Low-Power Hardware Implementation of a Support Vector Machine Training and Classification for Neural Seizure Detection

Heba Elhosary, Michael H. Zakhari, Mohamed A. Elgammal, Mohamed A. Abd El Ghany, Khaled N. Salama, *Senior Member, IEEE*, and Hassan Mostafa, *Senior Member, IEEE*

***Abstract*—In this paper, a low power support vector machine (SVM) training, feature extraction, and classification algorithm are hardware implemented in a neural seizure detection application. The training algorithm used is the sequential minimal optimization (SMO) algorithm. The system is implemented on different platforms: such as field programmable gate array (FPGA), Xilinx Virtex-7 and application specific integrated circuit (ASIC) using hardware-calibrated UMC 65 nm CMOS technology. The implemented training hardware is introduced as an accelerator intellectual property (IP), especially in the case of large number of training sets, such as neural seizure detection. Feature extraction and classification blocks are implemented to achieve the best trade-off between sensitivity and power consumption. The proposed seizure detection system achieves a sensitivity around 96.77% when tested with the implemented linear kernel classifier. A power consumption evaluation is performed on both the ASIC and FPGA platforms showing that the ASIC power consumption is improved by a factor of 2X when compared with the FPGA counterpart.**

***Index Terms*—Accelerator IP, ASIC, classification, feature extraction, FPGA, low power, sequential minimal optimization (SMO), support vector machine (SVM).**

Manuscript received July 11, 2019; revised September 2, 2019 and October 2, 2019; accepted October 5, 2019. Date of publication October 14, 2019; date of current version December 31, 2019. This work was supported in part by ONE Lab at Zewail City of Science and Technology and at Cairo University, NTRA, ITIDA, and ASRT. This paper was recommended by Associate Editor H. Jiang. *(Heba Elhosary and Michael H. Zakhari contributed equally to this work.) (Corresponding author: Hassan Mostafa.)*

H. Elhosary is with the Department of Electronics, German University in Cairo, New Cairo 11511, Egypt (e-mail: heba.diaa.elhosary@gmail.com).

M. H. Zakhari and M. A. Elgammal are with the Department of Electronics and Communications Engineering, Cairo University, Giza 11114, Egypt (e-mail: michael.hany.mofeed@gmail.com; mohamed.adel567@gmail.com).

M. A. Abd El Ghany is with the Department of Electronics, German University in Cairo, New Cairo 11511, Egypt, and also with the Integrated Electronic Systems Lab, Technische Universität Darmstadt, Darmstadt 64289, Germany (e-mail: moh_salim@hotmail.com).

K. N. Salama is with King Abdullah University of Science and Technology, Thuwal 23955, Saudi Arabia (e-mail: khaled.salama@kaust.edu.sa).

H. Mostafa is with the Department of Electronics and Communications Engineering, Cairo University, Giza 11114, Egypt, and also with the University of Science and technology, Nanotechnology and Nanoelectronics Program, Zewail City of Science and Technology, Giza 12578, Egypt (e-mail: hmostafa@uwaterloo.ca).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TBCAS.2019.2947044

I. Introduction

EPILEPSY is a neurological disorder characterized by recurrent seizures as a result of abnormal electrical discharges in the brain. The condition affects approximately 1–2% of the world' s population [1]. The sudden and unpredictable nature of seizures is one of the most disabling aspects of epilepsy disease [2]. Anti-epileptic drugs (AEDs) are used to affect the brain's chemistry to reduce epilepsy. AEDs' main disadvantage is that they are utterly experimental; AED type and concentration are adopted for each case individually [3]. Electrical stimulation is used to overcome this problem by reducing the effects of epilepsy. The stimulation is applied only when a seizure takes place. Accordingly, seizure detection and prediction become very important. In this paper, a hardware implemented system that uses EEG signals to aid seizure detection is proposed.

EEG signal processing is widely used for assessing disorders of brain function and; especially for epilepsy diagnosis. The traditional method used to identify seizures is heavily dependent on visual analysis of EEG recordings by trained professionals. It is a very costly and tedious task to review 24 hours of continuous EEG recordings, particularly as the number of EEG channels increases [4]. Therefore, automatic seizure detection systems using machine learning (ML) have evolved.

Recently, ML and artificial intelligence (AI) have become very hot topics for all software and hardware researchers. ML is the science of teaching computers how to deal with different situations and to perform some tasks without being programmed. It plays a significant role in many fields. SVM is gaining much attention among researchers for statistical classification and regression analysis problems. SVM performs well on various problems such as pattern recognition, face detection, handwritten recognition, and bio-informatics [6]. Consequently, SVM is chosen in binary classification problems as the proposed automatic seizure detection with proven excellent accuracy as stated in [4], [7], [8]. In [9], testing of different machine learning algorithms has been conducted and the performance metrics -RMSEs, average accuracy rates, and average training time- have been evaluated. In this work, by following the said testing method, the algorithm with the best performance has been determined to be SVM. Linear SVM is chosen for classification as higher degrees of kernel function SVMs are more accurate on the expense of higher computational complexity,

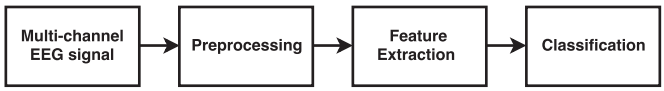


Fig. 1. Block Diagram of the Seizure Detection System [5].

and accordingly higher power consumption [10]. In the proposed system, Low power consumption and area utilization are the main objectives and correspondingly, high computational complexity is avoided. Automatic seizure detection methods consist of five main phases as shown in Figure 1. First, EEG signals are measured through electrodes, and several studies in the literature have been investigating techniques to improve the EEG measurement efficiency [11]. Then, a preprocessing phase is conducted in which the removal of unwanted aspects, such as artifacts and high-frequency content, and a normalization of the EEG data is carried out. After that, many features are extracted from the EEG signal' s time domain [12], frequency domain, and time-frequency (Wavelet) domain [4]. Selecting discriminating features that best represent the characteristics of the EEG signals is key to the overall performance of the seizure detection system [13], [14].

A training Phase is then executed with these extracted features to create a hyper-plane that separates two labeled sets of training examples. Following that, a classifier is used with the extracted features to detect seizures and to classify unlabeled testing examples into one of the two classes [15].

In the training phase, SVM searches for the hyper-plane that gives the largest margin between the two sets. Finding the hyper-plane requires solving a quadratic programming (QP) problem subject to constraints [16]. The size of the QP problem is directly related to the number of training data points. For example, if a data set has a huge number of training samples, the QP problem becomes very complex and energy consuming. Correspondingly, having dedicated hardware for SVM training results in reducing the training time and accelerating the system. Besides, having a very low-power training accelerator chip is useful to tune the system parameters as seizure pattern differs from one patient to the other [17]. Moreover, having a low power detection system gives the implemented system longer battery lifetime.

Much research has been done on implementing hardware accelerators for SVM training [18]. Keerthiet *et al.* [19] propose a parallel implementation of multiple CPUs for processing partitioned data sets. The use of multiple CPUs leads to an increase in overall performance, but on the other hand, the power consumption is significantly increased. Caoet et al. [20] developed a hardware implementation of an SVM training circuit using MATLAB HDL coder. Performance degrades due to lack of optimizations. Chih-Hsiang et al. [21] propose a reconfigurable chip with SMO-based SVM training. The proposed architecture decreases the routing overhead, accelerates the kernel function update and uses pipelining. However, multiple hardware resource utilization and training speed problems are reported. Lazaro et al. [22] propose a hardware-software architecture to speed up SVM training using sequential minimal optimization (SMO). As the dot product takes up most of the calculation time in SMO, it was chosen to be implemented on the hardware. Jhing-Fa et al. [23] propose a hardware-software co-design solution for multiclass SMO training. A hardware-software co-design system for accelerating the SVM learning phase is presented based on another decomposition algorithm instead of the common SMO algorithm [25]. Rabieah et al. [26] propose a complete FPGA-based system for nonlinear SVM learning using ensemble learning. Wang et al. [27] propose an FPGA-based reconfiguration framework to speed up the online least square support vector machine (LS-SVM) training. However, the block RAM utilization and the reconfiguration efficiency are the main challenges. In this paper, more work is done in the area of implementing SVM training accelerators to produce better results without the need for complex transformations or complex kernel functions such as those proposed in [28], [29], and [30]. In addition, the proposed work performs the hardware implementation of the SVM using FPGA implementation and ASIC implementation to show the energy gap between both technologies. Moreover, experimental results are reported for the FPGA implementation conducted in this work.

The rest of the paper is organized as follows. Section II articulates the feature extraction strategies. Section III provides background on the SVM learning and the SMO algorithm. Section IV provides a detailed description of the proposed hardware implementation of SMO training accelerator. Section V discusses the SVM classifier hardware implementation and basic operation. Section VI articulates the optimizations implemented in the system to achieve lower power consumption and less area utilization along with the associated performance degradation. Section VII arrays the software simulation results, the FPGA and ASIC implementation results and provides analysis and comparison of the results. Finally, the conclusion is drawn in Section VIII.

II. Feature Extraction

In this section, a combination of linear and nonlinear features, namely, Fractal dimension [34], Hurst exponent [35], and Coastline [36] chosen for implementation is discussed in details. The selection of features is based on the best performing features acquired by [37]. Twenty linear and non-linear features are implemented, and a total of 1140 combination of features are tested along with linear SVM and the best performing combination is obtained.

