Chapter 12

On Decoding Brain Electrocorticography Data for Volitional Movement Intention Prediction: Theory and On-Chip Implementation

Mradul Agrawal, Sandeep Vidyashankar and Ke Huang

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/66149

Abstract

Brain-computer interface (BCI) has recently received an unprecedented level of consideration and appreciation in medical applications, such as augmentation and reparation of human cognitive or sensorimotor activities. Brain signals such as electroencephalogram (EEG) or electrocorticography (ECoG) can be used to generate stimuli or control device though decoding, translating, and actuating; this communication between the brain and computer is known as BCI. Moreover, signals from the sensors can be transmitted to a person's brain enabling them to see, hear, or feel from sensory inputs. This two-way communication is referred as bidirectional brain-computer interface (BBCI). In this work, we propose a field-programmable gate array (FPGA)-based on-chip implementation of two important data processing blocks in BCI systems, namely, feature extraction and decoding. Experimental results showed that our proposed architecture can achieve high prediction accuracy for decoding volitional movement intentions from ECoG data.

Keywords: ECoG data decoding, brain-computer interface, volitional movement intention prediction, FPGA implementation

1. Introduction

Brain-computer interface has developed immensely in recent times. It has reached a point where a subject can use data collected from their brain to actually control external devices. This process involves feature extraction, decoding, signal processing, and actuating [1]. In the

last few years, BCIs have received a lot of recognition in various other fields apart from medical industry. They have gained a lot of popularity in the entertainment such as gaming industry. Electrical stimulations from the brain can be recorded either noninvasively which is called electroencephalography (EEG) or invasively called the electrocorticography (ECoG) signals. These electrical stimulations from the brain can be collected over time and decoded to know more about brain signals and its activities. EEG signals are recorded by placing the electrodes along the scalp of the brain. They can be used to diagnose multiple symptoms such as coma, epilepsy, sleep disorders, etc. ECoG signals are recorded in the cerebral cortex of the brain. In this case, the electrodes are implanted into the brain in order to record brain activities. Since ECoG signals are recorded invasively using implanted electrodes, this type of recording data has an advantage of higher spatial resolution and higher sampling rate than its counterpart. However, since the electrodes in this case are implanted into the brain, this setup involves a surgeon to operate on the subject and place the electrodes inside the skull. This unique feature of the ECoG data has made it more suitable for BCI applications that mainly focus on restoration of sensorimotor functions. A suitable example of this scenario is shown in [2]. Here, a subject with severe motor disabilities is able to control the prosthesis using the ECoG data recorded from the subject's brain activities. The ECoG data obtained from the electrodes have to be processed in the first place in order to interpret the information contained in them and to decode its volitional movements. Techniques such as time-frequency analysis, including power-spectrum analysis and fast Fourier transforms, have been proposed in Ref. [2]. Neuroscientists enjoy such off-line techniques of decoding ECoG cortical data as it enables them to further go in deep, study the brain activities, and map them to the volitional movements that were intended to perform. In Ref. [3], dynamic mode decomposition is proposed which is also an off-line decoding technique. But such off-line techniques cannot be implemented in applications that are focused on restoration of sensorimotor functions as they have to be on the fly and real time. Retrieving one's voluntary movements by sending the spinal stimulations data to paraplegics or by enabling the actions of the prosthetic control requires real-time signal decoding. Such real-time signal decoding circuits and data processing blocks provide the right platform for the abovementioned applications. Any such system would have (i) an analog front-end circuit, which is used to amplify the raw signals recorded and filter the noise; (ii) an analog-to-digital converter (ADC), which as the name suggests, converts the incoming raw analog signals to digital format; (iii) data processing block, which is the most vital of them all as this block is used to decode the digitized brain signals into interpretable format, for example, any movement intention; and (iv) stimulator back-end circuits which are particularly placed to perform the actions or movements that are predicted by the previous data processing block. These actions could be anything such as enabling prosthetic actuators or delivering spinal stimulations for voluntary movements.

2. On-chip computing in BCI applications

Decoding raw ECoG signals to restore sensorimotor function has been a great motivation. The most advanced techniques comprise time-frequency analysis with power-spectrum analysis,

http://dx.doi.org/10.5772/66149

fast Fourier transform [2], and dynamic mode decomposition [3] are used to decode unprocessed ECoG signals. External computational sources that treat the data received either from electrodes or the feedback from sensors can generate the signal for stimulus or can trigger the prosthetic control efficiently. This type of analysis and decoding ECoG data in an off-line way is very slow and not suitable for decoding in real time. Moreover, as these off-line-based computational resources are very complex to implement in terms of area and power consumption, they are impractical for portable applications. Hence, it is essential to design an on-chip decoding system which is handy, power efficient, and also fast enough for real-time use. Many different on-chip techniques are presented for decoding ECoG-recorded signals. An interface between inserted chip and recording and stimulating electrodes was projected in [4], which is portable as well as operates independently. In Refs. [5–8], various on-chip signal recording and processing model can be seen. Such action potential of a basic computing unit (a neuron) was detected that produces the electrical stimuli by the use of time-amplitude discriminator. Mainly focuses on recording and processing (amplifying, filtering, etc.) cortically recorded signals to trigger the action stimuli signals. Another on-chip implementation is based on look-up table (LUT) that generates the corresponding stimulus action (such as eye blinking) by classifying the extracted, amplified, and filtered brain signals [7]. Moreover, in Ref. [9], discrete cosine transform and linear classifier are implemented on hardware to decode the ECoG movement intentions. As explained in Ref. [4], the large number of neurons that reside on the cortical surface controls the hand movements, and these hand movements usually occur in high-dimensional space; hence, giving a typical motor behavior range is still a challenge. Since the classification based on look-up tables and linear classifiers is narrow, developing a better and versatile on-chip classification method is essential that can take care of more complex task.

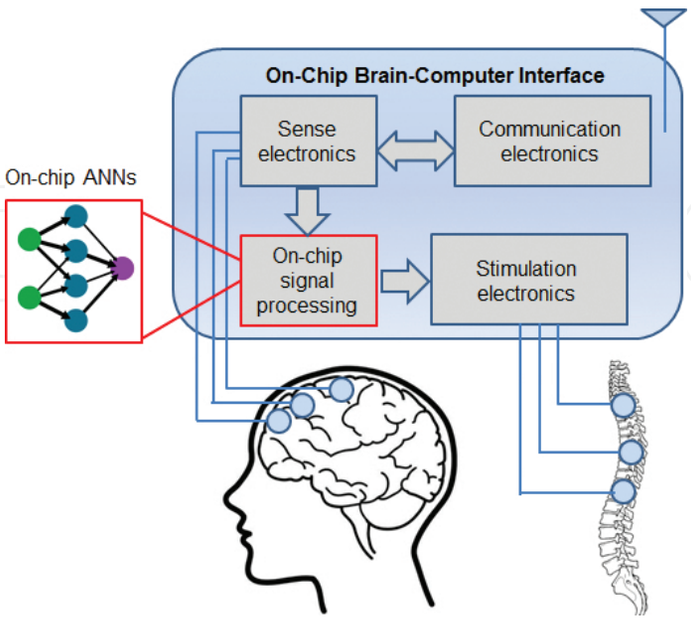

Figure 1. Overview of the BCI system in which the proposed framework is implemented.

