FPGA-based Online-learning using Parallel Genetic Algorithm and Neural Network for ECG Signal Classification

Yutana Jewajinda
National Electronics and Computer Technology Center
National Science and Technology Development Agency
Bangkok, Thailand
yutana.jewajinda@nectec.or.th

Prabhas Chongstitvatana
Department of Computer Engineering
Chulalongkorn University
Bangkok, Thailand
prabhas@chula.ac.th

Abstract— **This paper presents FPGA-based ECG signal classification based-on a parallel genetic algorithm and block-based neural network. The proposed parallel genetic algorithm has cellular-like structure which is suitable for hardware implementation. With online learning using hardware parallel genetic algorithm to block-based neural network, the complete ECG signal classification can be implemented in hardware. The proposed hardware can be implemented in FPGA or ASIC for a portable personalized ECG signal classifications for long term patient monitoring.**

I. INTRODUCTION

Optimizing the weights of artificial neural network (ANN) has been traditionally performed using training algorithm such as back propagation (BP) and conjugate gradient. BP has drawbacks due to its use of gradient descent so it often gets trapped in a local minimum of the error function. Evolutionary Algorithm (EA) has been proposed to optimize weights of ANNs. There is a class of ANN called evolutionary artificial neural network (EANN) that uses evolutionary algorithm to search for its parameters [1]. EANN employs the evolutionary algorithms to find important parameter of ANN such as weights, connections, learning rules.

Block-based neural network (BBNN) is an EANN that use evolutionary algorithm to tune its parameters [1]. The BBNN consists of a two-dimensional (2-D) array of basic neural-network blocks with integer weights. BBNN is suitable for hardware implementation especially using reconfigurable hardware such as field programmable logic arrays (FPGAs).

The numerous digital hardware implementation of artificial neural network (ANN) has been proposed over the years. However, among the plethora of ANN hardware, majority of those ANN hardware are implemented without online learning in hardware. Even some researchers present online hardware learning, most of them are based-on backpropagation algorithm. Genetic algorithms have been successfully proposed to optimize structure and weight of ANN. However, traditional GAs for ANN are implemented in software. Furthermore, GAs tend to take long time to reach the optimum solution. The hardware implementation of GAs can provide the solution to the time consuming optimization of GAs

This paper presents an online learning hardware based-on parallel genetic algorithm and block-based neural network for online ECG signal classification. The proposed evolvable hardware can be implemented in FPGA or ASIC for a portable personalized ECG signal classifications for long term patient monitoring.

The rest of this paper is organized as follows. Section II describes the parallel genetic algorithm and block-based neural network. In Section III, system architecture ECG signal classification is presented. Section IV presents classification results. The paper concludes with a summary in Section V.

II. BLOCK-BASED NEURAL NETWORK AND PARALLEL GENETIC ALGORITHMS.

The BBNN consists of a two-dimensional array with support integer weights. The BBNN structure is suitable to be implemented in hardware especially using field programmable logic arrays (FPGAs). However, in the past research, the genetic algorithm for BBNN was done in software off-line on computer system. With our proposed genetic algorithm in hardware, the whole system can be implemented in a single chip FPGA.

A. Elitism-based cellular compact genetic algorithm (EC-CGA)

Elitism-based cellular compact genetic algorithm (EC-CGA) is a parallel GA [2]. The EC-CGA consists of uniform cellular compact genetic algorithm cells connected in a cellular automata space. Each EC-CGA cell only exchange probability vectors to its neighbors as shown in Figure 1.

Figure 1 shows the pseudocode of the EC-CGA. Each cell of the EC-CGA has the identical algorithm. For each cell, one bit of the GA is represented by a probability vector. There are eight steps in the algorithm. For detail of the algorithm can be found in [4].

B. Hardware Architecture

The proposed architecture of EC-CGA can be scaled to the problem size with little modification to the hardware. The top level of our parallel hardware architecture is shown in Figure 1. From Figure 1, EC-CGA with four nodes is presented. Each EC-CGA node has four neighbors. Each node has to exchange data from the other four neighbors then it needs four input buffers. While at the output side, it needs only one output buffer because the same data from output of each node is passed to other nodes. The data from four input

buffer then passed to the 4:1 multiplexor to be selected by the main controller to the Bit Module block.