A. Fractal Dimension

It measures the complexity of the input EEG signal over multiple scales. In other words, it is a measure of how many times a pattern can be found in a signal. Higuchi' s algorithm with k= 5 is used to calculate the fractal dimension.

$$L_m(k) = \frac{\sum_{i=1}^{\frac{N-m}{k}} \frac{|x(m+ik)-x(m+(i-1)*k)|}{N-1}}{D-m} \tag{1}$$

$$FD = \sum_{m=1}^{k} \frac{ln(L_m(k))}{ln(\frac{1}{k})} \tag{2}$$

Where x is a time series that consists of D data points, and m is a constant that ranges from 1 to k. The input is first fed to one of five accumulators depending on its index. The output of each accumulator is divided by $D - m$ resulting L_m. The final value of the FD is the summation of the five values resulting from dividing the natural logarithms of L_m by the natural logarithm of 1/k.

B. Hurst Exponent

It is a technique that quantifies the meaningfulness of the input signal. If the output value is in the range from 0.5 to 1, then the input EEG signal contains meaningful patterns, on the other hand, if the output value equals 0.5, it is identified as noise.

$$\begin{aligned} R &= ||\text{MAX}(X - \text{MAV})| - |\text{MIN}(X - \text{MAV})|| \\ S &= \text{STD}(X) \\ H &= \frac{Ln(\frac{R}{S})}{Ln(T)} \end{aligned} \tag{3}$$

Where x is a time series that consists of D data points, MAV is the mean absolute value, R is the range of the cumulative deviation from the mean, S is the standard deviation, and T is a constant $= \frac{1}{256}$.

C. Coastline

It quantifies the number of fluctuations in the given epoch. A seizure is recurrent discharges in brain neurons, which means a higher frequency of fluctuations than the normal case.

$$CL = \sum_{i=1}^{D} |x_{i+1} - x_i| \tag{4}$$

where X_i is the i_{th} data point in a designated window, D is the number of data points in the designated window.

III. SVM Learning

This section presents more details on the SVM algorithms and techniques. SVM is a widely used classification technique as it finds the hyper-plane with the largest margin. SVM was first introduced by Vladimir N. Vapnik *et al.* in 1963 [31].

SVM learns from a training set of N-dimensional vectors x_i where N is the number of features extracted and their associated classes (labels) $y_i \in \{1, -1\}$. It deals with linearly separable data points directly. The non-linearly separable data points are mapped into a higher dimensional domain, in which the mapped data points become linearly separable. Thus, SVM finds the maximal margin hyper-plane in the new domain. The hyper-plane is defined by the following equation:

$$w.\Phi(x) + b = 0 \tag{5}$$

where w is the normal to the hyper-plane, $\Phi(x)$ is the mapping function used to map each input vector to the feature space, and b is the bias.

The distance from the nearest points to the hyper-plane from each side equal $\frac{2}{||w||}$. Therefore, the optimization problem is formulated as follows:

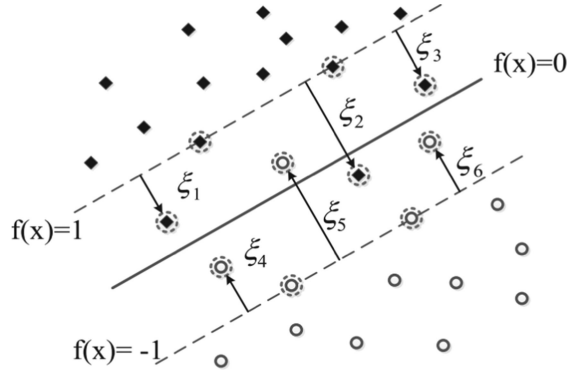


Fig. 2. Soft-margin SVM [32].

$$\min_{w,b} \frac{||w||^2}{2} \tag{6}$$

subject to $y_i(w.\Phi(x) + b) \geq 1$.

This is denoted by a hard margin SVM, where the hyper-plane perfectly separates the two sets according to (5). A modified version of SVM introduces a trade-off between the size of the margin and the number of errors in the classification process, as in (7). This is performed by defining a penalty parameter C. The optimization problem is formulated as:

$$\min_{w,b} \frac{||w||^2}{2} + C.\Sigma\xi_i \tag{7}$$

subject to $y_i(w.\Phi(x) + b) \geq 1 - \xi_i, \xi_i \geq 0$.

where ξ_i is the slack for the i_{th} training point as shown in Figure 2. The penalty parameter C should be selected carefully for each data set. If C is selected large, the weight of any wrongly classified point is considerable, so the convergence of the problem requires a high number of iterations. If C is selected small, some errors are allowed to maximize the margin and obtain the solution in fewer iterations than in the large C scenario.

The modeled problem is solved using Lagrange multiplier as follows:

$$\min_{\alpha} \psi(\alpha) = \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} y_i y_j K(x_i, x_j) \alpha_i \alpha_j - \sum_{i=1}^{N} \alpha_i \tag{8}$$

subject to $\sum_{i=1}^{N} y_i \alpha_i = 0, 0 \leq \alpha_i \leq C$, and $i = 1, \ldots, n$

where α is a Lagrange multiplier, and kernel functions K as stated in [10] may be linear, polynomial, exponential as follows:

Linear kernel:

$$K(x_i, x_j) = x_i.x_j \tag{9}$$

Polynomial kernel:

$$K(x_i, x_j) = (x_i.x_j + 1)^d \tag{10}$$

where d is the polynomial degree.

Exponential kernel:

$$K(x_i, x_j) = exp(-\gamma||x_i - .x_j||^2) \tag{11}$$

TABLE I
PSUEDO CODE OF SEQUENTIAL MINIMAL OPTIMIZATION ALGORITHM

initial	$w = 0,\ \alpha = 0,\ b = 0,\ E = 0$
iterate till convergence	
	select i and j
	read x_i and x_j from memory
	calculate Kernel functions $k_{ii},\ k_{ij},\ k_{jj}$
	calculate the errors$E_i,\ E_j$
	calculate the limits L and H
	$\eta = 2k_{ij} - k_{ii} - k_{jj}$
	$\alpha_j^{new} = \alpha_j^{old} + \frac{y_j(E_j^{old} - E_j^{new})}{\eta}$
	$\alpha_i^{new} = \alpha_i^{old} + y_i y_j(\alpha_j^{old} - \alpha_j^{old,clipped})$
	calculate the bias b
	check for convergence

All the training points are classified into three classes:

1) $\alpha_i = 0$ represents the correctly classified points outside the margin
2) $0 < \alpha_i < C$ represents the points that define the margin
3) $\alpha_i = C$ represents the wrongly classified points

The optimum hyper-parameters are trained using MATLAB and then evaluated in two stages coarse and fine trainings. In the coarse training stage, the starting point is a rough estimate of the parameters and then the performance is evaluated. In the fine training stage, continuing from the initial estimate, the parameters are tuned finely and the performance is measured for each small change until the best combination of hyper-parameters are achieved. This is done by evaluating the performance sensitivity to each hyper-parameter change.

SMO Algorithm

The SMO algorithm was introduced and explained by John Platt in [16]. The main idea of the SMO technique is to break any large QP problem into multiple smaller ones. It solves the constrained quadratic programming problem efficiently as it iteratively narrows the optimization problem to just two Lagrange multipliers in each iteration. The selection of the two Lagrange multipliers to be optimized in each iteration is performed heuristically. However, depending on the application, the SMO algorithm scales somewhere between linear and quadratic with the number of the data training set.

The SMO algorithm optimizes the objective function by jointly optimizing two Lagrange multipliers. The fact that optimizing two Lagrange multipliers is performed analytically makes the SMO algorithm advantageous. The algorithm is summarized in Table I.

The SMO algorithm starts by selecting two Lagrange multipliers to optimize the objective function and calculates their bounding values. The bounding values of only two Lagrange multipliers are depicted in a 2-D square as in Figure 3. The square sides represent the maximum and the minimum values of the Lagrange multipliers while the diagonal line represents the values the multipliers are allowed to take.

Denoting the two Lagrange multipliers by: α_1 and α_2, it is required to obtain new values for the multipliers, α_1^{new}, α_2^{new}, from the old set of all Lagrange multipliers $\{\alpha_1^{\text{old}}, \alpha_2^{\text{old}}, \alpha_3, \alpha_4 \ldots, \alpha_N\}$, where α_1^{old}, α_2^{old} have the initial

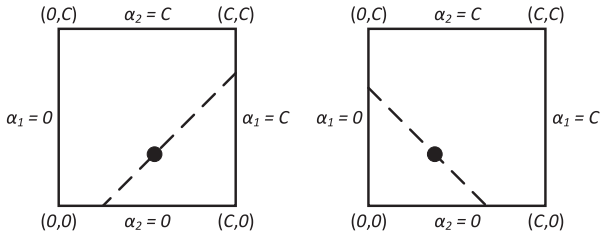


Fig. 3. The bounding values of two Lagrange Multipliers. On the left, the Bounding Square when $y_1 \neq y_2$ hence, $\alpha_1 - \alpha_2 =$ constant. On the Right, the Bounding Square when $y_1 = y_2$ hence, $\alpha_1 + \alpha_2 =$ constant.

value zero. Given the constraint equation $\sum_{i=1}^{N} \alpha_i.y_i = 0$, the following condition is derived:

$$y_1.\alpha_1^{\text{new}} + y_2.\alpha_2^{\text{new}} = y_1.\alpha_1^{\text{old}} + y_2.\alpha_2^{\text{old}} \tag{12}$$

Following the derivations in [16], α_j^{new} is obtained by:

$$\alpha_j^{\text{new}} = \alpha_j^{\text{old}} + \frac{y_j(E_j^{\text{old}} - E_j^{\text{new}})}{\eta} \tag{13}$$

where $k_{ii} = x_i^T.x_i,\ k_{jj} = x_j^T.x_j,\ k_{ij} = x_i^T.x_j,\ \eta = 2.k_{ij} - k_{ii} - k_{jj}$, and $E_i = w^T x_i - b - y_i$.

