-based pointing device), implementation of the LLGMN in software would increase the CPU load dramatically. To deal with these problems, a dedicated digital hardware realization of LLGMN is required.

In recent years, continuous performance increase has been conducted in the field of digital hardware. Also, due to the reliability, programmability and availability of numerous design tools, various neural networks have been implemented on digital hardware [7]-[9]. In this paper, we describe the digital realization of LLGMN on a field programmable gate array (FPGA) chip, and verify it on the EMG-based pointing device system [5]. The FPGA is chosen in contrast to other digital platform (e.g., application specific integrated circuit (ASIC)), since the FPGA is flexible in practical application under consideration, i.e., it can be reconfigured at will. Furthermore, there are many commercial off the shelf chips available, which make the development cost rather small.

In the following section, a brief review of the LLGMN is described. Then, section III introduces some issues of the FPGA implementation of LLGMN and its performance. Experimental results of a bioelectric human interface system, that is the EMG-based pointing device integrated on one FPGA chip, are presented in Section IV. Finally, Section V gives some discussions and a summary of the paper.

II. LLGMN

The LLGMN [2] is based on the GMM and the log-linear model of probability distribution function (pdf). By applying the log-linear model to a product of the mixture coefficient and the mixture component of GMM, the semiparametric model of pdf is incorporated into a three-layer feedforward NN (Fig. 1).

First, in the pre-process, the input vector $\mathbf{x} \in \Re^d$ is converted into the modified vector $\mathbf{X} \in \Re^H$ as follows:

$$\begin{aligned}\mathbf{X} &= (1, \mathbf{x}^{\mathrm{T}}, {x_1}^2, x_1x_2, \cdots, x_1x_d, {x_2}^2, x_2x_3, \cdots, \\ &\quad x_2x_d, \cdots, {x_d}^2)^{\mathrm{T}} \end{aligned} \tag{1}$$

where $x_i, i = 1, 2, \cdots, d$, are the elements of $\mathbf{x}$ and $H = 1 + d(d+3)/2$. The first layer consists of H units corresponding to the dimension of $\mathbf{X}$ and the identity function is used for activation of each unit. ${}^{(1)}O_h$ $(h = 1, \cdots, H)$ in Fig. 1 denotes the output of the hth unit in the first layer.

In the second layer, each unit receives the output of the first layer weighted by the weight $w_h^{(c,m)}$ $(c = 1, \cdots, C;\ m = 1, \cdots, M_c)$ and outputs the posterior probability of each Gaussian component. Here, C denotes the number of classes, and M_c is the number of Gaussian components in class c. The relationships between the input of unit $\{c, m\}$ in the second layer (${}^{(2)}I_{c,m}$) and the output (${}^{(2)}O_{c,m}$) are defined as

$${}^{(2)}I_{c,m} = \sum_{h=1}^{H} {}^{(1)}O_h w_h^{(c,m)} \tag{2}$$

$${}^{(2)}O_{c,m} = \frac{\exp[{}^{(2)}I_{c,m}]}{\sum_{c'=1}^{C}\sum_{m'=1}^{M_{c'}} \exp[{}^{(2)}I_{c',m'}]} \tag{3}$$

0-7803-8346-X/04/$20.00

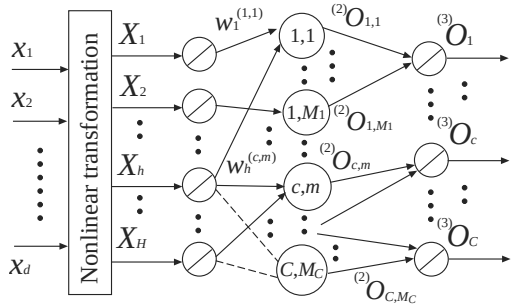

Fig. 1. Structure of LLGMN [2].

where $w_h^{(C,M_C)} = 0$ $(h = 1, \cdots, H)$.

The third layer consists of C units corresponding to the number of classes. The unit c sums up the outputs of M_c components $\{c, m\}$ in the second layer. The function between the input and the output is described as

$$^{(3)}O_c = {}^{(3)}I_c = \sum_{m=1}^{M_c} {}^{(2)}O_{c,m} \tag{4}$$

where the output $^{(3)}O_c$ corresponds to the posterior probability of class c.