The proposed hardware architecture EC-CGA is based-on the architecture of a single EC-CGA cell. Each EC-CGA cell consists of four main blocks: Bit Module (BM), Package Switch Box (PSB), Fitness Evaluation (FE), and Main Controller (MC). Figure 2 shows the hardware architecture of each EC-CGA node. For more detail of hardware design of the EC-CGA can be found in [5].

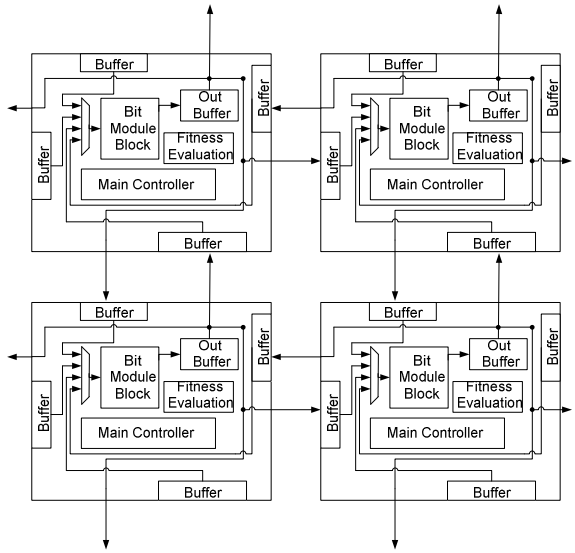

Fig. 1. Top level Hardware Architecture of four EC-CGA Cells

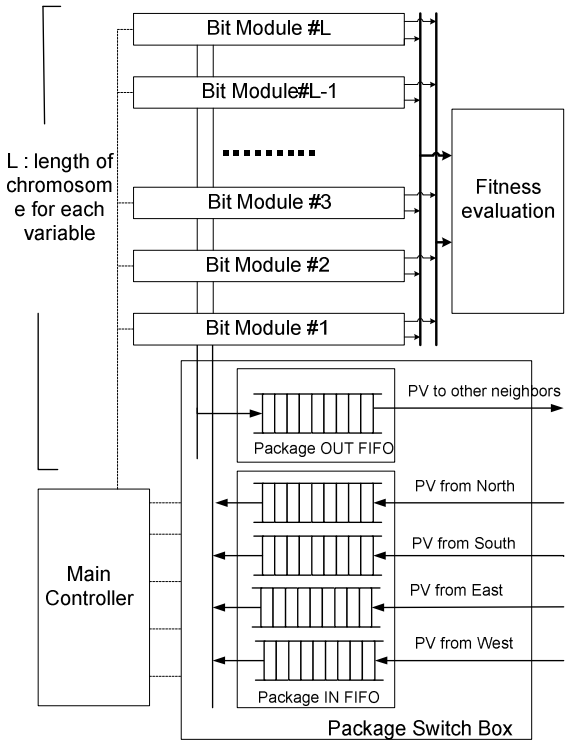

Fig. 2. Hardware Architecture of a EC-CGA Cell

L is chromosome length, N is population size
cc is Confident Counter, CA is Cellular Automata space
C is initial value of the probability vector

for each cell in CA **do in parallel**
  Initialize each p[i]
    For i := 1 to L do
      p[i] := C;
  Initialize cc
    cc := 0;
end parallel for
for each cell i in CA **do in parallel**
 while not done **do**
 1. Generate two individuals from the vector

```
if the first generation then
    E := generate(p);
    e := 0
    N := generate(p);
```

 2. Let them compete and update the cc counter

```
winner, loser := compete (E, N);
if winner != loser &
   winner != E       then
    cc  := cc + 1;
if e ≤ η then
  E := winner;
  e := e + 1;
else
  E := generate(p := C);
  e := 0;
```

 3. Update the probability vector toward better one and Increment Confidence Counter

```
for i := 1 to L do
if winner[i] != loser[i] then
if winner[i] == 1
then p[i] += 1/N
else p[i] -= 1/N
```

 4. Check if cc reaches a target level then Send p and cc to the neighboring cell

```
cc := 0;
```

 5. Receives p and cc from neighbors
 6. Use local search and the adaptive convex recombination with the received p from the neighbor and its own p
 6.1 Find the highest cc among neighbors' cc

```
cc_max := 0;
for i := 1 to M do
   if (cc[i] > cc_max )
       cc_max := cc[i];
p_ccmax := p[i]
```