Referring to the constraints depicted in Figure 3, α_j^{new} is clipped to be in the feasible range. Therefore, $\alpha_j^{\text{new, clipped}}$ is obtained by:

$$\alpha_j^{\text{new, clipped}} = \begin{cases} H, & \alpha \geq H \\ \alpha_j^{\text{new}}, & L < \alpha_j^{\text{new}} < H \\ L, & \alpha_j^{\text{new}} \leq L \end{cases} \tag{14}$$

And therefore α_i^{new} is calculated as follows:

$$\alpha_i^{\text{new}} = \alpha_i^{\text{old}} + t\left(\alpha_j^{\text{old}} - \alpha_j^{\text{old, clipped}}\right) \tag{15}$$

where $t = y_i.y_j$.

IV. HARDWARE IMPLEMENTATION OF THE SMO TRAINING ACCELERATOR

In this section, the full hardware implementations of the proposed SVM training and the SMO algorithm are described in detail. Different approximate computing techniques are used in implementing the proposed SMO accelerator to reduce power consumption. First of all, a fixed point implementation is used instead of the computationally expensive floating-point. Using software simulation results, it is found that a 16-bit word length is enough to achieve acceptable performance for both techniques (i.e., accuracy and sensitivity). Reducing the word length to less than 16 bits achieves more power saving at the cost of performance degradation. At a certain word length, the full dynamic range of the bits should be used to achieve the highest accuracy for this configuration, which requires a smart selection of the integer and fraction portions of the fixed point word length. Second, computation skipping is used in different steps in the two algorithms (i.e., multiplying by zero is skipped).

Finally, inaccurate arithmetic techniques are adopted in the implementation. Using inaccurate arithmetic operations introduces some errors which are acceptable in a specific range, but

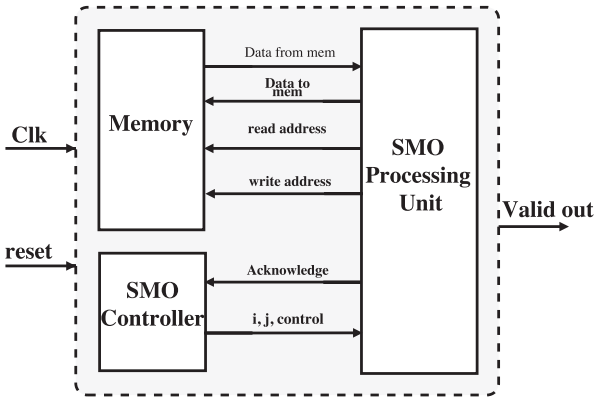


Fig. 4. SMO Algorithm Training Architecture.

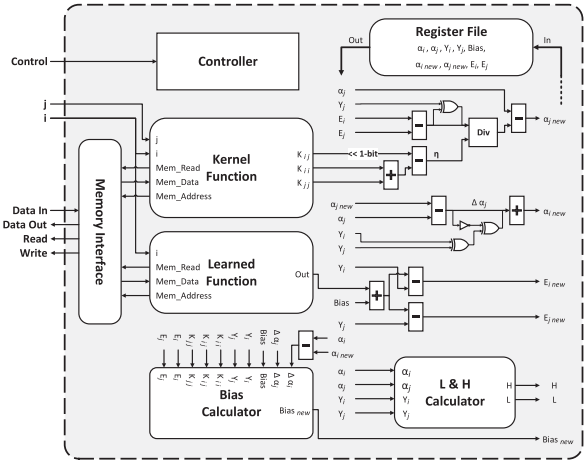


Fig. 5. SMO Processing Unit.

using them reduces the system power and energy consumption significantly. As multipliers are one of the most power-hungry blocks, the signed truncated multiplier proposed in [33] is utilized. The signed truncated multiplier consumes less power than accurate multipliers by summing an optimized partial products matrix (PPM). A truncated accumulation is used rather than an accumulation of the whole output of the multiplier (i.e., the output of the multiplier is truncated to the specified word length, and then the accumulation operation is performed). This also reduces the size/power of the required accumulator and has a small impact on accuracy.

SMO Training Architecture: To keep the architecture generalized for any heuristic model of selecting a Lagrange multiplier, the SMO training architecture is divided into three main blocks: the SMO processing unit, the SMO controller and the main memory, as shown in Figure 4.

A. The SMO Processing Unit

The SMO processing unit is responsible for calculating the new values of the two previously selected Lagrange multipliers. Figure 5 shows the building blocks of the SMO processing unit.

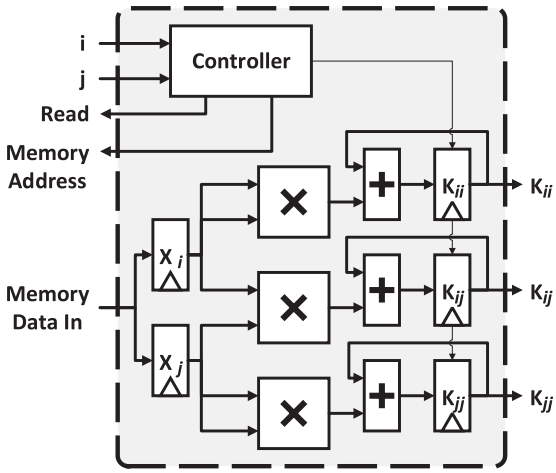


Fig. 6. Linear Kernel Function.

TABLE II
LEARNED FUNCTION PSEUDO CODE

```
for j=1:N
    for d=1:dimensions
        k = k + x_{j,d}.x_{i,d}
    sum = sum + α_i.y_i.k
func=sum
```

Register file: In order to speed up processing and avoid repeated memory access, some variables are cached in a register file to be processed later by the other SMO processing units. The variables chosen to be cached in the register file are α_{i}, α_j, y_i, y_j, B, α_i^{new}, α_j^{new}, E_i, $and\, E_j$.

Kernel function: The calculation of η requires the calculation of the two Lagrange multipliers' self and cross kernel. Hence, the kernel function unit calculates the value of $k_{ii}, k_{jj}, and\, k_{ij}$ simultaneously. After receiving the index of current Lagrange multipliers, the kernel function unit reads from the memory the value of the two Lagrange multipliers and passes them to three multiply-add units, as shown in Figure 6. In the case of a polynomial kernel instead of a linear one, the design also has an adder to add 1 to each K and then uses a multiplier to raise the value to the polynomial degree in multiple clocks. The kernel function unit includes an internal controller to manage the iterative process of reading the Lagrange multiplier and updating the kernel function value.

Learned function: The learned function is used to calculate $w^T x$ or $\sum_{i=1}^{N} \alpha_i.y_i.K(x_{i,}, x_j)$, which is used in calculating the error E. By expanding the equation $\sum_{i=1}^{N} \alpha_i.y_i.K(x_{i,}, x_j)$, the pseudo code in Table II is obtained.

The implementation requires two multiply-add units: one to calculate the kernel and the other to update the learned function. However, since the two calculations are dependent, one multiply-add unit is shared to calculate both values.

The FSM of the learned function is shown clearly in Figure 7. In the first state, α_i is read. If $\alpha_i \neq 0$, the FSM is moved to the kernel calculation state. Then, y_j is read to update the learned function value.

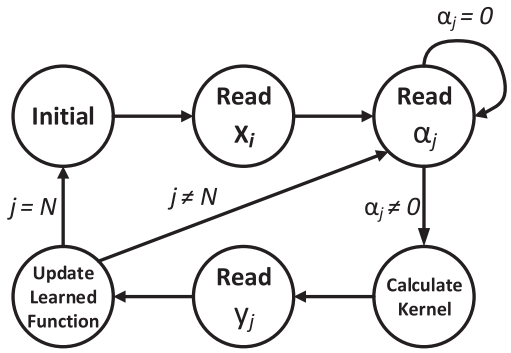


Fig. 7. Learned Function FSM.

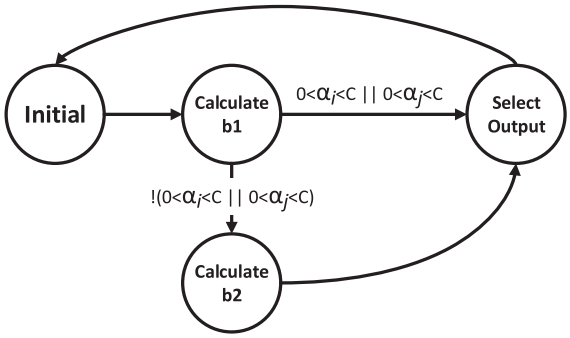


Fig. 8. Bias Calculator FSM.

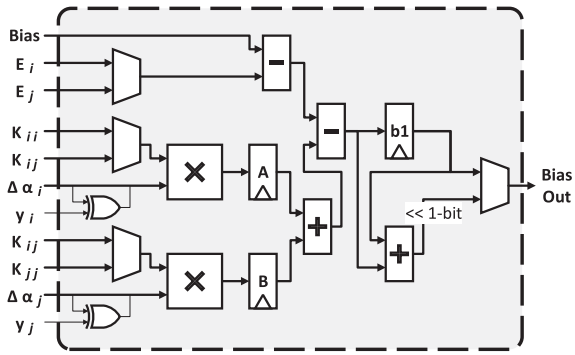


Fig. 9. Bias Calculator.

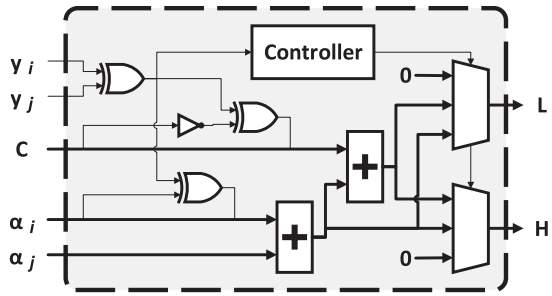


Fig. 10. Limits Calculator.