A simple back-propagation training algorithm is feasible [2]. Consider a training set $\{\mathbf{x}^{(n)}, \mathbf{T}^{(n)}\}$ $(n = 1, \cdots, N)$, where $\mathbf{T}^{(n)} = \{T_1^{(n)}, \cdots, T_C^{(n)}\}$. If the input vector $\mathbf{x}^{(n)}$ belongs to class c, $T_c^{(n)} = 1$, and $T_{c'}^{(n)} = 0$ for all of the other class c'. An energy function, which is the negative log-likelihood, can be defined as:

$$J = -\sum_{n=1}^{N}\sum_{c=1}^{C} T_c^{(n)} \log {}^{(3)}O_c^{(n)}. \tag{5}$$

When probabilistic teacher signals that take continuous values of $\{0, 1\}$ are used, instead of (5), an energy function based on the Kullback information criterion is introduced

$$J' = -\sum_{n=1}^{N}\sum_{c=1}^{C} T_c^{(n)} (\log {}^{(3)}O_c^{(n)} - \log T_c^{(n)}). \tag{6}$$

In the training process, weight $w_h^{(c,m)}$ is modified to minimize the energy function ((5) or (6)). In what follows, the LLGMN is expected to approximate the posterior probability $P(c|\mathbf{x})$, when an input vector $\mathbf{x}$ is presented.

III. FPGA IMPLEMENTATION

A. Basic Design

Before proceeding hardware implementation of the LLGMN, one must consider some important issues, such as the length of data representation, realization of the non-linear operations involved in the LLGMN's algorithm, the processing speed, the FPGA chip size, etc. In this paper, a fixed-point format of data representation is used to speed the processing up. Numbers of bits for the integer and the fractional parts should be determined with respect to the evaluation criteria of the given application.

In the LLGMN's algorithm, exponential and logarithmic functions are involved. One straightforward and efficient implementation of these non-linear functions is using a look-up table (LUT). This method usually requires a large number of registers, so that more chip area is needed for LUT. To overcome this problem, the spacing of two adjacent breakpoints is modulated according to the characteristics of corresponding functions. For the LUT of the exponential function, for example (see Fig. 2), the spacing is wider on the side (a), where the output value Y varies much less than that of the opposite side (c). The spacing of the breakpoints is set as 2^l $(l \in \mathbf{Z})$ in order to speed the computation up using the table. In this paper, the input domain of the exponential function is simply divided into three ranges: $[-10, -5), [-5, 5]$ and $(5, 10]$, and the corresponding spacing as: 0.5, 0.25 and 0.125.

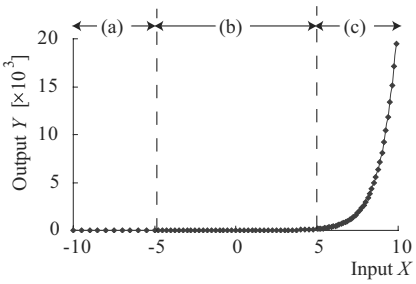

Fig. 2. The LUT-based exponential function.

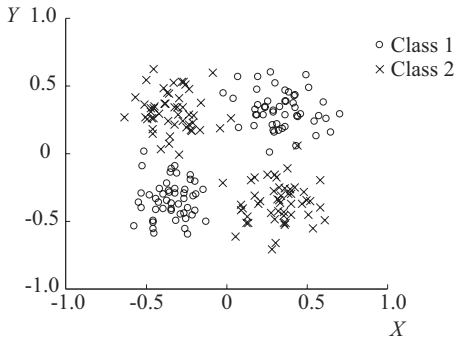

Fig. 3. Scatter graph of discrimination data (100 data/class).

Also, pipeline processing is utilized in the proposed design to gain high throughput, and to ease the bottleneck of data exchange between FPGA chip and external memories. Furthermore, in order to improve the processing speed, some multiplications by given constants are substituted with bit shifting operations.

B. Implementation and Performance Evaluation

We implemented the LLGMN on a development board (RC1000, Celoxica) hosting a Xilinx Virtex family BG560 FPGA chip with up to two million system gates (XCV2000E-6BG560). On RC1000, there is 8Mb of SRAM directly connected to the FPGA in four 32-bit wide memory banks. The LLGMN hardware is described in Handel-C language and designed with the Celoxica DK2 design suite.