 6.2 Convert cc_{max} to $\beta : 0 \le \beta \le 1$

```
β := 1 / cc_max ;
```

 6.3. Update p_1 with β

```
for i := 1 to L do
  p1[i] := β p1[i] + (1 - β) p_ccmax[i]
```

 7. Check if the vector has converged

```
for i := 1 to L do
   if p[i] > 0 and p[i] < 1 then
      goto step 1
```

 8. p represents the final solution
end while
end parallel for

Fig. 3. Top level hardware architecture 4-node EC-CGA

C. *Block-based neural network*

The block-based neural network (BBNN) was proposed by S. W Moon et al in [1]. The BBNN model consists of a 2-D array of basic blocks. Each block is a basic processing element that corresponds to a feedforward neural network with four variable input/output. BBNN can implement both

feedforward and feedback network configuration. A block of BBNN consists of four nodes. These four nodes can be represented by one of the three different types of internal configurations. Figure 4 shows four types of a BBNN cell.

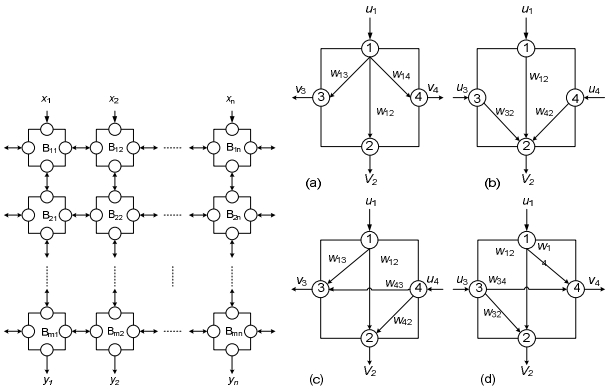

Fig. 4 BBNNs (a) structure (b) Four different internal configurations of a BBNN block (a) 1/3, (b) 3/1, (c) 2/2, (d) 2/2

D. The Layer-based Architecture for integration of CCGA and BBNN

We propose the layer-based architecture for the block-based neural network and the cellular compact GA [3]. From Figure 5, the BBNN occupies different layer from the cellular compact GA. These two layers interact to each other between nodes of BBNN and CCGA. Figure 5 shows the layer-based architecture with the eight nodes for each BBNN and CCGA layers. If l is the problem size that is the number of node of the BBNN, with n_p CCGA nodes, then each CCGA node contains l / n_p probability vectors.

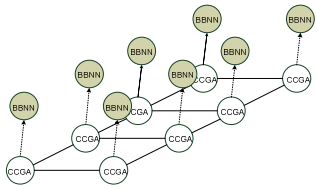

Fig. 5 Layer-multiplexed architecture of BBNN layer and 4x2 CCGA nodes

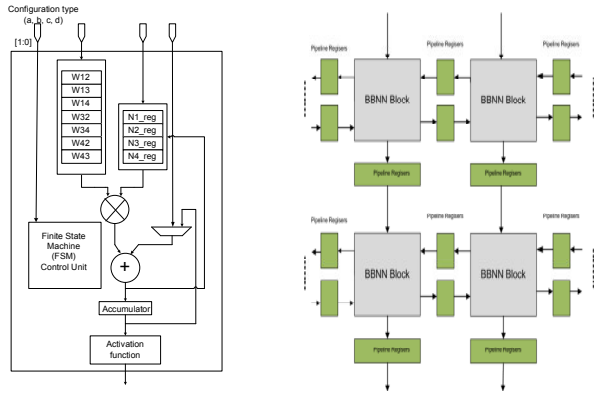

Fig. 6 (a) Hardware design of a BBNN block (b) 4 BBNN blocks connected

III. SYSTEM ARCHITECTURE FOR ECG SIGNAL CLASSIFICATION

The proposed system for on-line ECG beat recognition approach is shown in Figure 7. It consists of two step processes: feature selection and classification. The feature selection consists of QRS detection and Hermite basis functions coefficient extraction, delivering these coefficients as the features to the input of classification process. The classification process uses block-based neural network (BBNN) learning by elitism-based cellular genetic algorithm (EC-CGA). In this paper, the feature selection process was carried on by our own functions created in the Matlab software for which suitable solving Hermite polynomial and performing QRS window construction. Also, this feature selection process can be done using a soft-processor or embedded processor in FPGA. For the classification process, the design was coded in Verilog HDL and simulated using Model Sim simulator and synthesized by Xilinx ISE software.