Bias calculator: The change in the threshold is computed by forcing E_i^{new} to be zero if $0 < \alpha_i^{\text{new}} < C$ and then

$$b_1 = E_i + y_i.\triangle\alpha_i.k_{ii} + y_j.\triangle\alpha_j.k_{ij} + b \tag{16}$$

where $\triangle\alpha_i = (\alpha_i^{\text{new}} - \alpha_i)$, and $\triangle\alpha_j = (\alpha_j^{\text{new}} - \alpha_j)$.

Otherwise, the threshold is computed by forcing E_j^{new} to be zero if $0 < \alpha_j^{\text{new}} < C$. Therefore,

$$b_2 = E_j + y_i.\triangle\alpha_i.k_{ij} + y_j.\triangle\alpha_j.k_{jj} + b. \tag{17}$$

Finally, the new bias is calculated as follows:

$$b = \begin{cases} b_1, & 0 < \alpha_i^{\text{new}} < C \\ b_2, & 0 < \alpha_j^{\text{new}} < C \\ (b_1 + b_2)/2, & \text{otherwise} \end{cases} \tag{18}$$

Figure 8 shows clearly the FSM of the bias calculator which consists of different states: calculating b1, calculating b2, and then choosing between them or their average.

Figure 9 illustrates the implementation of the bias calculator unit. The unit is implemented using only two multipliers, four adders, and three intermediate registers A, B, and b1. To exploit the similarities between (16) and (17), they are rewritten as:

$$b_1 = E_1 + T_1 + T_2 + b \tag{19}$$

$$b_2 = E_2 + T_3 + T_4 + b \tag{20}$$

where $T_1 = y_i.\triangle\alpha_i.k_{ii}$, $T_2 = y_j.\triangle\alpha_j.k_{ij}$, $T_3 = y_i.\triangle\alpha_i.k_{ij}$, and $T_4 = y_j.\triangle\alpha_j.k_{jj}$.

Noticing the similarity between T1 and T3, only one multiplier is used to calculate $\triangle\alpha_i.k_{ii}$ and $\triangle\alpha_i.k_{ij}$, and therefore the values of T1 and T3. Based on the condition $0 < \alpha_i^{\text{new}} < C$ and the condition $0 < \alpha_j^{\text{new}} < C$, either k_{ii} or k_{ij} is selected to be an input to the multiplier. If both conditions are satisfied, both b_1 and b_2 gives the same value. In the proposed hardware implementation, priority is given to b_1 to reduce hardware complexity. Therefore, the value of register A is calculated. The fact that y has a unity value, with a positive or negative sign, and the adoption of sign and magnitude representation results in reducing the multiplication of y to a single XOR gate between the y sign and the multiplicand sign. Similarly, T_2 and T_4 calculations require only one multiplier and then the value of the B register is obtained in parallel with the calculation of register A. If both conditions are not satisfied, the calculation is carried out to determine the value of b_1, and then the process is repeated to determine the value of b_2, and finally the values of b_1 and b_2 are averaged.

Limits calculator: The values of the lower band L and the upper band H depend on the slope in Figure 10. Therefore, the value of the limits is obtained as follows:

$$\text{if } y_i \neq y_j \rightarrow L = \max(0, \alpha_j - \alpha_i), \quad H = \min(C, C + \alpha_j - \alpha_i) \tag{21}$$

$$\text{if } y_i = y_j \rightarrow L = \max(0, \alpha_j + \alpha_i - C), \quad H = \min(C, \alpha_j + \alpha_i) \tag{22}$$

Again, comparing y_i and y_j is done using a single XOR gate. From (21) and (22), L and H take on the values 0, C, $\alpha_j \pm \alpha_i$, or $\alpha_j \pm \alpha_i \pm C$. Therefore, only two adders are required to calculate L and H, while the signs are determined using XOR gates. To further understand the implementation, the limits calculation process is described using the pseudo code in Table III.

TABLE III
LIMITS CALCULATOR PSEUDO CODE

```
if y_i ≠ y_j then
    if α_j − α_i is positive then
        L = α_j − α_i
        H = C
    else then
        L = 0
        H = C + α_j − α_i
    end if
else then
    if α_j + α_i − C is positive then
        L = α_j + α_i − C
        H = C
    else then
        L = 0
        H = α_j + α_i
    end if
end if
```

In the first part, the first adder is adjusted to add $\alpha_j - \alpha_i$ and the second adder is adjusted to add C to the output of the first adder, (i.e., $C + \alpha_j - \alpha_i$). Then a multiplexer is used to select between the values 0 and $\alpha_j - \alpha_i$ for L, and the values C and $C + \alpha_j - \alpha_i$ for H.

In the second part, the first adder is adjusted to add $\alpha_j + \alpha_i$ and the second adder is adjusted to add $-C$ to the output of the first adder, (i.e., $\alpha_j + \alpha_i - C$). Then a multiplexer is used to select between the values 0 and $\alpha_j + \alpha_i - C$ for L, and the values C and $\alpha_j + \alpha_i$ for H. The sign adjustment of α_i and C is controlled by first examining if $y_i \neq y_j$. This examination is performed using an XOR gate. Accordingly, the signs of α_i and C are altered by two more XOR gates. Notice that the cases when α_i is required to be negative are the same cases when C is required to be positive. That is why a NOT gate is added to the sign of C.

Memory interface: The memory interface is responsible for receiving requests for the memory read and write operations and handling the memory access separately by different blocks, which increases the memory access parallelism.

Controller: This unit controls the other units by initiating a triggering signal for each unit and manages the data flow between them. Figure 11 summarizes the controlling state machine of the control unit.

B. The SMO Controller

The SMO controller is responsible for selecting the two Lagrange multipliers and controls the SMO processing unit. The SMO controller keeps iterating over Lagrange multipliers till conversion happens or the maximum number of iterations is exceeded. Compared to the SMO processing unit, the SMO controller hardware occupies a smaller area and consumes less power.

SMO Results: The SMO implemented in this work utilizes an on-chip memory as shown in Figure 4. The use of on-chip memory results in a significant reduction in power consumption and performance enhancement than the off-chip memory [17]. For example, this work, achieves a high throughput of 0.46 Gbit/sec which is almost 100X higher compared to [17] which reports a throughput of 5.07 Mbits/sec.

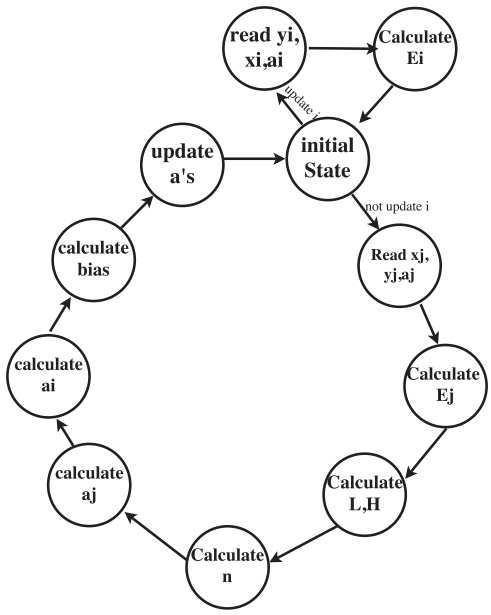


Fig. 11. SMO Processing Controller.

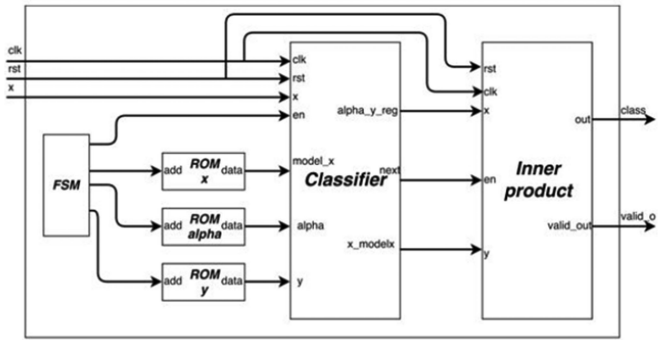


Fig. 12. SVM Classifier.

V. CLASSIFIER

In this section, the implementation of the low power classifier is presented, and several techniques are used to reduce the power consumption of the SVM classifier.

A. Algorithm

After the completion of the training phase, the classification phase starts. For any input vector x_{test} by substituting in the following formula using the final value of $\alpha's$ and b, the corresponding class y_{test} is calculated as follows:

$$y_{test} = \sum \alpha_j y_j x_{test} x_j + b$$

The training of SVM is done offline or using the hardware accelerator proposed in Section IV which is the SMO. Figure 12 shows the architecture of the top-level design of the SVM classifier, which consists of 6 main blocks: three ROM blocks, classifier block, and inner product block. The first ROM block

is used to save the input vectors of the support vector points. The width of this ROM is the same as the data width, while the depth equals to the number of support vectors multiplied by the number of the classification problem dimensions. The second ROM block is used to save the values of non-zero 's. The width of this ROM is the same as the data width, while the depth equals the number of support vectors. The third ROM block is used to save the values of the true labels of the support vector points. The width of this ROM is one bit, while the depth is the number of support vectors. The finite state machine (FSM) is responsible for generating the addresses of the three ROMs and the enable signal of the classifier block.

The classifier block is the main block of the architecture. First, each α is multiplied by its corresponding label y. As the implementation used for negative numbers is sign-magnitude implementation, the multiplication is performed using an XOR gate instead of a multiplier. The values of α_i and y_i are stored in a register. An inner product block of size equal to the number of dimensions is used to multiply the input test vector by the input vector of the i_{th} support vector point. The output of the classifier block is fed to the inner product block to calculate the class. The inner product block is a multiplier-add block with only one adder, and one multiplier that multiplies two vectors of size equal to the number of non-zero $\alpha' s$, The output of this block is the class and a valid out signal.