Designing of data processing block and implementation are the main focus of this work. Proposed framework has been highlighted in **Figure 1** showing an outline of typical BCI system. The signals are the ECoG signals, recorded from the electrodes placed invasively on the cerebral cortex. There are several parameters that have to be taken care of when designing such circuits. Mainly, the area occupied on the chip should be really small with low power consumption and should be resistant to temperature and voltage variations and of course the limitation of the system on which this has been developed. Thus, it is not feasible to build such off-line hardware-hungry decoding schemes for real-time BCI applications. Our work involves a low-power and area-efficient hardware realization of principal component analysis (PCA) and multilayer perceptron (MLP), implemented on FPGA to show the usefulness of on-chip data decoding model for BCI applications. Openly accessible ECoG recordings and the experimental results from FPGA show the accuracy of 80% for predicting single-finger movement.

3. Feature extraction: principal component analysis

Feature extraction is a process whose aim is to reduce the dimensionality and decrease the complexity of the dataset to a fewer dimensions with the largest amount of information possible. For BCI applications, it is an important requirement that the computational complexity of the system is very less. It must also be robust against noise influences and should only depend on historical data samples. Brain signals depend on different thinking activities that occur in the brain. BCI is considered to be a pattern recognition system that differentiates between different patterns and classifies them into different classes based on the features. The features extracted using BCI not only reflect the similarities to a certain class but also the differences from the rest of the classes. The features are measured using the properties of the signal that contain the information needed to distinguish between different classes.

Feature extraction or dimensionality reduction techniques such as PCA or independent component analysis can be applied to reduce the dimensions of the original brain signal data collected to help in removing irrelevant and redundant information. Such techniques will also reduce the overall computation cost as well.

3.1. Principal component analysis (PCA)

PCA is an effective and powerful tool for analyzing data and finding patterns in it. It is used for data compression, and it is a form of unsupervised learning. Dimensionality reduction methods can significantly simplify and progress process monitoring procedures by projecting the data from a higher dimensional space to a lower dimensional space that exactly characterizes the state of the process. PCA is a dimensionality reduction technique which produces a lower dimensional representation of a given data in such a way that the correlation between the process variables is conserved and is also good in terms of covering the maximum possible variance in the given data. The projection of higher dimensional data to a lower dimensional

http://dx.doi.org/10.5772/66149

data as explained before happens in a least square sense; small inconsistencies in the data are ignored, and only large inconsistencies are considered.

3.2. Characteristics of principal components

1. The first principal component accounts for a maximum amount of variance in the observed variables which means that this component is correlated with most of the observed variables.
2. The second component extracted will have two characteristics:
   a. This component covers most of the variance that was unaccounted for in the first principal component, which means that the second component will be correlated with most of the variables that did not display strong correlation with the first component.
   b. The second characteristic is that it is completely uncorrelated with the first component. The correlation between the two will be zero when it is matched.
3. All the remaining components extracted during the analysis will exhibit the same two characteristics:
   a. Each component calculated will account for a maximum variance of the variables that were not covered by the preceding component.
   b. Each component will be uncorrelated with all the preceding components calculated.

From all the above characteristics, it is clear that with each new component calculated, it accounts for progressively smaller and smaller amounts of variance which clearly explains why only the first few components are generally considered for any data analysis and interpretation. When the analysis is complete and all the principal components are obtained, each of these components will display varying amounts of correlation with the input variables but are all completely uncorrelated from each other.

4. Feature decoding: artificial neural networks

The ECoG-recorded data need to be decoded to be able to detect the intended movement. Once the dimensionality of data is reduced by PCA, we can further decode the data to trigger external devices. Artificial neural network (ANN) is a very good choice for decoding such signals.

4.1. Artificial neural networks (ANNs)

Artificial neural networks are designed to model the data processing abilities of a biological nervous system, which are the major paradigm for data mining applications. The human brain is estimated to have around 10 billion neurons each connected with an average of 10,000 other neurons. The basic cell of artificial neural network is a mathematical model of a neuron represented in **Figure 2**. There are three basic components in an artificial neuron:

1. The connecting links possessing weights to the inputs (analogous to synapses in biological neuron).
2. The weighted input values are summed in an adder with a bias, w_0, $\sum = x_1 * w_1 + x_2 * w_2 + x_3 * w_3 + w_0$.
3. An activation function that maps the output on a neuron.