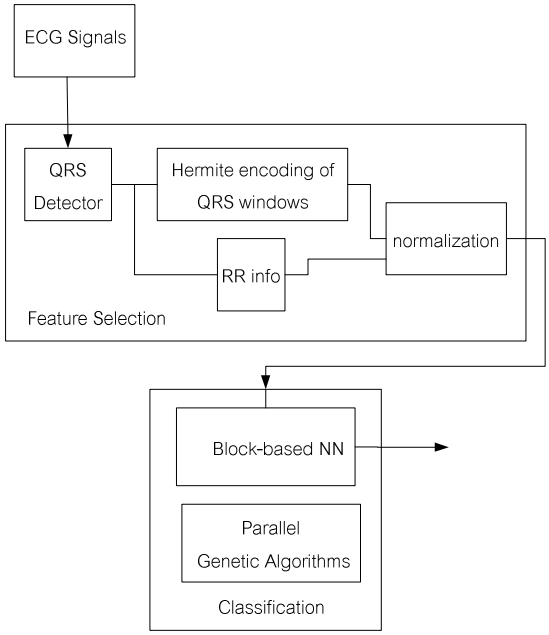

Fig. 7. The proposed methodology

AAMI recommends that each ECG beat be classified into the following five heartbeat types: beats originating in the sinus node, supraventricular ectopic beats, ventricular ectopic beats, fusion beats, and unclassifiable beats [6].

A. ECG Dataset

The MIT-BIH arrhythmia database [6] is used in the experiment. The database contains 48 records obtained from 47 different individuals (two records came from the same patient). Each record contains two-channel ECG signals measured for 30 min. The data are bandpass filtered at 0.1–100 Hz and sampled at 360 Hz. There are over 109 000 labeled ventricular beats from 15 different heartbeat types. The largest class is “Normal beat” (NORMAL) with over 75 000 examples and the smallest class is “Supraventricular premature beat” (SPC) with just two examples.

In agreement with the AAMI recommended practice, the four recordings containing paced beats were removed from the analysis. The remaining recordings were divided into two datasets with each dataset containing ECG data from 22 recordings shown in Fig. 8.

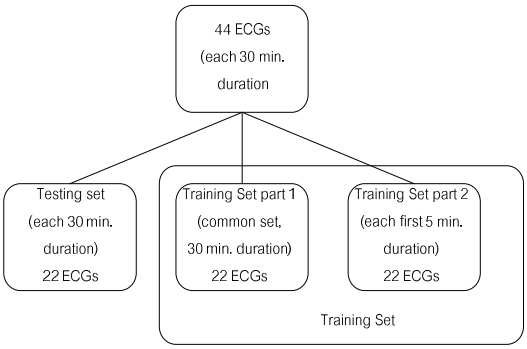

Fig. 8. Training and testing sets

Table 1: Records in training and testing sets from the MIT-BIH arrhythmia database

Records in Testing Set	Records in Training Set
101, 103, 105, 111, 112, 113, 117, 121, 122, 123, ,200, 202, 210, 212, 213, 214, 219, 221, 222, 228, 231, 234	100, 106, 108, 109, 114, 115, 116, 118, 119, 124, 201, 203, 205, 207, 208, 209, 215, 220, 223, 230, 232, 233

B. Feature Selection

Basis function representations have been shown to be an efficient feature extraction method for ECG signals [6, 9]. Hermite basis functions provide an effective approach for characterizing ECG heartbeats and have been widely used in ECG signal classification [7, 9]. Hermite basis function expansion has a unique width parameter that is an effective parameter to represent ECG beats with different QRS complex duration. The coefficients of Hermite expansions characterize the shape of QRS complexes and serve as input features. Let us denote *x(t)* be the discrete time QRS complex of ECG curve. The expansion of *x(t)* into Hermite series may be presented in the following way:

$$x(t) = \sum_{n=0}^{N-1} a_n \emptyset_n(t,\sigma) \qquad (1)$$

where a_n (n = 0, 1, 2, …, N – 1) are the expansion coefficients while $\emptyset_n(t,\sigma)$ is the Hermite basis function defined as

$$\emptyset_n(t,\sigma) = \frac{1}{\sqrt{\sigma 2^n n! \sqrt{\pi}}} e^{-t^2/2\sigma^2} H_n(t/\sigma) \qquad (2)$$