In the hardware implementation of the SVM classifier, fixed-point representation is used. Using software simulation results, it is found that a 16-bit word length is enough for achieving the same performance (i.e., accuracy). Same as that used in the training accelerators, computation skipping is adopted to save more power/ area.

VI. Optimization and Hardware Implementation of Selected Features

A. Simulation Setup

The Seizure detection system implemented in this paper is tested and simulated on neural seizure detection as a case study. At first patients' EEG signals are processed. Then, the SVM algorithm is applied to the features' vectors to find the optimal decision between seizure and normal activities. The CHB-MIT scalp data-set from the Physio-Net library is adopted to test the implemented system. The data-set was collected at the Children' s Hospital Boston from subjects with intractable seizures. Recordings were collected from 22 subjects (5 males and 17 females). The age of the subjects ranged from 3 to 22 years old in males and from 1.5 to 19 in females. The signals were sampled at 256 samples per second with a 16-bit resolution. For each subject, 23 channels were recorded from different electrodes. The data-set came with labeling on the epileptic sessions for different patients [38], [39]. The software for extracting features from the EEG, testing algorithm efficiency and collecting performance results are implemented using MATLAB 2017a.

The performance is evaluated through three different metrics commonly used in seizure detection, namely sensitivity, specificity, and accuracy. Sensitivity is the algorithm's ability to detect seizures correctly, whereas specificity is the algorithm's ability to avoid false alarms. There is always a trade-off between sensitivity and specificity. Accuracy is a combining matrix between the two of them.

TABLE IV
Length of Input Vector Optimization

Total number of bits	I	F	Sensitivity	Specificity	Accuracy
16	10	6	98.38%	92.14%	92.16%
10	10	0	98.38%	91.91%	91.92%
12	8	4	98.38%	91.73%	91.75%
8	8	0	98.38%	91.62%	91.64%

$$\text{Sensitivity} = \frac{TP}{TP + FN}$$

$$\text{Specificity} = \frac{TN}{TN + FP}$$

$$\text{Accuracy} = \frac{TP + TN}{TP + TP + FP + FN}$$

Where TP denotes true positives, FN denotes false negatives, TN denotes true negatives and FP denotes false positives. These values are averaged by the number of iterations in the testing data to measure the performance of the implemented system.

B. Optimization Steps

In this section, design specifications and all possible optimizations before moving to actual hardware implementation are discussed. Moving to the Hardware implementation requires deciding certain design specifications such as the length of the input vector that represents each sample of the EEG signal, the size of the intermediate signals from one block to the other, and finally the length of output vector of the whole module.

1) Length of Input Vector Optimization: Originally, When fractal dimension, Hurst exponent, and coastline are used to extract the features, the sensitivity is 98.38%, the specificity is 92.14%, and the accuracy is 92.16%. To decide the optimum length of the input vector that represents each EEG signal sample, several values are tested. Table IV shows the sensitivity, specificity, and accuracy acquired in response to representing each EEG sample with I bits for the integer part, and F bits for the fraction part.

It is clear that the fractional part makes no significant contribution to the sensitivity, specificity, and accuracy, thus it is neglected. When the total number of bits is the same as the number of bits in the integer part equals 8, the sensitivity remains the same, and the specificity slightly decreases by 0.5%. In both Hurst exponent and fractal dimension, the input is directly fed to accumulate modules, where every new data input is added to the sum of all the previous data inputs. For this reason, choosing the least possible number of bits for each EEG input sample is mandatory. Consequently, 8 bits are chosen for representing each EEG sample.

2) Linear and Nonlinear Scaling: Division by a constant can be considered as linearly scaling all values by the same amount. In this case, the existence of a division does not affect the classification, because whether there is a division or not, this

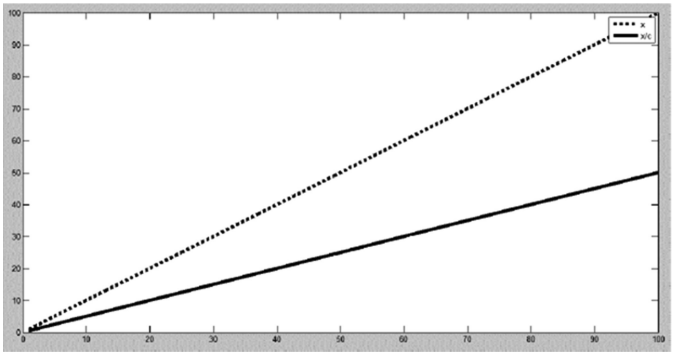


Fig. 13. Linear Scaling.

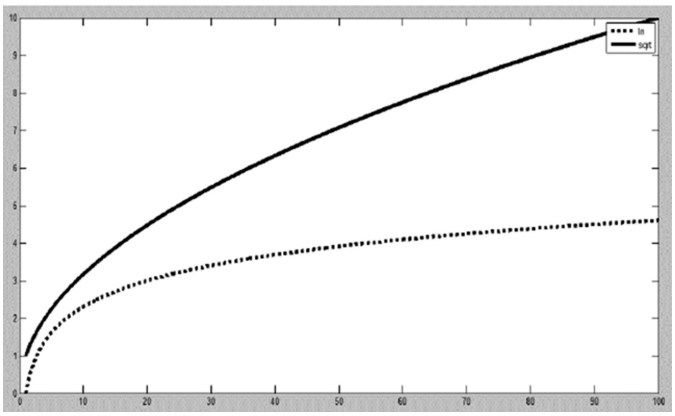


Fig. 14. Non-linear Scaling.

TABLE V
FINAL OPTIMIZATION RESULTS

Feature	Optimization	sensitivity	specificity	Accuracy
Hurst Exponent	Removing the division by the standard deviation	96.77%	90.34%	90.36%

merely determines whether the hyper-plane drawn by the linear SVM classifier is shifted by the scaling value or not. Figure 13 shows the linear scaling of $\frac{x}{c}$ value when the division by the constant c is removed.

The same principle applies to taking the natural logarithm of certain values, it can be considered as non-linearly scaling values with a specific manner which is in this case Ln. If another mathematical functions can behave in an approximately similar manner to the Ln, the Ln can be, in this case, replaced by the other mathematical function. Reviewing all curves of the mathematical functions compared to the Ln curve, the square root curve is the best approximation to the Ln. Figure 14 shows the analogy between Ln and square root.

To verify the proposed concept, the division by a constant is removed from the final stages of the selected features. The division in mean absolute value is not, however, removed because the existence of a division can be in some cases advantageous to the hardware design when it is located after accumulate modules. In those cases, division minimizes the input that is to be fed

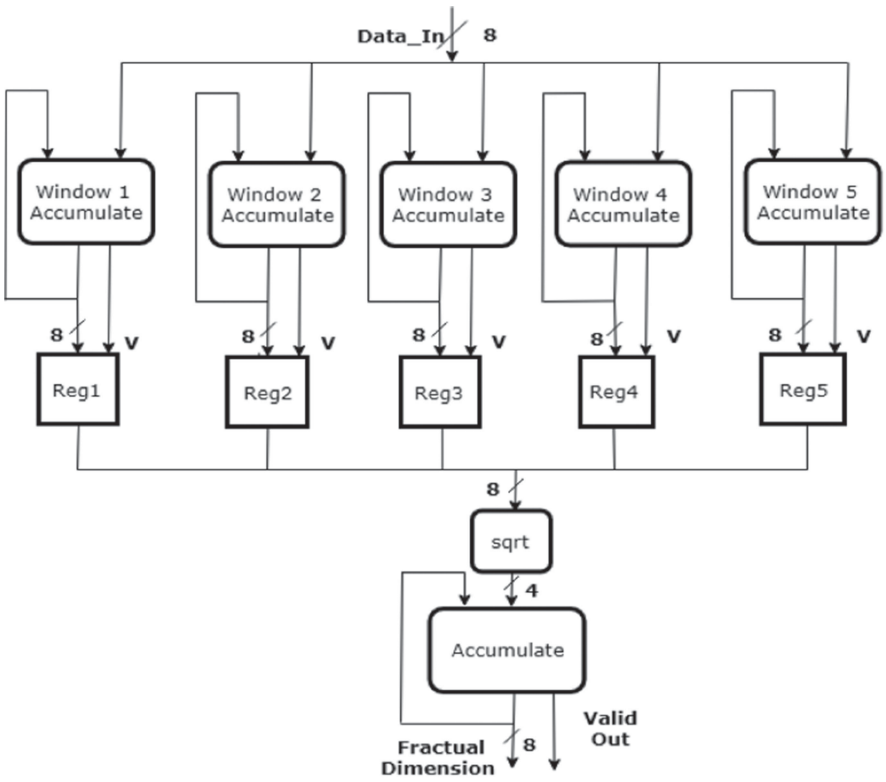
TABLE VI
PROPOSED OPTIMIZED EQUATIONS FOR CALCULATING FEATURES

Feature	Proposed Equations
Fractal Dimension	$L_m(k) = \sum_{i=1}^{\frac{D-m}{k}} \lvert x(m+ik) - x(m+(i-1)*k)\rvert$ $FD = \sum_{m=1}^{k} \sqrt{L_m(k)}$ (23)
Hurst Exponent	$MAV = \sum_{i=1}^{D} \lvert X_i\rvert$ $R = \lvert\lvert MAX(X-MAV)\rvert - \lvert MIN(X-MAV)\rvert\rvert$ $H = \sqrt{R}$ (24)
Coastline	$CL = \sum_{i=1}^{D} \lvert x_{i+1} - x_i\rvert$ (25)

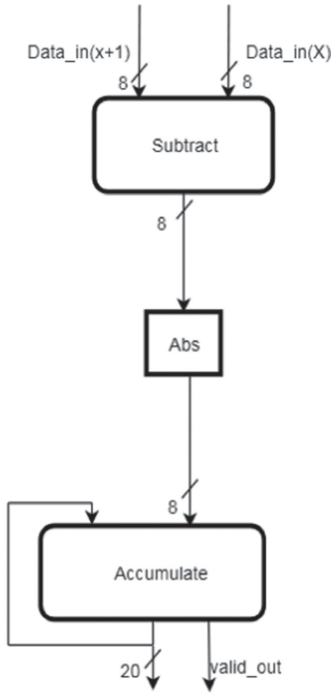
to the following stages resulting in smaller and faster design. To avoid the complexities of designing a divider, a custom division through shifting is exploited, then every Ln is replaced by a square root. When the division is removed, the sensitivity remains as originally obtained 98.38%, and when the every Ln is replaced by a square root function, the sensitivity drops to 96.77% (a drop by 1.61%), and the specificity is dropped to 91.62 (a drop by 0.52%).