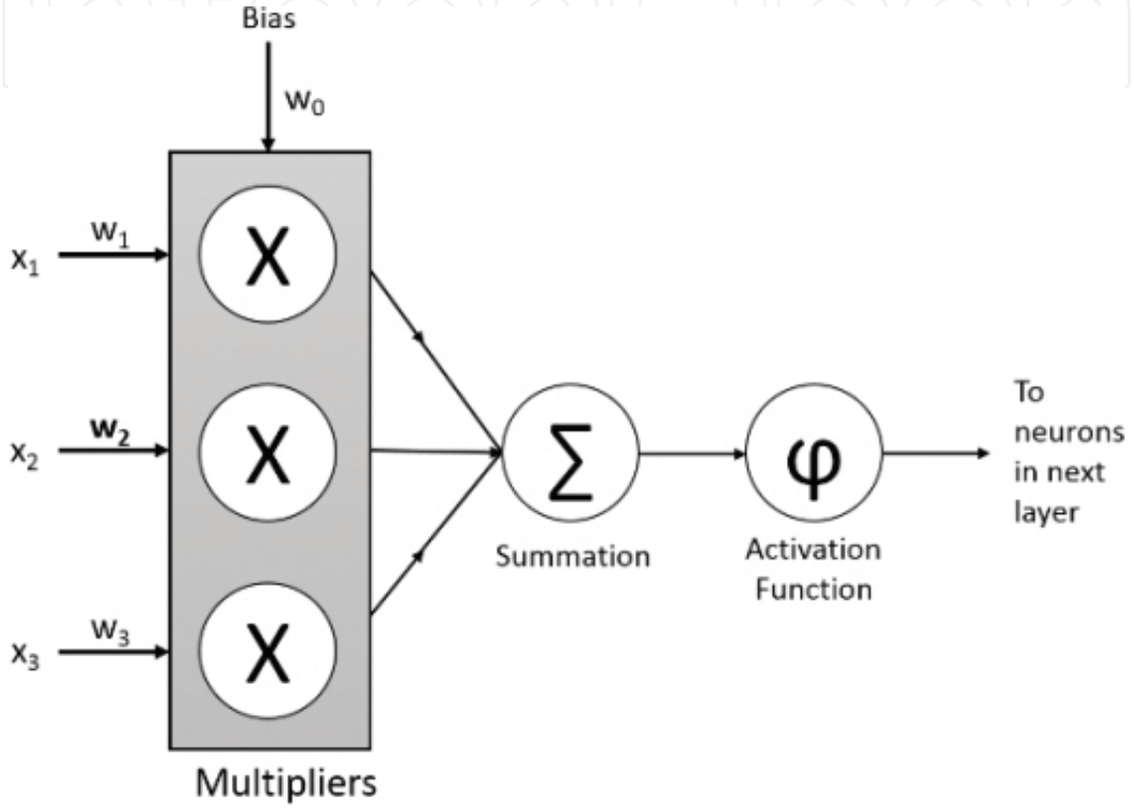

Figure 2. Mathematical model of a basic cell of artificial neural networks.

4.1.1. Multilayer perceptron

The feedforward networks with more than one layer are called multilayer perceptron (MLP). This is a very popular multilayer feedforward architecture. The neurons in each layer of MLP (minimum two layers, one hidden layer, and one output layer) are connected to the neurons of the next layer. The input layer accepts input values and forwards them to the successive layers. The last layer is called the output layer. Layers between input and output layers are called hidden layers. In this work, we adopt the sigmoid (or the logistic) function for implementing activation function. **Figure 3** shows an example of MLP architecture. There is no universal approach to systematically obtain the optimal number of neurons and number of layers. Cross validation is a common practice to obtain the optimal MLP structure, although some other practical constraints such as area and power overhead should also be taken into account when on-chip implementation is considered.

http://dx.doi.org/10.5772/66149

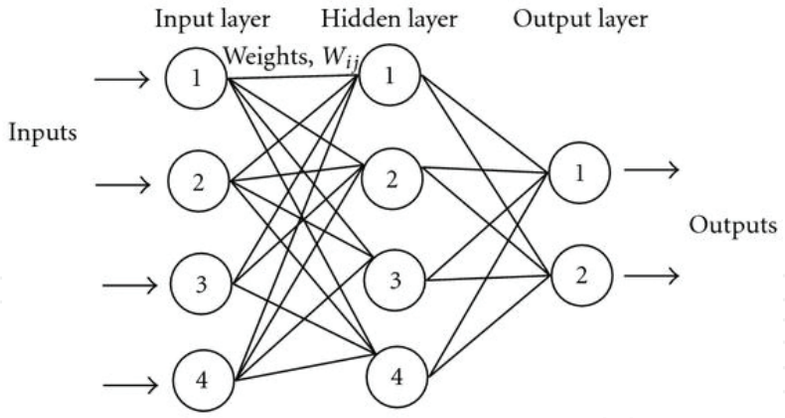

Figure 3. Example of MLP architecture.

Training of MLP consists of tuning the weight values associated with each neuron in an iterative manner such that the outputs of MLP gradually change toward the desired outputs, also known as target values. The most commonly used training algorithm is backpropagation. Backpropagation is an expression for the partial derivative of the cost function with respect to any weight or bias in the network, which tells us how quickly the cost changes when we change the weights and biases. The algorithm requires a desired output (target) for each input to train the neural network. This type of training is known as supervised learning. When no target values are specified during the training, the procedure is then refer to as unsupervised learning.

4.2. Training of artificial neural networks

Training artificial neural networks using backpropagation is an iterative process, which uses chain rule to compute the gradient for each layer having two distinct passes for each iteration, a forward pass and a backward pass layer by layer.

Carrying the forward pass, the outputs of each layer are calculated by first considering arbitrary weights and inputs until the last layer is reached. This output is a prediction of neural nets that are compared to the targets. Based on this, weights are updated in the backward pass, which starts with calculating the error for each neuron in the output layer and then updating weights of the connections between the current and previous layer. This continues until the first hidden layer and one iteration is completed. A new matrix of weights is then generated, and the output is calculated in the forward pass. The input to neural networks is the data that need to be decoded. In the context of BCI data decoding, ECoG signals are recorded from an array of implantable electrodes, which are then processed, and volitional movement intentions are predicted.

5. Proposed framework

Till now, we have shown the theory behind the data processing blocks. Here, detailed explanation of the real-time on-chip ECoG signal extraction and decoding is given. The discussion

includes the overview of proposed design and in-depth description of principal component analysis and multilayer perceptron implementation.

5.1. Overview of the proposed framework

In **Figure 4**, we have shown the overview of different blocks in BCI system, and as mentioned before, our work mainly focuses on the design of data processing block—PCA and MLP. To record the electrical activity from the brain, an array of implantable electrodes is placed invasively on the cerebral cortex, and raw ECoG signals are recorded. The interpretable actions for prosthesis control or simulation by the spinal cord are produced by various preprocessing on-chip blocks after recording of ECoG signals. This includes:

a. Raw signal amplifiers and noise filters combine *an analog front-end circuit*.

b. An analog-to-digital converter (ADC) that converts analog ECoG signals into its digital version.

c. Feature extraction and feature decoding blocks—*data processing block*.

d. A stimulator that sends the spinal stimulations or triggers the prosthetic actuators.

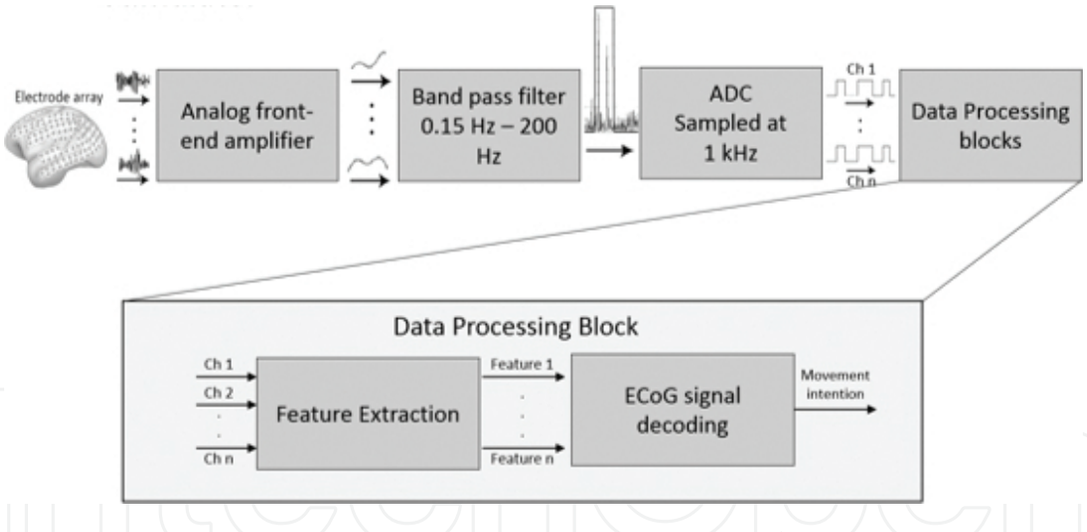

Figure 4. Overview of the BCI system in which the proposed framework is implemented.