Pattern discrimination experiments on artificial data were conducted to compare the LLGMN based on hardware with the one based on software (C language). There are two classes $(C = 2)$ on a two-dimensional input space; each consists of two Gaussian sources $(M_c = 2)$. The centers of Gaussian distributions in class one are $(0.35, 0.35)$ and $(-0.35, -0.35)$, in class two $(0.35, -0.35)$ and $(-0.35, 0.35)$, and all have a variance of 0.0225. Examples of the data are shown in Fig. 3,

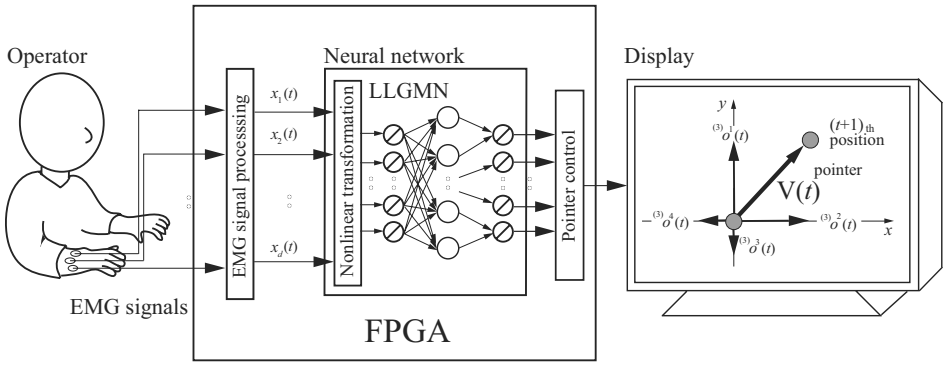

Fig. 4. Structure of the EMG-based pointing device system.

TABLE I
EVALUATION RESULTS ON THE ERROR.

Mean	S. D.	Maximum	Minimum
0.831×10^{-2}	0.0118	0.0826	0.70×10^{-5}

where the samples from the two classes are represented by $\circ$ and $\times$. For each class, we generated 100 samples for on-chip learning, and then validate the trained LLGMN chip with test data (1000 samples per class). The theoretical limit of the error probability for the test dataset is 2.00%.

The classification rates of the implementations on software and hardware are 97.8%, and 97.9%, respectively. The computational accuracy of hardware implementation is evaluated with a mean squared error as

$$err(t) = \frac{1}{C}\sum_{c=1}^{C}\sqrt{(P_{\mathrm{S}}(c|\mathbf{x}^{(t)}) - P_{\mathrm{H}}(c|\mathbf{x}^{(t)}))^2} \qquad (7)$$

where $P_{\mathrm{S}}(c|\mathbf{x}^{(t)})$ and $P_{\mathrm{H}}(c|\mathbf{x}^{(t)})$ indicate the posterior probability of the input pattern $\mathbf{x}^{(t)}$ computed in software and hardware, respectively. Some statistics on the error are shown in Table I. It is believed that this error can be reduced by further improving the approximation accuracy of LUT and the precision of data representation. In general, the LLGMN implemented in hardware can perform almost the same as the one realized in software.

In the training process, it was verified that 517 clock cycles were necessary for training of one sample datum, and a batch training [10] (on 200 training samples) requires 103488 cycles. As for discrimination phase, it costs 247 cycles for processing one datum. It was confirmed that a clock frequency of 20 MHz was realized.

IV. APPLICATION TO A BIOELECTRIC HUMAN INTERFACE SYSTEM

The FPGA implementation of PNNs is very meaningful for design of a compact human interface device, because an SoC design of stand-alone intelligent application becomes feasible with other relevant modules being integrated into the same FPGA. To verify this idea, the EMG-based pointing device called EMG-mouse [5], which is based on the LLGMN, was implemented on one FPGA chip.