In a similar manner to obtaining the optimum length of the input EEG samples, the optimum length for each of the intermediate signals is obtained. As discussed, the input EEG sample is represented by only 8 bits in the integer part, thus the output of each accumulate module is an integer. However, limiting accumulate modules output to the least possible number of bits ensures faster and smaller design. First of all, the output of the five accumulate modules in the fractal dimension is limited to 8 bit, second the output of the mean absolute value utilized in the Hurst exponent feature is limited to 10 bits, then the output of the accumulator within the standard deviation module utilized in the Hurst exponent feature is bounded to 20 bits, followed by bounding the output of the whole Hurst exponent feature to 30 integer bits. Finally, the output of the coastline feature is bounded to 20 integer bits. The previous optimizations does not affect any of the sensitivity, specificity, or accuracy.

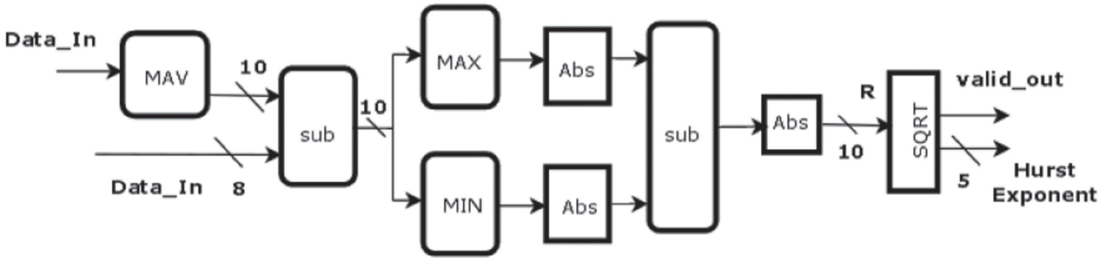
A final approximation is done in the Hurst exponent feature by removing the division by the standard deviation. Removing this division affects the obtained sensitivity, specificity, and accuracy since it is not constant, but rather dependent on the values of the EEG samples of each window. The motive behind removing the division by the standard deviation, even if it causes slight performance degradation, is that it is fraction division (dividing 8 bits by 20 bits) which would make the hardware design more complex. Table V shows that while removing the Ln function has dominant effect on the sensitivity, removing the standard deviation has the dominant effect on the specificity. Consequently, the specificity drops to 90.34% after removing the division by the standard deviation, while the sensitivity achieved remains 96.77%.



(a) Optimized Fractal Dimension



(b) Optimized Coastline



(c) Optimized Hurst Exponent

Fig. 15. Block Diagram of the Optimized Features.

C. Hardware Implementation of Optimized Features

The proposed modified feature extraction strategies are shown in Table VI, while their detailed architecture showing the actual length of I/O vector lengths, along with the vector lengths of the intermediate signals, is depicted in Figure 15.

VII. Results and Comparison

In this section, the results obtained are shown and compared to prior work. Xilinx ISE 14.2 and Vivado 2016.4 are utilized to design and develop the VLSI architecture of the algorithms. The design is synthesized on Xilinx Virtex-7 FPGA. For the implementation on ASIC, Synopsis Design Compiler (DC) B-2008.09 with hardware-calibrated UMC 65 nm CMOS technology is adopted. The final layout is conducted using Cadence SoC-Encounter.

Results are reported in two main phases. The first phase objective is to evaluate the performance, simulation results using MATLAB 2017a as shown in Tables IV and V. The second phase objective is to calculate the hardware implementation performance metrics such as area utilization and power consumption for both ASIC and FPGA implementations.

A. FPGA Implementation Results

The results shown in Figure 16 are obtained from the integrated logic analyzer (ILA) after the implementation step on the FPGA. The Figure shows two epochs of EEG signals that are supplied to the proposed system; one epoch contains an inter-ictal signal seizure, and the other contains ictal signal. The whole epoch is processed by the feature extraction module supplying the feature vector to the classifier module. The classifier

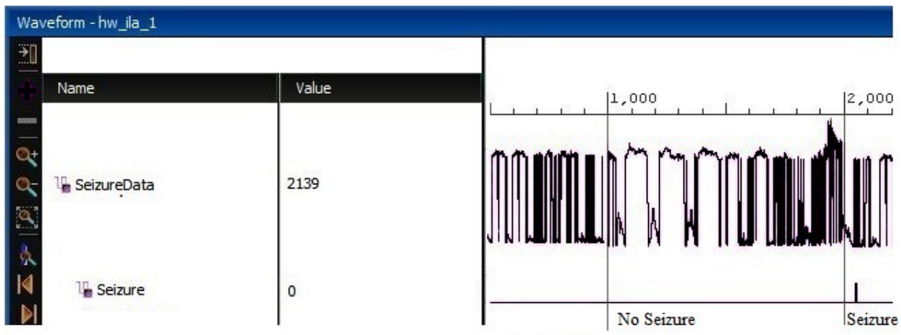


Fig. 16. The Waveform of the Implemented Design on FPGA.

TABLE VII
FPGA IMPLEMENTATION RESULTS

Block	Area		Power (mW)
	LUT	Registers	
SMO	3078	566	17
Feature Extraction	1326	700	7
Classifier	238	137	6

TABLE VIII
ASIC UMC 65 nm IMPLEMENTATION RESULTS

Block	Number Of Cells	Number Of Gates	Area (μm^2)	Power (mW)
SMO	33722	107920	112225	9.65
Feature Extractor	11171	43356	57600	2.55
Classifier	8554	26370	32400	2.71

output named seizure is also shown in both cases giving (logic-1) when an epoch has a seizure and (logic-0) when the epoch is seizure-free. The classifier output is generated after 16 clock cycles after receiving the input feature vector.

Table VII lists the resources utilized in Xilinx Virtex-7 FPGA, such as LUTs and register slices as well as the dynamic power consumption. The proposed SMO was chosen to save power consumption and area utilization as much as possible, the accelerator utilizes 3078 LUTs compared to 6040 utilized LUTs in the SMO implementation proposed in [23], and 3644 logic elements compared to 6836 logic elements in MSMO [24].

B. ASIC Implementation Results

The hardware implementations of the SVM learning and testing circuits are presented on both FPGA and ASIC platforms. Table VIII shows the ASIC implementation results using UMC 65 nm CMOS technology, with a clock frequency of 100 MHz.The table also shows the total number of the system's gates and cells, occupied area, and power consumption, while Figure 17 shows the full layout of the system.

Power analysis is conducted for both FPGA and ASIC implementations. Its is found that the FPGA implementation consumes 2X the power that the ASIC implementation consumes; the FPGA implementation of the proposed system consumes 30 mW, while the ASIC implementation consumes only 14.91 mW.

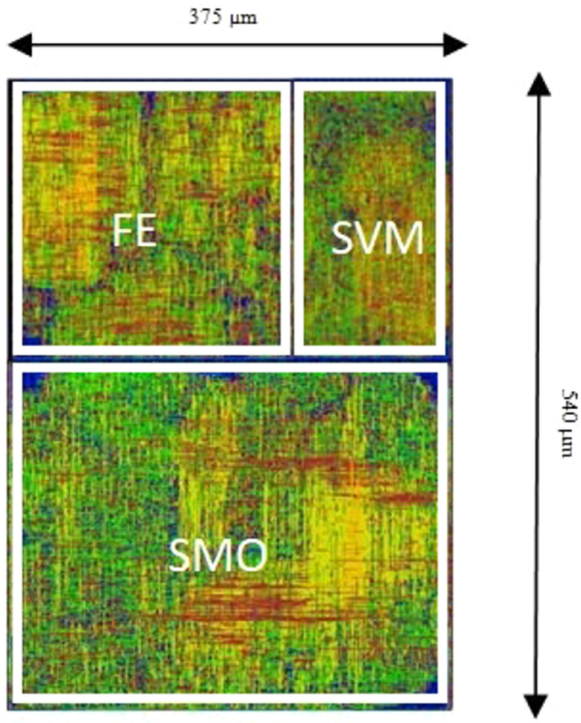


Fig. 17. The Layout of the Proposed System.

