Hindawi
IET Computers & Digital Techniques
Volume 2024, Article ID 5596468, 15 pages
https://doi.org/10.1049/2024/5596468

Research Article

An Efficient RTL Design for a Wearable Brain–Computer Interface

Tahereh Vasei, Mohammad Ali Saber, Alireza Nahvy, and Zainalabedin Navabi

School of Electrical and Computer Engineering, College of Engineering, University of Tehran, North Kargar, Tehran 1417614411, Iran

Correspondence should be addressed to Tahereh Vasei; tahere.vasei@ut.ac.ir

Received 22 July 2023; Revised 3 February 2024; Accepted 20 February 2024; Published 8 March 2024

Academic Editor: Seok-Bum Ko

This article proposes an efficient and accurate embedded motor imagery-based brain–computer interface (MI-BCI) that meets the requirements for wearable and real-time applications. To achieve a suitable accuracy considering hardware constraints, we explore BCI transducer algorithms, among which Infinite impulse response (IIR) filter, common spatial pattern, and support vector machine are used to *preprocess*, *extract features*, and *classify data*, respectively. With our hardware implementation of these tasks, we have achieved an accuracy of 77%. Our system is designed at register transfer level (RTL) targeting an ASIC implementation, which significantly decreases power consumption, latency, and area compared to the state-of-the-art (SoA) architectures for embedded BCI systems. To this end, we fold IIR filters using time-shared and RAM-based techniques and use hardware-friendly algorithms for the implementation of other tasks. The RTL design is realized on 45 nm CMOS technology consuming 4 mW power and 0.25 mm^2 area, which outperforms the SoA platforms for embedded BCI systems. To further illustrate the outperformance of our design, the proposed architecture is implemented on Virtex-7 field program gate array as a prototyping platform consuming 6 μJ energy with 1.52% area utilization.

1. Introduction

Brain–computer interface (BCI) provides a nonmuscular channel between the brain and the external environment to send commands to an external device. BCI enables humans to control external devices without physical movements by solely thinking about desirable movement, called motor imagery (MI) [1]. BCIs are divided into two groups, invasive and noninvasive. The majority of noninvasive systems work based on electroencephalogram (EEG) signals.

Despite the number of designed BCI systems, due to their computational complexity and resource requirements, these systems are limited to personal computers and cloud computing with high-performance computers. However, wearable and real-time BCIs have recently attracted academic and industrial research. In addition, it is necessary to move BCI technologies from the laboratory to the real world and daily life of people. Therefore, because of the importance of power consumption, area, latency, and wearability in BCI applications, they are implemented in the embedded platform.

Embedded BCI systems are implemented as software architecture, hardware architecture, or hardware/software architecture [2]. There are numerous BCI applications for functional replacements, such as wheelchair control and prosthetic limb, which are used by people with disabilities. The majority of BCI systems are implemented as software code embedded within a microcontroller, such as ARM, Nios, and MicroBlaze [2]. Lin et al. [3] proposed an EEG-based smart living environmental control system to adjust the living environment using algorithms such as dawn sampling, Hanning window multiplier, and ICA decomposition, which was implemented on a dual-core processor within OMAP1510 platform. The accuracy, delay, and power consumption of this system were 78%, 2 s, and 1 W, respectively. An field program gate array (FPGA)-based P300 speller was developed in [4] using an integrated forward filter and fisher linear discriminator algorithm. This system was codded in C/C++ and implemented on the Xilinx Spartan BE-FPGA board. The proposed system reached an accuracy of 65.37% without any report of power consumption and run-time.

Jiang et al. [5] used a software code and implemented it on an iPhone as an embedded platform to control the wheelchair using brain signals achieving 61.6% accuracy, 6W power consumption, and 34 ms delay. Li and Chung [6] proposed an embedded platform for detecting the level of driving drowsiness. The C/C++ code of the algorithms such as infinite impulse response (IIR), fast Fourier transform (FFT), and support vector machine (SVM) used in this system was implemented within STM32F103CB, LMC6464, and L3G4200D platforms based on ARM processors. The accuracy of this system was 96%, while consumed 9 W power. Software architecture is useful for prototyping to achieve a reasonable time and cost. However, for BCI implementation, it does not meet critical time and power consumption constraints.