Designing the data processing blocks to decode the volitional movement intentions is main focus of this work. More accurate ECoG signals are obtained by intracranial depth electrode technology, which also gives good spatial resolution and high sampling rate. For example, ECoG signals can be recorded using electrodes placed in an array manner. In Ref. [10], each array consists of 62 electrodes. Hence, in real time, it is challenging to decode high-dimensional ECoG data. As shown in **Figure 4**, the two main parts of data processing blocks are feature extraction and ECoG signal decoding. The following sections will explain in detail the hardware friendly implementation of these components.

http://dx.doi.org/10.5772/66149

5.2. Hardware friendly PCA

Principal component analysis (PCA) is a classical data processing technique that retains a minimum set of data from the original set with a maximum amount of variance. It is a popular algorithm that is used in the feature extraction methods. In PCA, the most challenging part is the calculation of the eigenvectors. These eigenvectors can be calculated using the covariance matrix, and this is very challenging [11]. Furthermore, we need to come up with a hardware-implementable algorithm, as our ultimate goal is to implement it onto the FPGA platform. In order to achieve this, we are using a hardware friendly PCA algorithm, which is not only very efficient in terms of implementation but also helps in extracting those features that are very significant for classification. These features are extracted from the recorded ECoG data [11, 12].

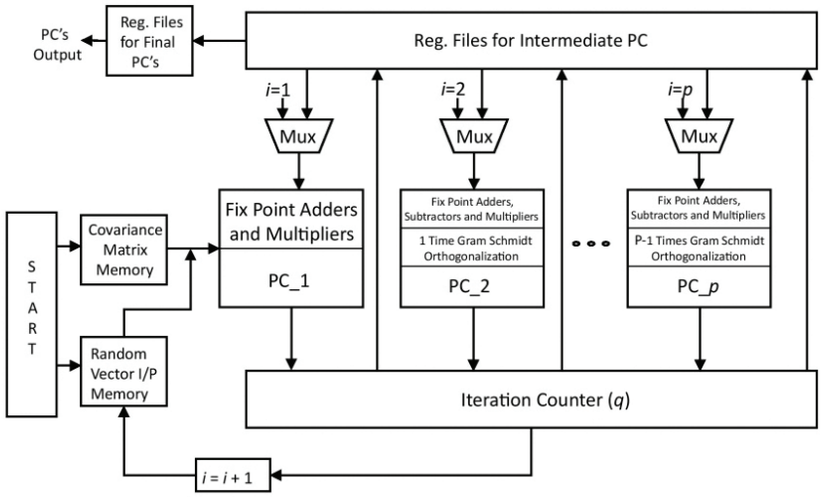

Figure 5. Functional blocks of the hardware friendly version of the PCA.

Figure 5 briefly comprises the functions of the hardware friendly PCA that is used in this system. The input to this algorithm is a covariance matrix and random variables to generate the eigenvectors. These inputs to the algorithm are stored in a look-up table (LUT). The covariance matrix Σ_{cov} is calculated from the input data, which is the recorded ECoG data. This input data is of the order of × n, where n is the number of features and k is the number of samples collected for each feature. The algorithm is designed such that two parameters, viz., the total number of eigenvectors required p ($p \leq n$) and the iteration number to calculate each of the eigenvectors r, have to be declared initially. Once the input to the algorithm is stored and the initial values are declared, the declared random variables are constantly multiplied with the covariance matrix till the iteration number is reached and the first principal component *PC* is obtained. This technique is called eigenvector distilling process [11]. Subsequently, to compute the remaining eigenvectors, they require r iterations as it was declared at the beginning. To compute all the other *PCs* apart from the first *PC*, an additional step other than the eigenvector distilling process is required which is the orthogonal process. We use Gram-Schmidt orthogonalization in this orthogonal process. This process is used to do away with all the previously measured $p - 1$ *PCs* from the current eigenvector and its intermediary values.

All four basic math operations were used to compute these eigenvectors in the original algorithm which is the fast PCA algorithm [12]. They made use of addition, multiplication, and norm operators which include division and square root operations. In Ref. [11], flipped structure is proposed to achieve a minimum power and lesser area requirements. Here the equation $\varphi_p = \varphi_p/\|\varphi_p\|$ which is a norm operation is eliminated and substituted with $\varphi_p = \varphi_p - \left(\varphi_p^T \varphi_j \Big/ \| \varphi_j \|\right)\left(\varphi_j \Big/ \| \varphi_j \|\right)$. This equation is then multiplied with $\| \varphi_j \|^2$. By doing so, the orthogonal process turns into $\varphi_p = \left(\varphi_j^T \varphi_j\right)\varphi_p - \left(\varphi_p^T \varphi_j\right)\varphi_j$. This equation can now be implemented only using adders and multipliers which are much more efficient in terms of hardware implementation than the division and square root operations. However, this implementation drastically escalates the dynamic range of all the values. In Ref. [11], an adaptive level-shifting scheme was proposed to keep the dynamic range within a limit, but since we are implementing this algorithm using fixed-point mathematical operators, these operators automatically keep a check on the dynamic range and hence we have eliminated this adaptive level shifting scheme in our implementation. The algorithm described so far is shown in Algorithm 1 in **Figure 6**. This algorithm is designed to pick p eigenvectors which gives a $k \times p$ feature matrix of M'.