The EMG-mouse system, as shown in Fig. 4, consists of three parts: (1) EMG signal processing, (2) Neural network, and (3) Pointer control. It provides an interface tool for wearable computers and mobile devices. In the EMG-mouse, the LLGMN is used to estimate probabilities of the operator's intended movement on a finite number of base directions. Then the pointer movement can be represented by combining these base directions (for details, see [5]). In this paper, all three parts are implemented on one FPGA chip. The EMG signals measured from four pairs of electrodes are amplified and digitized by an AD converter, and then input to the FPGA. The EMG-mouse system determines the direction of pointer's movement and its speed from the EMG signals, and the results are output to a display.

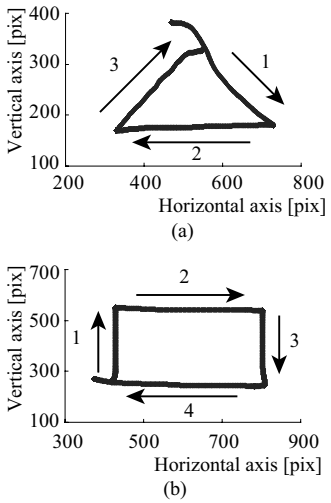

Fig. 5. Trajectories of pointer.

To examine the performance of the FPGA implementation of the EMG-mouse system, we conducted pointer control experiments. First, the LLGMN was trained with operator's EMG patterns corresponding to four base directions (up, right, down, and left). Then, the operator was asked to control the pointer to draw triangles/rectangles after the arrows' directions in the turn of the labels. Fig. 5 shows examples of the pointer trajectories. The detailed time histories of the drawing-a-triangle task, that is corresponding to Fig. 5 (a), is shown in Fig. 6. In this figure, the desired direction, the EMG signals, the posterior probability of each base direction output from the LLGMN module, the estimated movement direction, and the force level are plotted. The gray areas indicate that no movement was determined since the force level is less than a predefined threshold. It can be seen that, with the EMG-mouse system implemented on an

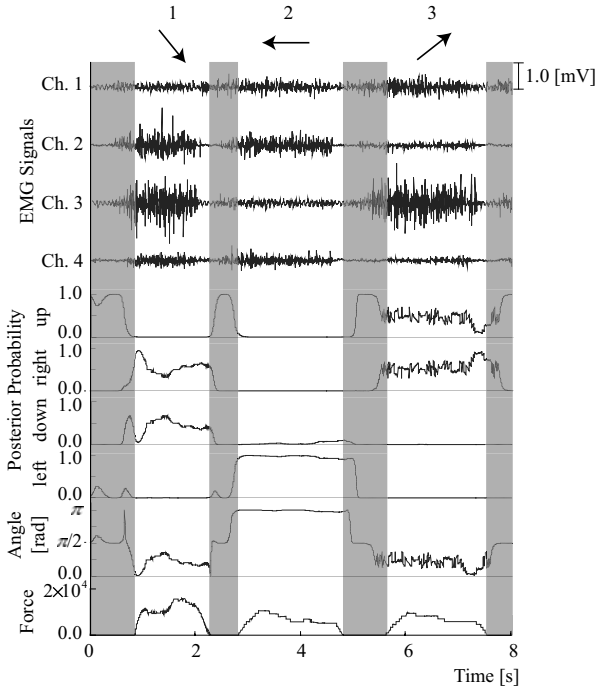

Fig. 6. An example of pointer control for drawing-a-triangle task.

FPGA chip, the pointer can be controlled approximately along the intended direction by the operator.

V. Conclusion

In this paper, we have discussed the FPGA implementation of the probabilistic NN, LLGMN, and a bioelectric human interface system based on it. It can be found that the FPGA implementation performs a comparable accuracy with the traditional solution based on a general-purpose computer. Furthermore, a human interface system realized on one FPGA chip provides much more portability, and is more compact in size. These features of the FPGA-based human interface system may be beneficial for wearable computers and other handheld devices. Because our research is still in its infancy, many efforts are needed for further development.

The LUT has been used to represent non-linear functions in the LLGMN. As mentioned in Section III, the LUT is recourse consuming, and there is a trade-off between the approximation accuracy of LUT and the chip area needed. This problem is non-trivial in cases of implementation of larger and more complicated neural networks, and more hardware efficient algorithms are required. The coordinate rotation digital computing (CORDIC) algorithm may be such a time and space efficient algorithm mainly used for calculation of the sine and cosine of a given angle. It can be also used for computing log, exponent and square root [11], [12].