To deal with timing constraints that cannot be satisfied by the software solution, some articles proposed hardware/software architecture. In this solution, the critical components of BCI system are implemented in the register-transfer level (RTL), and the noncritical components are kept in high-level abstraction. This enables the development of a program in a much more user-friendly programing context and is generally independent of the computer's hardware architecture. Therefore, the time-consuming components such as preprocessing ones are implemented on the hardware accelerator, and the rest of the components run on an embedded softcore processor. Belwafi et al. [7] proposed the prototype of a BCI system for controlling home devices using hardware/software architecture. In this system, the time-consuming block was the artifact removal component implemented as a hardware accelerator. Other components of the system ran on an embedded Nios-II softcore processor. To validate their filtering approach, an FPGA-based platform was used. The accuracy was reported at 94.6%; however, the power consumption exceeded 1 W. Belwafi et al. [8] used the hardware/software solution to propose an embedded BCI system to implement an adaptive filter bank for BCI systems. Detecting MI signals using a dynamic filter based on the weighted overlap-add (WOLA) technique was defined as a software code embedded within FPGA-based hardware architecture, running on the integrated core processors. They achieved an accuracy of 76.80% under the power consumption of 0.7 W.

In applications with timing and power constraints such as wearable applications for healthcare, software, and hardware/software architectures cannot meet these constraints. The pure hardware architecture is appropriate to meet crucial requirements of wearability such as low power consumption, area, and cost. Some articles implemented frequently used BCI algorithms. For instance, Palumbo et al. [9] introduced a hardware implementation for EEG acquisition and preprocessing components. They presented the design and implementation of spatial filtering, known as independent component analysis (ICA), using the compact-RIO platform. Karkon et al. [10] focused on a computational algorithm known as canonical correlation analysis (CCA), which required numerous complex matrix transformations. This algorithm is widely used in steady-state visual evoked potential (SSVEP)-based BCI systems. They used some simplifications and developed a fully hardware fixed-point CCA engine to achieve less power consumption and exploit it in real-time applications. This CCA engine was synthesized on an FPGA platform. Some articles used pure hardware implementation for a complete BCI system. For example, Malekmohammadi et al. [11] suggested an efficient hardware implementation for MI-based BCI systems. They used a DC block, surface Laplacian, separable common spatial spectral pattern (SCSSP), linear discriminant analysis (LDA), and SVM as different parts of the system. Their proposed system was tested on a Virtex-6 FPGA and achieved 80.55% accuracy for two classes and 0.09W power consumption.

In addition to conventional algorithms, feature extraction and classification can also perform as a single processing block using deep learning (DL). Widely used architectures in MI-BCI systems are convolutional neural networks (CNNs), RNNs, stacked autoencoders, and deep belief networks, which outperform traditional algorithms such as SVM and common spatial pattern (CSP) in terms of accuracy [12].

This article proposes an embedded implementation of a BCI system based on MI tasks with exploring in different RTL designs to meet the real-world requirements for BCI applications such as timing constraints, low power consumption, low cost, and wearability besides accuracy. The main contributions of this paper are as follows:

(1) Exploring among BCI transducer algorithms for the main components of a BCI system such as preprocessing, feature extraction, and classification optimizing their hardware cost besides accuracy.
(2) Designing components in RTL to meet power consumption, area, and latency constraints.
(3) Proposing methods to reduce the area and hardware resources without reduction in the number of channels and accuracy such as time sharing.
(4) Using pipelining to reduce latency and hardware-friendly formula for computationally heavy and time/power-consuming algorithms.
(5) Using ASIC as an efficient target platform to implement the proposed system to meet wearable and real-time requirements. Therefore, the proposed architecture is realized on 45 nm CMOS technology as well as synthesized on an FPGA as the prototyping platform.

The rest of this paper is organized as follows. Section 2 introduces all required components for a BCI system. Preevaluation of design in software platforms is explained in Section 3. Section 4 depicts the RTL designs of all components and the complete system. Section 5 presents and discusses a wide range of experimental results to demonstrate the efficiency and applicability of the proposed MI-BCI system, and the paper is concluded in Section 6.