Algorithm 1 Eigenvector Distilling Algorithm

```
procedure EIGENVECTOR_DISTILLING
Select the desired number of eigenvectors p
Select the maximum iteration number r
Initialize input data matrix M
Compute covariance matrix Σ_cov
i ← 1
q ← 1
Initialize a random φp eigenvector
    while i ⩽ p do
        while q ⩽ r do
            φp = Σ_cov φp
            φp = φp − Σ_{j=1}^{p−1} (φp^T φj) φj
            φp = (φj^T φj) φp − (φp^T φj) φj
            q = q + 1
        i = i + 1
end procedure
```

Figure 6. Eigenvector distilling algorithm.

http://dx.doi.org/10.5772/66149

5.3. MLP design

For the classification of data particularly for volitional movement intentions, the design of MLP is the next step once the dimensionality is reduced using on-chip PCA algorithm described above. The structure of MLP contains multiple layers of nodes with each layer fully connected to the next one. The basic computing unit is called an artificial neuron, which is designed using fixed-point multiplication and addition operators. The inputs to these neurons are first multiplied with preferred weights and added. The output of this combination is summed together and is given to a nonlinear differentiable activation function, log-sigmoid here. LUT is used to implement the transfer function. The number of neurons in each layer and total number of layers are reconfigurable. Training the MLP is done off-line manner, and learned weights are updated using random access memory (RAM) to embed on FPGA board.

5.3.1. Fixed-point multiplier

A parameterized fixed-point-signed multiplier is designed for the multiplication operation taking place inside a neuron. The parameters that can be altered based on the design requirements and available bit widths are the bit widths of two operands (WI1 + WF1; WI2 + WF2) and that of the output (WIO + WFO), where WIx is the integral part bit width and WFx is for fractional part. In a normal multiplier operation, the integral and fractional bit widths (WIO and WFO) of output are obtained by adding the operand's integral and fractional bit widths: WIO = WI1 + WI2 and WFO = WF1 + WF2. However, as we see, the bit width is doubled after every operation. To make it hardware efficient, truncation and rounding are done to reduce the bit width according to the needs. In this experiment, integer and fraction bit widths are kept equal for the two operands. Truncation is done in integral part by removing all the extra bits (WI1 + WI2−WIO) and keeping the signed bit. In the fractional part, only required bits (WFO) are kept and truncate all extra lower significant bits. Overflow flag, which represents the incorrect result, is set to 1 if the signed bits are not similar to the truncated bits.

5.3.2. Fixed-point adder

Similar to the multiplier, in case of normal addition operation, the bit width of integer part is equal to one plus the integral bit width of operand having more bits than the other (if WI1 > WI2, WIO = WI1 + 1 else WIO = WI2 + 1), and for the addition of fractional part, bit width is equal to the greater fractional bit-width operand (if WF1 > WF2, WFO = WF1 else WFO = WF2). Similarly, as in the multiplier, we perform truncation and rounding with the overflow flag.

5.3.3. Activation function

As mentioned before, a nonlinear differentiable transfer function is used at the final stage of a neuron. Activation function accepts the addition of product of all inputs with their weights as an input to generate a nonlinear output. The most used activation function is logistic function (log-sigmoid or tan-sigmoid) for multilayer perceptron for pattern recognition. The output of log-sigmoid function generates outputs in range of 0 and 1 as the input of the neuron's net goes from negative infinity to positive infinity, while tan-sigmoid func-

tion ranges between −1 and +1. Log-sigmoid function is an exceptional case of logistic function. The equation and the curve of log-sigmoid function are shown in **Figure 7**:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{1}$$

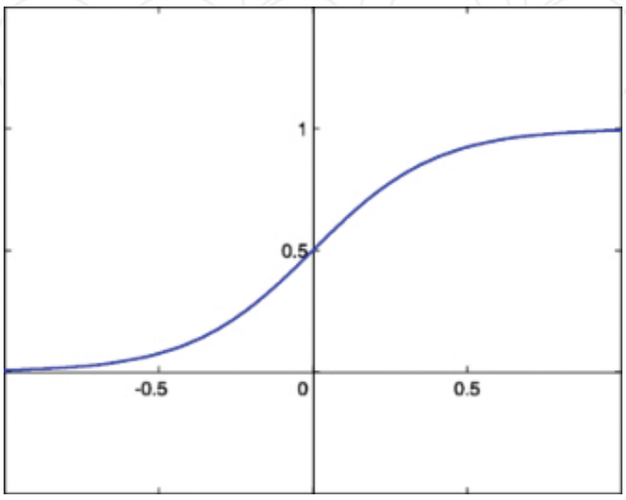

Figure 7. Equation and curve of log-sigmoid function.

The value of x (input to activation function) is considered between −5 and +5 for this work at a 5-bit precision. This will give 32 (2^5) values for every two integers that totals 320 values. These fixed-point binary values of f(x) are stored in LUT, whose address can be represented by 9 bits.

5.3.4. Implementing LUT on hardware

The LUT in our case is used in an efficient manner as compared to LUT usage in general. One way was to use separate LUTs, for input x and for output $f(x)$ with 320 values each, but only one LUT for $f(x)$ is used. By doing this, the area utilization is reduced by half for every neuron, and speed is doubled. The input for the sigmoid function is a 12-bit binary number (six for integer part and six for fractional part) coming from the output of an adder. The consideration of 5-bit precision is to get the address of LUT from the fractional part, ranging from 00,000 to 11,111. The output should be 0 or 1 if the input is less than −5 or more than +5, respectively. Hence, 0 is stored at the first address of LUT for mapping any value less than or equal to −5. So for −2.96875 (111,101:00,001) as input, the 65th address of LUT is the output value, which is (−3 + 5) *32 + 1, where (111,101) or −3 is the input integer part and (111,100) or 1 is the input fractional part. Equation 2 is used to calculate the address, specifically when the input lies in the range −5 and +5:

$$address_{9bit} = \left(input_{integer} + 5\right)*32 + input_{fraction} \tag{2}$$

http://dx.doi.org/10.5772/66149

A number of multipliers and adders (i.e., number of inputs) required to form a neuron depend on the number of neurons in the previous layer.

5.3.5. Implementing MLP on hardware

After designing a neuron, the next task is to form neural networks, which have hidden layers and output layer with a number of neurons interconnected in the defined manner. There is no thumb rule to decide the number of layers and neurons in each layer. A single increase of neuron in hidden layer will increase at least one multiplier and one adder for every neuron in the next layer, which increase the hardware utilization.

These parameters can also be selected based on a criterion called Akaike information criterion (AIC). For better generalization, this statistical approach can be used to determine the optimum number of hidden units in a neural network, as it is complex because of strong nonlinearity. AIC can be represented by the following equation:

$$AIC = \mathrm{n}^{*}\ln\left(\frac{RSS}{n}\right) + 2 * K \tag{3}$$

where n is the number of observations , RSS is the residual sum of square errors, and k is the number of parameters (total number of weights). The lower the value of AIC, the better is the architecture of neural network. On increasing the number of neurons in the hidden layer, AIC may improve up to an extent. After certain number of neurons in hidden layer, AIC starts increasing and changes very less with change of architecture [13].