The functions $H_n(t/\sigma)$ are the Hermite polynomials. With $H_0(x) = 1$ and $H_1(x) = 2x$, the Hermite polynomials are defined recursively by

$$H_n(x) = 2xH_{n-1}(x) - 2(n-1)H_{n-2}(x) \qquad (3)$$

The approximation error depends on the number of coefficients. Five Hermite functions allow a good representation of the QRS complexes and fast computation of the coefficients [7, 9]. Besides the basis function coefficients and width parameter , the time interval between two neighboring R-peaks is included to discriminate normal and premature heart beats.

C. Classification Process

For the classification process, the two nodes of combined BBNN and EC-CGA are used to perform the experiment. Each node consists of 7x2 BBNN and two EC-CGAs. In BBNNs, the number of columns is equal to or greater than the number of input features. For EC-CGA, the first EC-CGA is for topology optimization since it requires only 2-bit per BBNN node which is 16x2 bits for this one combined 7x2 BBNN and EC-CGA. The second EC-CGA is for weight optimization which supports fixed-point 15-bit fraction number precision. The optimization of problems for the second EC-CGA is the size of 140 variables since each node of BBNN requires 10 parameters to be optimized. Each variable has the precision of 15-bit, so that it's the problem of total chromosome length of 2100 bit for each of the EC-CGA that find the weights of BBNN. Fitness function evaluates the quality of the solutions. The fitness of an individual BBNN is defined as

$$Fitness = \frac{\beta}{1 + \frac{1}{n_0 M_1}\sum_{l=1}^{M_1}\|d_c^l - y_c^l\|} + \frac{1-\beta}{1 + \frac{1}{n_0 M_2}\sum_{l=1}^{M_2}\|d_c^l - y_c^l\|}$$

where M_1 and M_2 are the numbers of samples in the common and patient-specific training data, respectively.n_0 donote the number of output blocks. β controls relative importance of the common and patient-specific training in the final fitness, and a small value less than 0.5 (0.2 in this paper) gives more weight for correctly classifying patient-specific patterns same as in [9]. Both common and patient-specific training patterns are considered in the fitness function. While patient-specific data may serve as the training data for evolving BbNN specific to a patient, the inclusion of common training data is useful when the small segment of patient-specific samples contains few arrhythmia patterns [9].

IV. CLASSIFICATION RESULTS

Figure 9 shows the results of the training experiment. The dotted and solid line corresponds to using 4-Node CCGA with communicating between CCGA nodes and without communicating between CCGA nodes. The evolution stops when the desired fitness is met after approximately 2200 generations. From Figure 9, the solid line indicates the

occurrences of near-optimum structure and weights. The others represent four non-optimal. The 4-node CCGA with communicating between nodes perform significantly better than then non-communicating one.

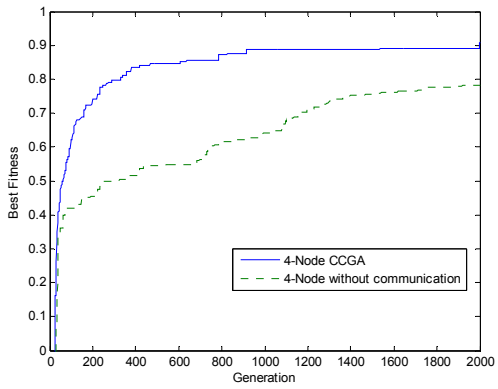

Fig. 9 Fitness and number of generation.

Table 2: Summary of beat-by-beat classification results for the five classes.

Truth	Classification Result				
	N	S	V	F	Q
N	23049	340	256	10	0
S	603	641	48	1	0
V	232	86	1978	7	1
F	71	37	105	107	0
Q	4	0	2	1	0

Table 2 summarizes classification results of ECG heartbeat patterns for test records.