2. An MI-BCI Block Diagram

The purpose of a BCI is to detect and quantify the characteristics of brain signals and to translate measurements into

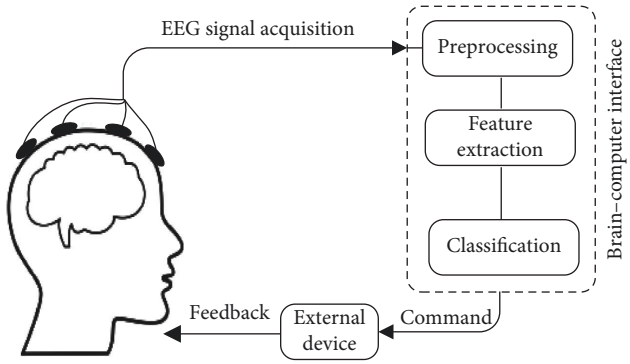

FIGURE 1: Brain–computer interface system.

desired commands in real time [13]. Three steps are required to achieve this goal: preprocessing, feature extraction, and classification. First, the signal-to-noise ratio of recorded signals increases in the preprocessing step [14]. Then, the signals are fed into the feature extraction algorithm to extract the features in the time and frequency domains. Finally, the features are translated by a classifier to convert the independent variables to dependent ones [15]. Figure 1 shows the components of a BCI system. Different methods have been proposed by researchers for each of these three sections. One type of preprocessing, which is commonly used in BCI systems, is spatial filtering, such as finite impulse response (FIR) and IIR filters. It is worth noting that IIR filters are more efficient than FIR ones from a computational point of view; however, their use is recommended when throughput is desired. Therefore, they are utilized when a causal filter is needed. On the other hand, FIR filters are easier to control, stable, and provide a clear pass band. However, their implementation is not cost-effective [16].

As for feature extraction, there are different methods including CSP [17–21], augmented CSP [22], FFT, and wavelet. In Aggarwal and Chugh's [23] study, these methods were compared. According to this comparison, FFT has a good frequency accuracy, and its speed is higher than the other methods. This method has many problems including adaptivity, localization of frequency, and timing window. The worst problem of this method is that it does not consider time information. The limitation of the wavelet method is that the proper mother wavelet is required to be selected. By reviewing the articles, it can be seen that the CSP method is used more than the others thanks to its suitability for multichannel signal analysis, but it is not able to handle temporal dynamics.

In the field of classification, various methods have been investigated including SVM [24, 25], CNN [26–29], deep neural networks, and LDA. Among these methods, SVM has better generalization properties and it is insensitive to dimensionality. However, it is not suitable for handling the dynamic nature of the signals. Neural networks balance the accuracy and speed, but they have large computational complexity for training and testing. The LDA method requires a few calculations, which makes it easier to use, but it is not suitable for nonlinear signals.

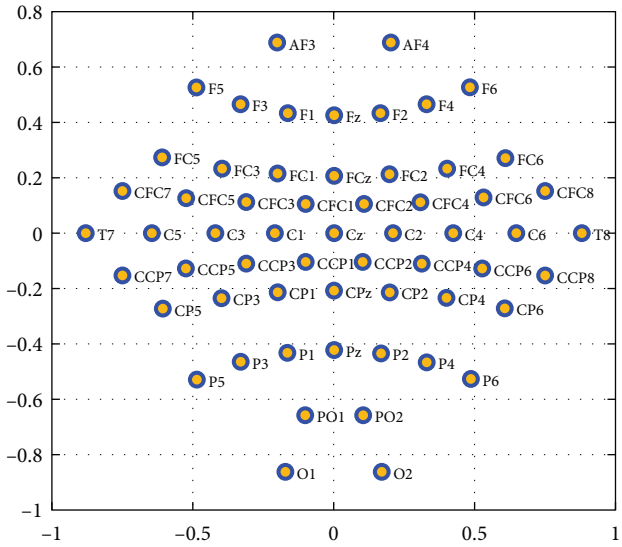

FIGURE 2: Location of electrodes on the head.

3. Predesign Evaluation

In this section, first, the dataset used to verify the proposed design is explained. Then, for each component of the system, several methods are investigated to achieve the proper method and calculate the coefficients. Finally, the coefficients obtained from the train phase are saved to be used in the test phase. To compare the performance of these systems, all the experiments are performed in MATLAB.

3.1. Dataset Description. In this work, the publicly available BCI Competition IV-2a [30], the dataset is used, containing EEG recordings of seven subjects. The goal of "BCI competition IV" is to validate the signal processing and classification methods for BCIs. Figure 2 shows the 2D position of the electrodes.