TABLE IX
PERFORMANCE COMPARISON TO PRIOR WORK

Method	Classifier type	Sensitivity	Specificity
[4]	RBF SVM	94.5%	95.3%
[40]	RBF SVM	95.0%	93.7%
[41]	RBF SVM	97.0%	96.0%
[42]	Gradient boosted trees	83.7%	88.1%
[44]	Linear SVM	95.1%	96.2%
[17]	RBF SVM	96.8%	99.1%
Proposed	Linear SVM	96.7%	90.34%

C. Comparison to Prior Work

In the proposed design, optimizing the power consumption and area utilization while keeping a high sensitivity are the main concern. The proposed system achieved sensitivity up to 96.77% which exceeds the maximum sensitivity achieved

TABLE X
FPGA SYSTEM-LEVEL COMPARISON

	System Blocks	FPGA	Area			Performance	
			LUTs	Registers	DSPs	Percentage	Metrics
[45]	Training + Feature Extraction + classification	Artix 7	20993	8248	6	91.93%	Specificity
[46]	Feature Extraction	PYNQ-Z1	17492	16685	46	92.9%	Sensitivity
						96.30%	Specificity
						95.80%	Accuracy
[47]	Feature Extraction + FFT +Classification	Zed-board	8001	8662	31	98.40%	Sensitivity
						0.356	FDR/h
[48]	Classification	Artix 7	1209		-	92.00%	Sensitivity
						81.00%	Accuracy
Proposed Work	Feature Extractor + SMO+ LSVM	Virtex 7	4642	1403	-	96.77%	Sensitivity
						90.34%	Specificity
						90.36%	Accuracy

by the software-implemented systems proposed in [4], [40], [44], and almost equals the one achieved in [41], as shown in Table IX. This performance is achieved without exploiting a pre-processing stage, and using a linear kernel instead of the computationally expensive radial basis function (RBF) kernel used in prior work and also exceeds that achieved by other classifiers as the Gradient boosted trees [42]. Although Table IX shows a relatively low specificity compared to previous work which can cause excess stimulation than needed resulting in inducing seizures in the normal brain [43] or drainage of the battery. In the proposed design, the focus is on the power consumption and the area utilization at the expense of some acceptable degradation in the other performance metrics such as sensitivity and specificity. However, the table shows that the proposed design exploits linear kernel (i.e., less area and power) while other designs exploit RBF kernel (i..e, higher power and larger area). In addition, the usage of linear kernel and the approximations introduced to the feature extractor module, such as removing the natural logarithm and divider blocks, significantly reduce the power consumed by the proposed system at the expense of only 0.2 false alarms in every 10 seizures (drop by less than 2% in the obtained specificity).

[17] implemented the whole system in hardware, achieving higher sensitivity by 0.03%, but at the expense of higher power consumption than the proposed system by a factor of 1.5X.

System-level comparison with previous work in Table X shows that the proposed system has significantly less area utilization. Also, it has achieved the highest sensitivity.

VIII. CONCLUSION

Many algorithms are used to train the SVM. In this work, accelerators such as SMO training algorithm are hardware-implemented on both FPGA and ASIC platforms. The implemented accelerator has been tested with a hardware-implemented feature extractor and classifier in a neural seizure detection application.

Multiple optimization techniques are introduced to the system to achieve the most optimum design in terms of resource utilization, and power consumption; for instance, replacing the computationally expensive functions such as division and natural logarithm by other analogous less computationally expensive functions. The proposed system achieved a sensitivity up to 96.77% using a linear kernel function, which exceeds the sensitivity obtained in prior work using an RBF kernel by a 1.4%. In addition, the system consumes 1.5X less power than prior work's implementation.

REFERENCES

[1] K. Devarajan, S. Bagyaraj, V. Balasampath, E. Jyostna, and K. Jayasri, "EEG-based epilepsy detection and prediction," *Int. J. Eng. Technol.*, vol. 6, no. 3, pp. 212–216, 2014.

[2] F. Mormann, R. G. Andrzejak, C. E. Elger, and K. Lehnertz, "Seizure prediction: The long and winding road," *Brain*, vol. 130, no. 2, pp. 314–333, 2006.

[3] A. Varsavsky, I. Mareels, and M. Cook, *Epileptic Seizures and the EEG: Measurement, Models, Detection and Prediction*. Boca Raton, FL, USA: CRC Press, 2016.

[4] Y. Liu, W. Zhou, Q. Yuan, and S. Chen, "Automatic seizure detection using wavelet transform and SVM in long-term intracranial EEG," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 20, no. 6, pp. 749–755, Nov. 2012.

[5] T. N. Alotaiby, F. E. A. El-Samie, S. A. Alshebeili, K. H. Aljibreen, and E. Alkhanen, "Seizure detection with common spatial pattern and support vector machines," in *Proc. IEEE Int. Conf. Inf. Commun. Technol.*, 2015, pp. 152–155.

[6] P. Bhuvaneswari and J. S. Kumar, "Support vector machine technique for EEG signals," *Int. J. Comput. Appl.*, vol. 63, no. 13, pp. 1–5, 2013.

[7] A. Temko, E. Thomas, W. Marnane, G. Lightbody, and G. Boylan, "EEG-based neonatal seizure detection with support vector machines," *Clin. Neurophysiol.*, vol. 122, no. 3, 464–473, 2011.

[8] A. Temko, E. Thomas, G. Boylan, W. Marnane, and G. Lightbody, "An SVM-based system and its performance for detection of seizures in neonates," in *Proc. Annu. Int. Conf. IEEE Eng. Med. Biol. Soc.*, 2009, pp. 2643-2646.

[9] T. Maeda, "How to rationally compare the performances of different machine learning models?" *PeerJ Preprints*, vol. 6, 2018, Art. no. e26714v1.

[10] M. A. Bin Altaf and J. Yoo, "A 1.83 μ J/classification, 8-channel, patient-specific epileptic seizure classification SoC using a non-linear support vector machine," *IEEE Trans. Biomed. Circuits Syst.*, vol. 10, no. 1, pp. 49–60, Feb. 2016.

[11] J. N. Aziz *et al.*, "256-channel neural recording and delta compression microsystem with 3D electrodes," *IEEE J. Solid-State Circuits*, vol. 44, no. 3, pp. 995–1005, Mar. 2009.

[12] B. Boashash, M. Mesbah, and P. Coldtiz, "Time-frequency detection of EEG abnormalities," in *Time-Frequency Signal Analysis and Processing: A Comprehensive Reference*. Amsterdam, The Netherlands: Elsevier, 2003, ch. 15, pp. 663–670.
[13] A. Subasi and M. I. Gursoy, "EEG signal classification using PCA, ICA, LDA and support vector machines," *Expert Syst. Appl.*, vol. 37, no. 12, pp. 8659–8666, 2010.
[14] K. Helal *et al.*, "Low-power high-accuracy seizure detection algorithms for neural implantable platforms," in *Proc. IEEE Int. Conf. Microelectron.*, 2017, pp. 231–234.
[15] M. S. Mercy, "Performance analysis of epileptic seizure detection using DWT & ICA with neural networks," *Int. J. Comput. Eng. Res.*, vol. 2, no. 4, pp. 1109–1113, 2012.
[16] J. Platt, "Sequential minimal optimization: A fast algorithm for training support vector machines," Microsoft Res., Redmond, WA, USA, Tech. Rep. MSR-TR-98-14, Apr. 1998.
[17] L. Feng, Z. Li, and Y. Wang, "VLSI design of SVM-based seizure detection system with on-chip learning capability," *IEEE Trans. Biomed. Circuits Syst.*, vol. 12, no. 1, pp. 171–181, Feb. 2018.
[18] S. M. Afifi, H. GholamHosseini, and S. Poopak, "Hardware implementations of SVM on FPGA: A state-of-the-art review of current practice," *Int. J. Innovative Sci. Eng. Technol.*, vol. 2, pp. 733–752, 2015.
[19] L. J. Cao *et al.*, "Parallel sequential minimal optimization for the training of support vector machines," *IEEE Trans. Neural Netw.*, vol. 17, no. 4, pp. 1039–1049, Jul. 2006.
[20] K.-K. Cao, H.-B. Shen, and H.-F. Chen, "A parallel and scalable digital architecture for training support vector machines," *J. Zhejiang Univ. Sci. C*, vol. 11, no. 8, pp. 620–628, 2010.
[21] C.-H. Peng, B.-W. Chen, T.-W. Kuan, P.-C. Lin, J.-F. Wang, and N.-S. Shih, "REC-STA: Reconfigurable and efficient chip design with SMO-based training accelerator," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 22, no. 8, pp. 1791–1802, Aug. 2014.
[22] L. Bustio-Martínez, R. Cumplido, J. Hernández-Palancar, and C. Feregrino-Uribe, "On the design of a hardware-software architecture for acceleration of SVM' s training phase," in *Proc. Mex. Conf. Pattern Recognit.*, 2010, pp. 281–290.
[23] J.-F. Wang, J.-S. Peng, J.-C. Wang, P.-C. Lin, and T.-W. Kuan, "Hardware/software co-design for fast-trainable speaker identification system based on SMO," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, 2011, pp. 1621–1625.
[24] L. Feng, Z. Li, Y. Wang, C. Zheng, and Y. Guan, "VLSI design of modified sequential minimal optimization algorithm for fast SVM training," in *Proc. 13th IEEE Int. Conf. Solid-State Integr. Circuit Technol.*, Hangzhou, China, 2016, pp. 627–629.
[25] S. Venkateshan, A. Patel, and K. Varghese, "Hybrid working set algorithm for SVM learning with a kernel coprocessor on FPGA," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 23, no. 10, pp. 2221–2232, Oct. 2015.
[26] M. B. Rabieah and C.-S. Bouganis, "FPGA based nonlinear support vector machine training using an ensemble learning," in *Proc. 25th Int. Conf. Field Programmable Logic Appl.*, 2015, pp. 1–4.
[27] S. Wang, Y. Peng, G. Zhao, and X. Peng, "Accelerating on-line training of LS-SVM with run-time reconfiguration," in *Proc. Int. Conf. Field-Programmable Technol.*, 2011, pp. 1–6.
[28] A. Bhattacharyya and R. B. Pachori, "A multivariate approach for patient-specific EEG seizure detection using empirical wavelet transform," *IEEE Trans. Biomed. Eng.*, vol. 64, no. 9, pp. 2003–2015, Sep. 2017.
[29] M. Sharma, R. B. Pachori, and U. R. Acharya, "A new approach to characterize epileptic seizures using analytic time-frequency flexible wavelet transform and fractal dimension," *Pattern Recognit. Lett.*, vol. 94, pp. 172–179, 2017.
[30] R. R. Sharma and R. B. Pachori, "Time–frequency representation using IEVDHM–HT with application to classification of epileptic EEG signals," *IET Sci., Meas. Technol.*, vol. 12, no. 1, pp. 72–82, 2018.
[31] A. Gammerman and V. Vovk, "Alexey Chervonenkis's bibliography: Introductory comments," *J. Mach. Learn. Res.*, vol. 16, pp. 2051–2066, 2015.
[32] Q. Wang, P. Li, and Y. Kim, "A parallel digital VLSI architecture for integrated support vector machine training and classification," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 23, no. 8, pp. 1471–1484, Aug. 2015.
[33] N. Petra, D. De Caro, V. Garofalo, E. Napoli, and A. G. Strollo, "Truncated binary multipliers with variable correction and minimum mean square error," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 57, no. 6, pp. 1312–1325, Jun. 2010.
[34] T. Higuchi, "Approach to an irregular time series on the basis of the fractal theory," *Phys. D, Nonlinear Phenomena*, vol. 31, no. 2, pp. 277–283, 1988.
[35] V. Vijith, J. E. Jacob, T. Iype, K. Gopakumar, and D. G. Yohannan, "Epileptic seizure detection using non linear analysis of EEG," in *Proc. Int. Conf. Inventive Comput. Technol.*, 2016, vol. 3, pp. 1–6.
[36] S. Raghunathan, A. Jaitli, and P. P. Irazoqui, "Multistage seizure detection techniques optimized for low-power hardware platforms," *Epilepsy Behav.*, vol. 22, pp. S61–S68, 2011.
[37] M. A. Elgammal *et al.*, "Linear and nonlinear feature extraction for neural seizure detection," in *Proc. IEEE 61st Int. Midwest Symp. Circuits Syst.*, 2018, pp. 795–798.
[38] A. L. Goldberger *et al.*, "PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals," *Circulation*, vol. 101, no. 23, pp. 215–220, 2000.
[39] A. H. Shoeb, "Application of machine learning to epileptic seizure onset detection and treatment," Ph.D. dissertation, MIT Div. Health Sci. Technol., Massachusetts Inst. Technol., Cambridge, MA, USA, 2009.
[40] Q. Yuan, W. Zhou, S. Li, and D. Cai, "Epileptic EEG classification based on extreme learning machine and nonlinear features," *Epilepsy Res.*, vol. 96, no. 1/2, pp. 29–38, 2011.
[41] S. Li, W. Zhou, Q. Yuan, S. Geng, and D. Cai, "Feature extraction and recognition of ictal EEG using EMD and SVM," *Comput. Biol. Med.*, vol. 43, no. 7, pp. 807–816, 2013.
[42] M. Shoaran, B. A. Haghi, M. Taghavi, M. Farivar, and A. Emami-Neyestanak, "Energy-efficient classification for resource-constrained biomedical applications," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 8, no. 4, pp. 693–707, Dec. 2018.
[43] J. Nissinen, T. Halonen, E. Koivisto, and A. Pitkänen, "A new model of chronic temporal lobe epilepsy induced by electrical stimulation of the amygdala in rat," *Epilepsy Res.*, vol. 38, no. 2/3, pp. 177–205, 2000.
[44] C. Zhang, M. A. Bin Altaf, and J. Yoo, "Design and implementation of an on-chip patient-specific closed-loop seizure onset and termination detection system," *IEEE J. Biomed. Health Inform.*, vol. 20, no. 4, pp. 996–1007, Jul. 2016.
[45] A. Mohammed and A. Demosthenous, "Complementary detection for hardware efficient on-site monitoring of Parkinsonian progress," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 8, no. 3, pp. 603–615, Sep. 2018.
[46] D. Serasinghe, D. Piyasena, and A. Pasqual, "A novel low-complexity VLSI architecture for an EEG feature extraction platform," in *Proc. 21st Euromicro Conf. Digit. Syst. Des.*, Prague, Czech Republic, 2018, pp. 472–478.
[47] H. Wang, W. Shi, and C. Choy, "Hardware design of real time epileptic seizure detection based on STFT and SVM," *IEEE Access*, vol. 6, pp. 67277–67290, 2018.
[48] L. Marni, M. Hosseini, J. Hopp, P. Mohseni, and T. Mohsenin, "A real-time wearable FPGA-based seizure detection processor using MCMC," in *Proc. IEEE Int. Symp. Circuits Syst.*, Florence, Italy, 2018, pp. 1–4.