Two sets of performance are reported: the detection of VEBs and the detection of SVEBs, in accordance with the AAMI recommendations. Four performance measures, classification accuracy (Acc), sensitivity (Sen), specificity (Spe), and positive predictivity (PP), are defined in the following using true positive (TP), true negative (TN), false positive (FP) and false negative (FN).

For VEB detection, the sensitivity was 85.8%, the specificity was 98.7%, the positive predictivity was 86.6%, and the overall accuracy was 97.7%. For SVEB detection, the sensitivity was 49.6%, the specificity was 98.2%, the positive predictivity was 58.06%, and the overall accuracy was 96.05%. From the results, the performance of SVEB detection is not as good as VEB detection, and the possible reasons include the more diverse types in class and the lack of class training patterns in patients [2], [5].

Table 3 summarizes sensitivity, specificity, positive predictivity, and overall accuracy of the four methods. The proposed method overall outperforms the methods in [7]. The method in [9] performs slightly better than the proposed method.

Table 3: Performance comparison of VEB and SVEB Detection (in percent)

Method	VEB				SVEB			
	Acc	*Sen*	*Spe*	*+P*	*Acc*	*Sen*	*Spe*	*+P*
Hu *et. al.* [7]	94.8	78.9	96.8	75.8	N/A	N/A	N/A	N/A
Chazal *et. al.*[8]	96.4	77.5	98.9	90.6	92.4	76.4	93.2	38.7
Jiang *et. al.* [9]	98.8	94.3	99.4	95.8	97.5	74.9	98.8	78.8
Proposed	97.7	85.8	98.7	86.6	96.0	49.6	98.2	58.1

V. Conclusion

The proposed technique is similar to the technique in [9]. However, in [9] the evolution algorithm with back-propagation learning is used for BBNN learning. Our purposed technique only uses the proposed EC-CGA for BBNN optimization. By using only EC-CGA, our proposed method is suitable for hardware implementation and by estimation, can provide speed up of over 100 times to software version in [9]. Our proposed method with four EC-CGA nodes has some disadvantage in comparison with work in [8, 9]. This is due to the precision of hardware implementation and the limited number of EC-CGA nodes. With our proposed solution, the complete hardware solution of ECG signal classification can be implemented in hardware if the wavelet transform is used for feature selection.

References

[1] Moon, S. W., and Kong, S. G. (2001). Block-based neural networks. IEEE Transaction on Neural Networks 12:307-317.

[2] E. Cantu-Paz, *Efficient and accurate parallel genetic algorithms*. Boston, MA: Kluwer Academic Publisher, 2000.

[3] Y. Jewajinda and P. Chongstitvatana, "FPGA Implementation of a Univariate Estimation of Distribution Algorithm and Block-based Neural Network as An Evolvable Hardware", IEEE World Congress on Computational Intelligence, Hong Kong, June, 2008.

[4] Y. Jewajinda and P. Chongstitvatana, "FPGA Implementation of a Cellular Genetic Algorithm, NASA/ESA international conference on Adaptive hardware and Systems (AHS-2008), Netherland, July, 2008.

[5] Y. Jewajinda and P. Chongstitvatana, "Hardware Architecture and FPGA Implementation of a Parallel Elitism-based Compact Genetic Algorihm," *Proc. IEEE TENCON*, 2009.

[6] Mark R. and Moody G. MIT-BIH Arrhythmia Database Directory [Online]. Available from: www.physionet.org

[7] Linh, T. H., Osowski, S., and Stodolski, M. (2003). On-line heart beat recognition using Hermite polynomials and neuro-fuzzy network. IEEE Transaction on Instrument and Measurement, 52: 1224–1231.

[8] Hu, Y., Palreddy, S., and Tompkins, W. J. (1997). A patient-adaptable ECG beat classifier using a mixture of experts approach. IEEE Transaction on Biomedical Engineering, 44: 891–900.

[9] Chazal, P. de., O'Dwyer, M., and Reilly, R. B. (2004). Automatic classification of heartbeats using ECG morphology and heartbeat interval features. IEEE Transaction on Biomedical Engineering, 51: 1196–1206

[10] W. Jiang, Moon, S. W., and Kong, S. G. (2007). Block-Based Neural Networks for Personalized ECG Signal Classification. IEEE Transaction on Neural Networks 18: 1750-1761.