3.2. Choosing Integer and Fractional Bits. In the proposed architecture, all intermediate calculations and results are performed in a 16-bit fixed-point numeric system. We proposed a technique to find the best integer and fraction length of data for each computational block. To this end, first, each block of the system is modeled in MATLAB and then the final accuracy of the system is checked. Further accuracy check for different integer and fractional length of data indicates that the filter block achieves better accuracy with higher integer length while the other components behave the exact opposite. Hence, 5-bit integers and 11-bit fractions are considered for the filters, while other parts of the system require 2 and 14 bits for the integers and fractions, respectively. It should be noted that in sensitive calculations such as the variance unit, a longer bit length is considered. In the proposed technique, the data are categorized based on the minimum number of bits required for the binary representation of the integer part of the data. Figure 3 shows the percentage of data loss versus the corresponding minimum number of bits. For example, by considering 5 bits for the integer part of the input data, 3% of the data is lost. According to this diagram, no data are lost for the input data with 16 bits.

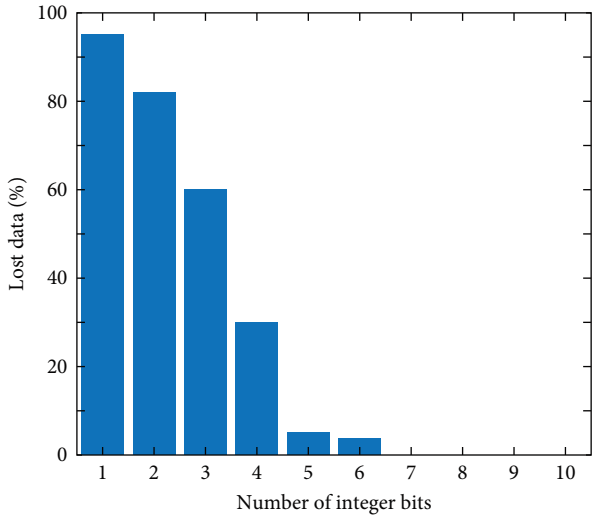

Figure 3: The percentage of loosing data in different number of integer bits.

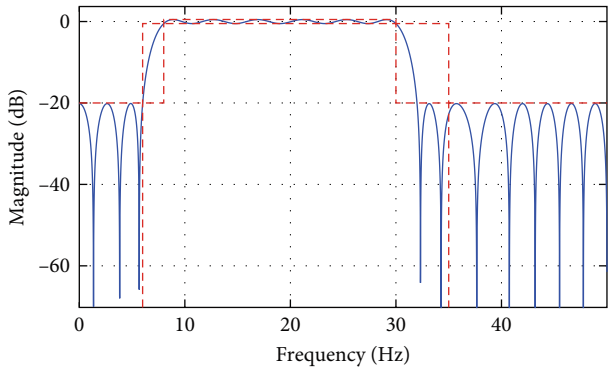

Figure 4: Frequency response of FIR filter.

3.3. Preprocessing. Recording and processing brain EEG is the key to a BCI-based system. Recorded EEG often has environmental noises. These noises originate from other EEG channels and electrical equipment noise. The first step of processing the EEG is purifying each EEG signal channel from all unwanted noises and selecting the proper frequency bandwidth. EEG contains five major brain waves: gamma, beta, alpha, theta, and delta. These brain waves have different frequency ranges from 0.5 Hz to above 30 Hz. Studies indicate that alpha and beta brain waves have all the necessary information for MI tasks. Therefore, filter bandwidth should pass alpha and beta brain waves (8–30 Hz) and eliminate others along with environmental noises.

To select the proper frequency bandwidth (8–30 Hz) including alpha and beta rhythm, the frequency response of the FIR and IIR filters were compared. Figure 4 shows that FIR has a good attenuation capability, but the degree of the designed filter is 48, which has a high hardware cost. However, according to Figure 5, to implement Chebyshev IIR, the minimum suggested order is 8. Chebyshev filter is closer to the ideal filter than other IIR filters since it has a higher attenuation capability because of its high ripple.