In our research, the shortage of chip space is also due to the hardware description by the Handel-C language. Although Handel-C is friendly to system designers, it is observed that hardware designed using Handel-C is usually several times larger than the VHDL or Verilog-HDL implementation [13]. Therefore, incorporating intellectural property (IP) core in VHDL or Verilog-HDL language would be helpful.

In future research, we would like to develop the hardware implementation of LLGMN in a more compact size. Also, we would keep our concentration on the improvement of processing speed.

Acknowledgment

This work was partly supported by VLSI Design and Education Center(VDEC), the University of Tokyo in collaboration with Celoxica, Ltd. The support from New Energy and Industrial Technology Development Organization (NEDO) of Japan are also much appreciated.

References

[1] D.F. Specht, "Probabilistic Neural Networks," *Neural Networks*, vol. 3, no. 1, pp. 109-118, 1990.

[2] T. Tsuji, O. Fukuda, H. Ichinobe, M. Kaneko, "A Log-Linearized Gaussian Mixture Network and Its Application to EEG Pattern Classification," *IEEE Transactions on Systems, Man, and Cybernetics-Part C: Application and Reviews*, vol. 29, no. 1, pp. 60-72, 1999.

[3] G.D. Zhang, "Neural Network for classification: A Survey," *IEEE Transaction on Systems, Man and Cybernetics Part C*, vol. 30, no. 4, pp. 451-462, 2000.

[4] O. Fukuda, T. Tsuji and M. Kaneko, "An EMG Controlled Robotic Manipulator Using Neural Networks," *Proc. of IEEE International Workshop on Robot and Human Communication*, pp. 442-447, 1997.

[5] O. Fukuda, T. Tsuji, and M. Kaneko, "An EMG-Controlled Pointing Device Using a Neural Network," *Proc. of the 1999 IEEE Int. Conf. on Systems, Man, and Cybernetics*, vol. 4, pp. 63-68, Tokyo, 1999.

[6] O. Fukuda, T. Tsuji, M. Kaneko, and A. Ohtsuka, "A Human-Assisting Manipulator Teleoperated by EMG Signals and Arm Motions," *IEEE Trans. on Robotics and Automation*, vol. 19, no. 2, pp. 210-222, 2003.

[7] M. Porrmann, U. Witkowski, H. Kalte and U. Rückert, "Implementation of Artificial Neural Networks on a Reconfigurable Hardware Accelerator," *Proc. of 10th Euromicro Workshop on Parallel, Distributed and Network-based Processing*, pp. 243-250, Gran Canaria Island, Spain, 2002.

[8] D. Anguita, A. Boni, and S. Ridella, "A Digital Architecture for Support Vector Machines: Theory, Algorithm, and FPGA Implementation," *IEEE Trans. Neural Networks*, vol. 14, no. 5, pp. 993-1009, 2003.

[9] A. Bermak, and D. Martinez, "A Compact 3-D VLSI Classifier Using Bagging Threshold Network Ensembles," *IEEE Trans. Neural Networks*, vol. 14, no. 5, pp. 1097-1109, 2003.

[10] S. Haykin, "Neural Networks: A Comprehensive Foundation," Second Edition, Prentice Hall, 1998.

[11] R. Andraka, "A Survey of CORDIC Algorithms for FPGA based Computers," *Proc. of 1998 ACM/SIGDA 6th Int. Sym. on Field Programmable Gate Arrays*, pp. 191-200, Monterey, CA, 1998.

[12] A. Meyer-Bäse, R. Watzel, U. Meyer-Bäse, and S. Foo, "A parallel CORDIC architecture dedicated to compute the Gaussian potential function in neural networks," *Engineering Applications of Artificial Intelligence*, vol. 16, no. 7-8, pp. 595-605, 2003.

[13] S.M. Loo, B.E. Wells, N. Freije, and J. Kulick, "Handel-C for Rapid Prototyping of VLSI Coprocessors for Real Time Systems," *Proc. of 34th IEEE Southeastern Symposium on System Theory*, pp. 6-10, Huntsville, AL, 2002.