As shown in **Figure 8**, our architecture has one hidden layer and an output layer with five and three neurons (3 bit output), respectively. The use of delay elements (boxes) is explained later in this chapter.

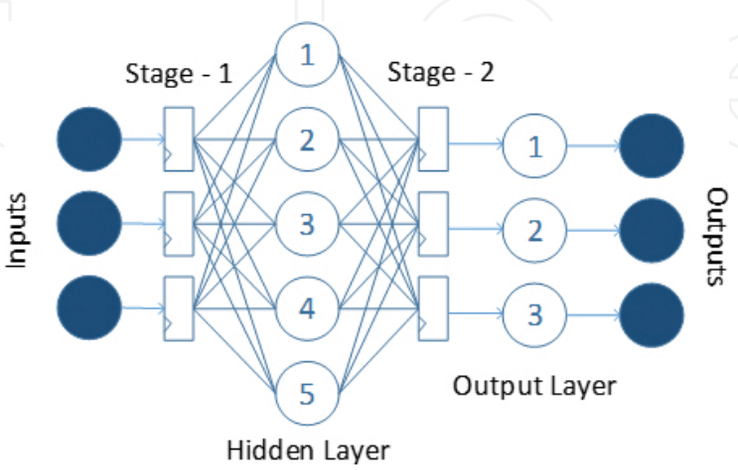

Figure 8. Architecture of the implemented MLP.

The neural networks are pipelined to fully utilize the hardware and give a better frequency of operation. The boxes seen in **Figure 8** are the delay elements (registers) that store temporary values of the previous outputs. Since two stages of pipeline are implemented to increase the throughput with exchange of latency, the frequency of operation is achieved up to 83 MHz.

6. Experimental setup and results

To show the on-chip implementation of the proposed work, we are using openly accessible ECoG data collected for studies related to sensorimotor restoration [10]. We will demonstrate that using our approach, voluntary movements can be decoded efficiently in real time from a high-dimensional ECoG data. This will strongly serve as a strong basement for a completely automated BCI system.

6.1. Experimental setup

In Ref. [10], an off-chip analog front-end amplifier/filter and an ADC were used to amplify and digitize the amplified ECoG signals that were collected using electrode grids. The electrodes were arranged in an array where each array had 62 platinum electrodes. Each of these electrodes was organized in an 8 × 8 manner. Therefore, 62 channels of ECoG data were collected at once, and each of these measurements was measured with respect to scalp reference and ground.

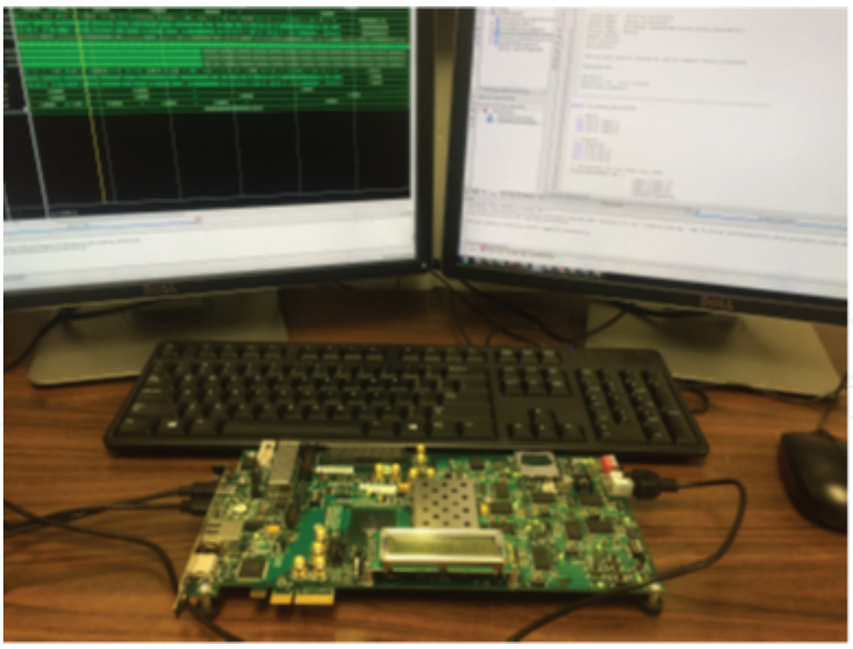

Figure 9. FPGA board used for the experimental study.

A computer display monitor is placed alongside the subject, and the finger to be moved is displayed in this monitor. The subject is asked to move that particular finger, and the ECoG

http://dx.doi.org/10.5772/66149

signals are collected during this movement. There is a 2-second gap between each movement, and during this time, the screen displayed nothing. Each of these movements was also recorded for a 2-second time period. Along with the recording of the ECoG signals, the position of the finger was also recorded. The data collection is explained in detail in [10]. The data set we used is a 400,000 62 matrix, where 400,000 is the number of observations collected for study purpose and each of these observations were collected across 62 channels. Here, our main focus is on the movement and non-movement of the finger. Thus, if a finger is moved, it is classified as 1 and 0 otherwise. The predictor model outputs one class for each of the five finger movements, and the sixth class is when all the fingers are at rest. The output of our model is a 400,000 × 1 matrix. We used a Xilinx ARTIX-7 FPGA kit for demonstrating the proposed model. As explained earlier, we are using the embedded RAM in the FPGA board to store our input values. An RS-232 serial port is used to read back the values from the FPGA after all the computations are finished. We achieved a 83.33 MHz frequency which is a maximum possible frequency we could achieve along with a +0.5 ns of worst negative slack. This can further be optimized to obtain 86 MHz frequency of operation. A snap of the Xilinx Artix-7 FPGA kit used for this work is showed in **Figure 9**.

6.2. Feature extraction based on on-chip PCA

The original dataset is divided into training S_{tr} and validation S_{vat}. S_{tr} is a 240 × 62 matrix where 240 is the samples across 62 channels. Out of 240 samples, we choose 40 samples in random for all the six classes as explained above. Thus, the validation matrix now becomes (400,000 − 240) × 62 for the equivalent classes. The finger positions of the six different classes are shown in **Figure 10** which is plotted as a function of 240 samples recoding time. With the aim of decreasing the size of the input data matrix, the hardware friendly PCA is used to extract features from the input matrix. A total of 240 samples are projected on the first and second principal components obtained after performing PCA. **Figure 11(a)** shows this setup as a scatter plot.