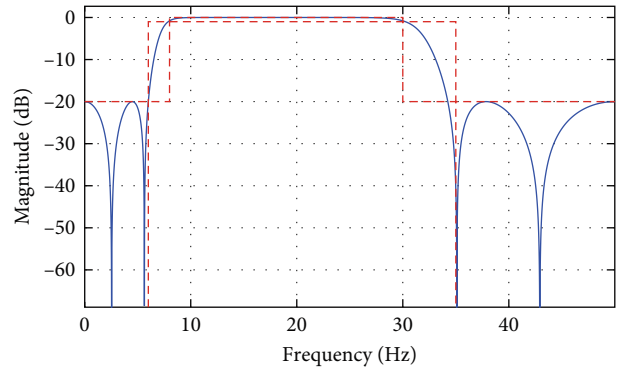

Figure 5: Frequency response of Chebyshev IIR filter.

3.4. Feature Extraction. The next step in processing the EEG signal is extracting features from raw filtered data. This process helps us to focus on the extracted features instead of raw data and leads us to achieve higher accuracy with simple classifiers. Consequently, using feature extraction techniques allows us to reduce silicon area and power consumption. By comparing the CSP and FBCSP methods, the accuracy of FBCSP is just 3% more than that of CSP, while FBCSP uses more hardware resources than CSP. Therefore, CSP feature extraction is used. This feature extraction technique separates the variance of each class on different axes and allows us to use a simpler classifier like SVM. Each channel was considered a feature. This algorithm aims to extract m optimal channels from n available channels. In the selected dataset, each trial includes 59×400 data which is converted to 2×400 after applying the CSP. This algorithm affects the variance in each class. Figure 6 shows that the variance of data in one class increased along the Y-axis and decreased along the X-axis. In contrast, the variance of other classes increased along the X-axis and decreased along the Y-axis. The salient feature that distinguishes two classes in the CSP output is the variance of data. Hence, by applying variance to the output, the output data of CSP become more separable. This is an important step to increase the accuracy of classification. Figure 7 shows the output of variance that feeds the classifier.

3.5. Classification. The evaluation parameters, such as accuracy are compared to choose the best classifier. Table 1 shows that KNN offers the highest accuracy than the others. It is notable that on average the difference between the accuracy of classifiers is just around 5%. Therefore, the classifiers are compared on the basis of hardware costs. Nonlinear SVM and KNN use all train data for each test data, so the hardware has to process as much as the train data. Using these classifiers is not suitable for hardware implementation because the dependence on the training data is high, and nonlinearity causes an increase in hardware resource usage. The only linear classifier is the linear SVM, the test phase of which is independent of the train data that only relies on the number of classes. As a result, the linear SVM is suitable for the hardware implementation of BCI systems.