Heba Elhosary received the B.Sc. degree in electronic engineering from the Faculty of Information Engineering and Technology, German University in Cairo, New Cairo, Egypt, in 2018. She did her bachelor project/thesis at Ruhr-University Bochum, Bochum, Germany, in 2017. She is currently a TA with the Department of Electronics, Faculty of Information Engineering and Technology, German University in Cairo. Her research interests include system on chips, computer arithmetic, computer architecture, FPGAs, VLSI, and hardware implementation of biomedical systems.

Michael H. Zakhari received the B.Sc. degree (with Hons.) in electronic and communication engineering from the Faculty of Engineering, Ain Shams University, Cairo, Egypt, in 2018. He is currently preparing for his M.Sc. degree in electronic and communication engineering from the Faculty of Engineering, Cairo University. His research interests include digital electronics, VLSI, FPGA, and ASIC platforms.

Mohamed A. Elgammal received the B.Sc. and M.A.Sc degrees (with Hons.) in electronics engineering from Cairo University, Cairo, Egypt, in 2016 and 2018, respectively. He is currently working toward the Ph.D. degree at the ECE Department, University of Toronto, Toronto, ON, Canada. He was a Teaching and Research Assistant with Cairo University. He was also a QA FPGA Prototyping Engineer with Mentor Graphics Egypt. His research interests include FPGA, digital electronics, machine learning, and hardware implementation.

Mohamed A. Abd El Ghany received the B.S. degree in electronics and communications engineering (with Hons.) and master's degree in electronics engineering from Cairo University, Cairo, Egypt, in 2000 and 2006, respectively, and the Ph.D. degree in the area of high-performance VLSI/IC design from German University in Cairo (GUC), New Cairo, Egypt, in 2010. From 2003 to 2006, he was with the National Space Agency of Ukraine's EGYPTSAT-1 project. From 2008 to 2010, he was an International Scholar with the Electrical Engineering Department, Ohio State University, Columbus, OH, USA. From 2012 to 2014, he was awarded the Alexander von Humboldt Foundation Postdoctoral Fellowship at Technische Universität (TU) Darmstadt, Darmstadt, Germany. He is currently an Associate Professor with GUC. He is a Project Manager for four international projects between TU Darmstadt, Ruhr-University Bochum, and GUC. He is the author of about 55 papers, two book chapters, and two books in the fields of high throughput and low-power VLSI/IC design and NoC/SoC. His research interests include low-power design, embedded system design, network on chip design and related circuit level issues in high-performance VLSI circuits, clock distribution network design, digital ASIC design, and SoC/NoC design and verification. He is a Reviewer and Program Committee Member of many IEEE international journals and conferences.

Khaled N. Salama (S'97–M'05–SM'10) received the B.S. degree from the Department of Electronics and Communications, Cairo University, Cairo, Egypt, in 1997, and the M.S. and Ph.D. degrees from the Department of Electrical Engineering, Stanford University, Stanford, CA, USA, in 2000 and 2005, respectively. He was an Assistant Professor with the Rensselaer Polytechnic Institute, Troy, NY, USA, from 2005 to 2009. In 2009, he joined King Abdullah University of Science and Technology (KAUST), Thuwal, Saudi Arabia, where he is currently a Professor, and was the Founding Program Chair until 2011. He is the Director of the sensors initiative, a consortium of nine universities (KAUST, MIT, UCLA, GATECH, MIT, UCLA, Brown University, Georgia Tech, TU Delft, Swansea University, the University of Regensburg, and the Australian Institute of Marine Science). He has authored 250 papers and 20 issued U.S. patents on low-power mixed-signal circuits for intelligent fully integrated sensors and neuromorphic circuits using memristor devices. His work on CMOS sensors for molecular detection has been funded by the National Institutes of Health and the Defense Advanced Research Projects Agency, awarded the Stanford-Berkeley Innovators Challenge Award in biological sciences, and was acquired by Lumina, Inc.

Hassan Mostafa (S'01–M'11–SM'15) received the B.Sc. and M.Sc. degrees (with Hons.) in electronics engineering from Cairo University, Cairo, Egypt, in 2001 and 2005, respectively, and the Ph.D. degree in electrical and computer engineering from the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada, in 2011. He is currently an Associate Professor with the Nanotechnology and Nanoelectronics Program at Zewail City of Science and Technology, Giza, Egypt—on leave from the Department of Electronics and Electrical Communications, Cairo University. He was an NSERC Postdoctoral Fellow with the Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON, Canada. His postdoctoral work includes the design of the next generation FPGA in collaboration with Fujitsu Research Labs in Japan/USA. He has authored/coauthored more than 170 papers in international journals and conferences and is the author/co-author of five published books. His research interests include neuromorphic computing, IoT hardware security, software-defined radio, reconfigurable low power systems, analog-to-digital converters, low-power circuits, subthreshold logic, variation-tolerant design, soft error tolerant design, statistical design methodologies, next generation FPGA, spintronics, memristors, energy harvesting, MEMS/NEMS, power management, and optoelectronics. He has been a member of the IEEE Technical Committee of VLSI Systems and Applications since 2017. He was the recipient of the University of Toronto Research Associate Scholarship in 2012, the Natural Sciences and Engineering Research Council of Canada Prestigious Postdoctoral Fellowship in 2011, the Waterloo Institute of Nano-Technology Nanofellowship Research Excellence Award in 2010, the Ontario Graduate Scholarship in 2009, and the University of Waterloo SandFord Fleming TA Excellence Award in 2008.