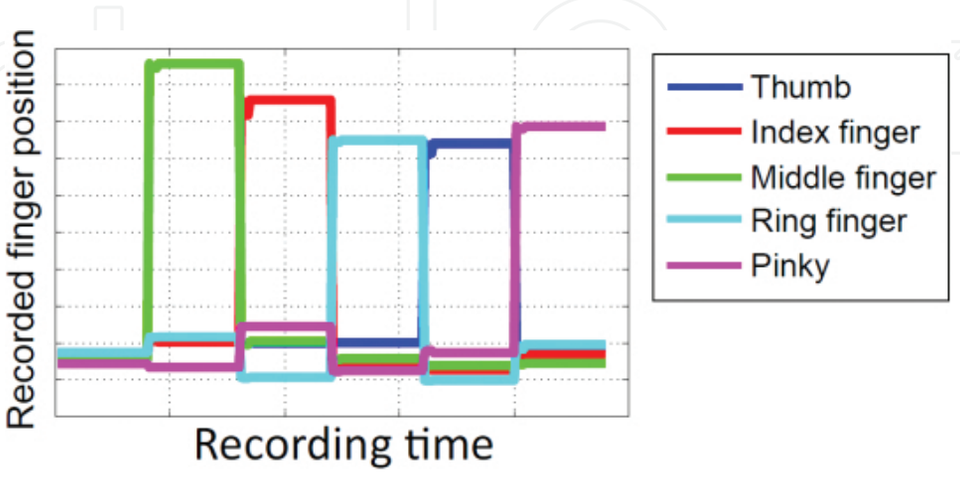

Figure 10. Merged recorded finger positions of the six classes as a function of recording time.

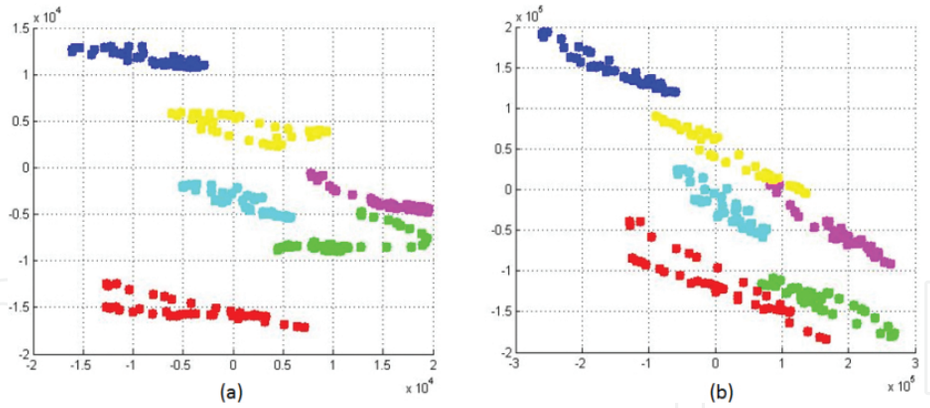

Figure 11. Projection of the 240 training samples onto the first two principle components using (a) the proposed on-chip PCA and (b) MATLAB PCA function based on singular value decomposition (SVD) algorithm.

Each color denotes a different class, and it is clearly evident from this figure that the training samples are all distinguished in this space. The hardware friendly PCA algorithm used in this work is compared with the singular value decomposition (SVD)-based MATLAB PCA function which is shown in **Figure 11(b)**. It is clear that the samples calculated from the proposed PCA algorithm closely match with that obtained from the traditional SVD-based MATLAB function. This goes on to prove the accuracy and the effectiveness of the proposed on-chip PCA algorithm. The number of principal components required to represent the reduced dataset is determined by the sum of the amount of variance covered by each of the principal components. In our case, the first three principal components add up to 80% of the total variance in the dataset. Considering more than three principal components would not increase the variance significantly but rather increase the computational capability of the algorithm in multiple folds in terms of hardware utilization.

Furthermore, **Figure 12** displays the mean squared error (MSE) for the first three principal components with different iterations by comparing the principal components obtained from the MATLAB function (which is used as a reference) and the algorithm used in this work. We obtained MSE values lesser than 0.2 for the three principal components computed with a maximum of 12 iterations to show the efficiency of the proposed framework.

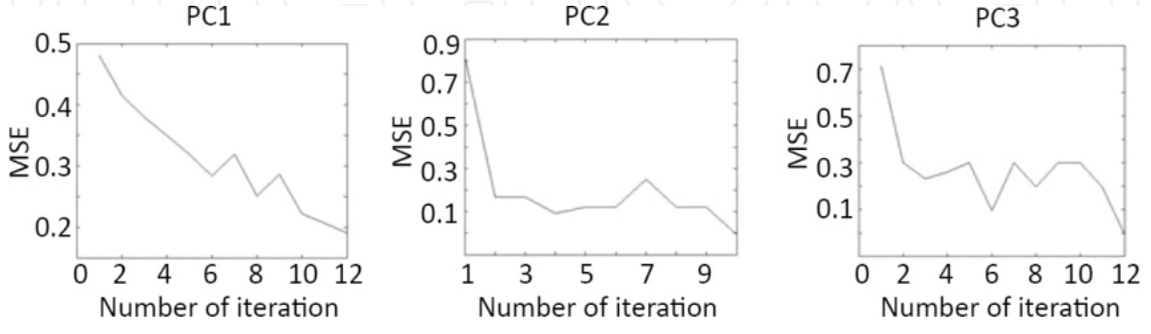

Figure 12. Mean square error of the first three principle components as a function of iterations by taking the principle component values computed using MATLAB SVD algorithm as the baseline.

http://dx.doi.org/10.5772/66149

6.3. Classification based on on-chip MLP

In this work, we have tested different structures of neural networks with different numbers of neurons in hidden layer and then calculating error of classification. For the output layer, three neurons are chosen corresponding to the three outputs of 1 bit each. For the hidden layer, 2–10 neurons work good for most applications. Testing for 3–8 neurons, three neurons give AIC equal to −929, and for four neurons, the value is −1082. This value is decreased up to −1286 for five and six neurons. But the total number of weights increased to 30 ((3*5) + (5*3)) for five neurons and 36 ((3*6) + (6*3)) for six neurons. Further increase in neurons will not show much improvement in AIC, though will increase the hardware utilization as the number of weights increases by 6 for each neuron and hence the computing elements (adders and multipliers) in the next stage.

Resources	Utilization	Available	Utilization %
LUT	525	133,800	0.39
FF	6	267,600	0.01
DSP	22	740	2.97
IO	63	400	15.75
BUFG	1	21	3.12

Table 1. Summary of area utilization of the proposed architecture.