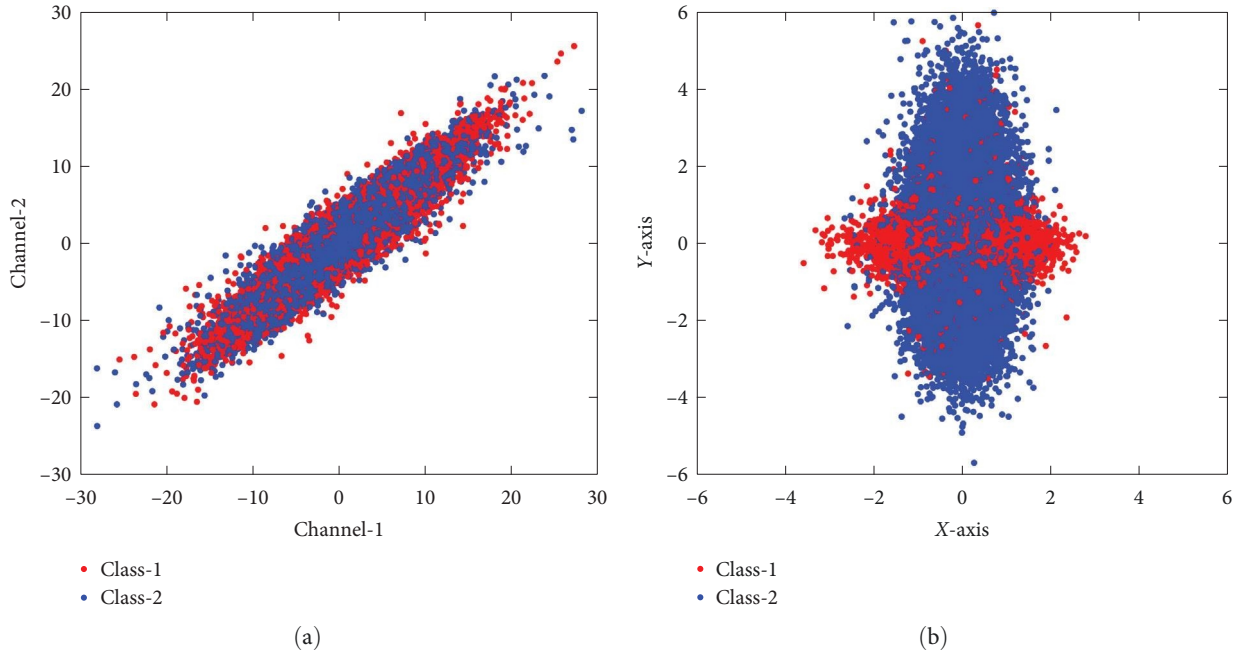

Figure 6: Data distribution on input and output of CSP block. (a) Distribution of raw data before applying CSP. Because of huge dimension of data, only two channels are drawn. (b) Distribution of extracted features after applying CSP.

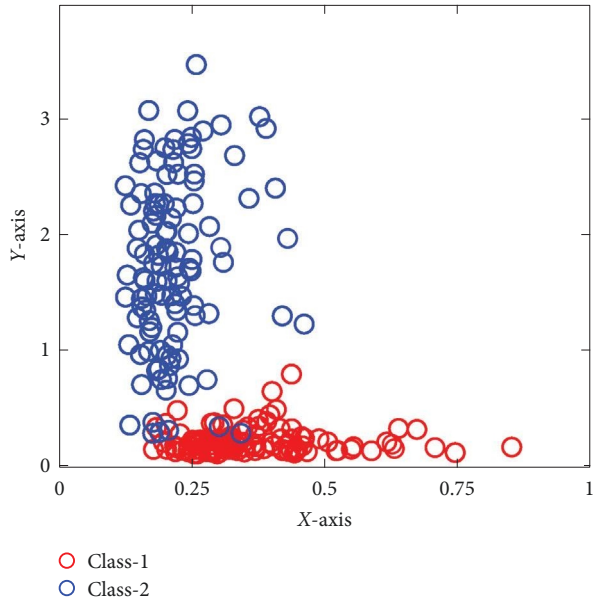

Figure 7: Distribution of classes after applying variance.

Table 1: Classification accuracy of different classifiers.

	Subject 1	Subject 2	Subject 3	Subject 4	Subject 5	Subject 6	Subject 7	Average
SVM	81	69	75	78.3	82	77.6	79.3	77.45714
KNN	85	74.8	77	82	84	80	82	80.68571
LDA	80	63.3	73.3	83.3	80	75	73.3333	75.4619

Table 2: Validation parameters of different subjects.

	Subject 1	Subject 2	Subject 3	Subject 4	Subject 5	Subject 6	Subject 7	Average
Accuracy	81	69	75	78.3	82	77.6	79	77.4
Sensitivity	76.6	73	70	76.6	80	73	76.6	75.1
Specificity	86	66	80	80	83.3	80	83.3	79.8

Table 2 demonstrates an accuracy of 77.4%, sensitivity of 75.1%, and specificity of 79.8% through k-fold cross-validation. The use of fivefold cross-validation is a strong indicator of the system's robustness. This method ensures that the performance metrics are not biased toward a specific subset of the data. By dividing the data into k equally sized folds and iteratively training and testing the model on different combinations, we minimize the likelihood of overfitting and provide more reliable performance estimates. A balanced combination of sensitivity and specificity is crucial for a robust BCI system. In this case, the sensitivity of 75.1% suggests that the system can correctly identify 75.1% of the true positive instances, while the specificity of 79.8% indicates that the system can correctly identify 79.8% of the true negative instances. This balance between sensitivity and specificity is essential for minimizing both false positive and false negative rates, which contributes to the overall reliability of the system. The reported performance metrics suggest that the proposed BCI system has the potential to generalize well to new data. This is particularly important in the context of BCIs, which often face challenges related to intersubject and intrasubject variability. The robust performance across different folds of data during cross-validation supports the system's ability to adapt to varying conditions and user-specific characteristics.