The neural network is trained by giving the first three principal components from the total of 62 as inputs, 40 samples for each class, and a data matrix of size 240 × 3. The implemented design with one hidden layer having five neurons and an output layer having three neurons has two-stage pipelining. The clock period is 12 ns so as the throughput; two stages will increase the latency to 24 ns. One hundred percent accuracy is reached for the training set (240 samples). The remaining validation set of 399,760 (400,000 − 240) samples are given to the trained neural network. This data gives the correct classification accuracy, which is 82.4%. Since we have considered all the noise and perturbations during recording, this is reasonable.

	Bit-widths	Hidden layer neurons	Frequency (MHz)	Power (mW)	Classification accuracy
1	12 (6.6)	5	83	199	82.485%
2	12 (6.6)	5	20	158	82.485%
3	12 (6.6)	4	20	156	65.87%
4	12 (6.6)	3	20	155	76.42%
5	16 (11.5)	5	33	171	80%

Table 2. Summary of difference performances for five different bit-width values.

The power consumption in this architecture is 152 mW. The area utilization is summarized in **Table 1**. As seen in the table, the area utilization of the proposed architecture is lesser than 25% of the available resources; this can be a good lead for future application-specific integrated circuit (ASIC) design development which can lead to even less power consumption.

We have also tried for various MLP architectures with different bit widths that give different accuracy, power, speed, and area. **Table 2** summarizes difference performances for five different bit-width values. The bit-widths column represents the number of bit widths used to represent data including the covariance matrix, intermediate results from the algorithm, and also the final principal components that are computed. They are represented in "integer length and fractional length" form.

6.4. Discussions

The bit width that we choose has a direct impact on the accuracy of the algorithm and also its power consumption. The bit width of the input covariance matrix, the intermediate results of bit width and also the bit width of the output of the algorithm need to be considered primarily in order to determine the accuracy of the algorithm for our applications. Second, the number of principal components needs to be carefully selected to improve computational efficiency. Third, the number of iterations required to compute each of the principal components should also be optimally chosen.

For a given operating frequency, the silicon area utilization and the power consumption mainly depend on the first parameter, whereas the processing capability of the algorithm is influenced by the second and third parameters. Processing capability is mainly determined by the number of channels that can be trained using the PCA algorithm under a given amount of time. Power consumption for different bit widths can be kept constant with reduced frequency or reduced speed of execution. When higher bit widths are chosen for the sake of accuracy, the area required for covariance matrix memory, register files, and processing units increases drastically in order to store more numbers of bits and process more data. It can be also observed that the power consumption increases with higher frequencies.

7. Conclusion

This chapter presents a structure of on-chip computation that decodes ECoG brain signals in a BCI system, serving a pathway to developing a real-time BCI system. The two main blocks of our proposed decoding model are a hardware friendly PCA model and an artificial neural network (ANN). Openly accessible ECoG recordings and the experimental results from FPGA show the accuracy of over 80% for predicting single-finger movement.

Acknowledgements

This project was carried by Award Number EEC-1028725 from the National Science Foundation. The author takes the full responsibility of the content, and the official views of the National Science Foundation are not represented.

Author details

Mradul Agrawal, Sandeep Vidyashankar and Ke Huang*

*Address all correspondence to: khuang@mail.sdsu.edu

Department of Electrical and Computer Engineering, Center for Sensorimotor Neural Engineering (CSNE), San Diego State University, San Diego, CA, USA

References

[1] G. Pfurtscheller, and C. Neuper. Motor imagery and direct brain-computer communication. Proceedings of the IEEE. 2001; 89(7):1123–1134.

[2] T. Yanagisawa, M. Hirata, Y. Saitoh, T. Goto, H. Kishima, R. Fukuma, H. Yokoi, Y. Kamitani, and T. Yoshimine. Real-time control of a prosthetic hand using human electrocorticography signals. Journal of Neurosurgery. 2011; 114(6):1715–1722.

[3] B.W. Brunton, L.A. Johnson, J.G. Ojemann, and J.N. Kutz. Extracting spatial-temporal coherent patterns in large-scale neural recordings using dynamic mode decomposition. Journal of Neuroscience Methods. 2015; 285:1–15.

[4] A. Jackson, C.T. Moritz, J. Mavoori, T.H. Lucas, and E.E. Fetz. The neurochip BCI: towards a neural prosthesis for upper limb function. IEEE Transactions on Neural Systems and Rehablitation Engineering. 2006; 14(2):187–190.

[5] S. Zanos, A. Richardson, L. Shupe, F.P. Miles, and E.E. Fetz. The neurochip-2: an autonomous head-fixed computer for recording and stimulating in freely behaving monkeys. IEEE Transactions on Neural Systems and Rehabilitation Engineering. 2011; 19(4):427–435.

[6] M. Azin, D.J. Guggenmos, S. Barbay, R.J. Nudo, and P. Mohseni. A battery-powered activity-dependent intracortical microstimulation IC for brain-machine-brain interface. IEEE Journal of Solid-State Circuites. 2011; 46(4):731–745.

[7] T. Chen, K. Chen, Z. Yang, K. Cockerham, and W. Liu. A biomedical multiprocessor SoC for closed-loop neuroprosthetic applications. IEEE International Solid-State Circuits Conference (ISSCC). Philadelphia, PA, USA. 2009; 434–435.

[8] A.T. Avestruz, W. Santa, D. Carlson, R. Jensen, S. Stanslaski, A. Helfenstine, and T. Denison. A 5 micro-w/channel spectral analysis IC for chronic bidirectional brain-machine interfaces. IEEE Jounal of Solid-State Circuits. 2008; 43(12):3006–3024.

[9] M. Won, H. Albalawi, X. Li, and D. E. Thomas. Low-power hardware implementation of movement decoding for brain computer interface with reduced-resolution discrete

cosine transform. IEEE Engineering in Medicine and Biology Society (EMBC). Chicago, IL, USA. 2014; 1626–1629.

[10] G. Schalk, J. Kubanek, K. Miller, N. Anderson, E. Leuthardt, J. Ojemann, D. Limbrick, D. Moran, L. Gerhardt, and J. Wolpaw. Decoding two-dimensional movement trajectories using electrocorticographic signals in humans. Journal of Neural Engineering. 2007; 4:264–275.

[11] T. Chen, W. Liu, and L. Chen. VLSI architecture of leading eigenvector generation for on-chip principal component analysis spike sorting system. 30th Annual International IEEE EMBS Conference. Vancouver, BC, Canada. 2008; 3192–3195.

[12] A. Sharma, and K. Paliwal. Fast principal component analysis using fixed-point algorithm. Pattern Recognition Letters. 2007; 28:1151–1155.

[13] G. Panchal, A. Ganatra, Y.P. Kosta, and D. Panchal. Searching most efficient neural network architecture using Akaike's information criterion (AIC). International Journal of Computer Applications. 2010; 1(5), 41–44.