To compare the proposed MI-BCI system with other methods that use the same dataset, other machine learning methods like effect-size based CSP (E-CSP) [31], spatiotemporal filtering strategy [32], Parzen window based methods [33], and spatiospectral feature representation [34] achieve an accuracy of 80.56%, 79.6%, 86.01%, and 86.96%, respectively. In addition, DL-based methods proposed in Schirrmeister et al.'s [35–38] study achieve an accuracy between 80% and 86%. While these methods improve their accuracy, they might be computationally more demanding and would require more resources due to their increased complexity. The proposed MI-BCI system offers advantages in terms of power consumption and area, making it a more energy-efficient and compact solution. However, it may not achieve the highest accuracy compared to other methods. Depending on the specific application and constraints, such as power and space limitations, the given approach might be more suitable compared to more complex methods. An accuracy of 77% is actually quite good given the difficulty of this problem. Many similar studies report classification accuracies in the 60%–93% range. However, it is essential to consider the trade-offs between the accuracy and energy consumption and area efficiency since achieving high accuracy levels often requires more energy-intensive processing techniques. For example, intracortical recordings may provide higher accuracy, but they are invasive and require significant amounts of energy to maintain.

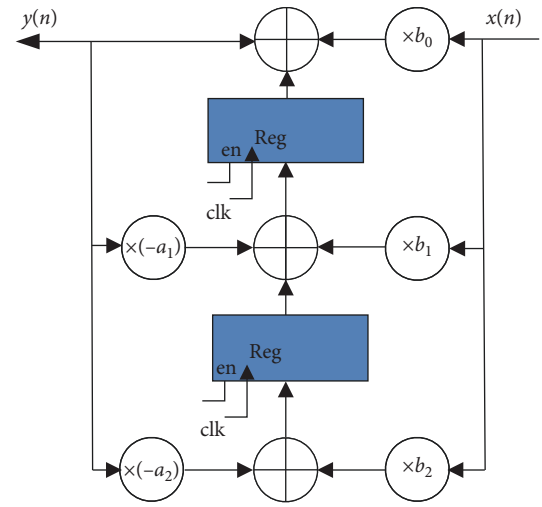

Figure 8: TDF-II IIR filter form.

Similarly, EEG-based systems are noninvasive but may be more susceptible to noise and other artifacts that reduce accuracy. The energy efficiency of a BCI system is an essential factor, especially for practical applications, such as developing portable, long-lasting, and wearable BCI devices. High energy consumption can limit the usability of a BCI system, as it may require frequent recharging or replacement of batteries, making it less practical for everyday use. Some applications, where a MI-based BCI with 77% accuracy could be sufficient, are assistive technologies for individuals with disabilities, gaming, and virtual reality control, and even neuromarketing research. High-risk or complex continuous control applications would require a higher accuracy. But for many entry-level or casual-use scenarios, this level of accuracy may be sufficient. While higher accuracy is always better, 77% can still be practical and useful for certain applications, especially given the other metrics like low cost, short training, good user experience, and low power consumption.

4. RTL Design and Implementation

This section explains details about RTL designs and improvements in the implementation of each component. Three complete BCI systems using designed components are proposed.

4.1. Preprocessing. Reduction in hardware leads us to use IIR filter instead of FIR filter due to lower order for a required bandwidth. In this study, 59 eighth-order filters are needed to filter each EEG channel. All filters are designed in Transposed-Direct-Form-II (TDF-II) IIR form; TDF-II IIR form is shown in Figure 8. Low sampling frequency allows us to propose three schemes for implementing the filter bank.

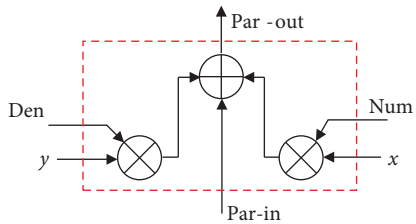

FIGURE 9: One order.

Figure 9 shows a module named "one order." This module, which contains two multipliers and a three-operand adder, is capable of calculating a single order of a filter. Later on, we used this module to build all three filters. The eighth-order IIR filter is architected as a series of eight identical first-order IIR filter modules connected in a cascaded fashion. Each module is responsible for applying a single stage of the IIR filtering process, which can be expressed as follows:

$$y_i[n] = (\text{num}_i \cdot x_i[n]) - (\text{denum}_i \cdot y_i[n-1]), \quad (1)$$

where:

(i) $y_i[n]$ is the output of the ith module at time n,
(ii) $x_i[n]$ is the input to the ith module at time n,
(iii) num_i is the numerator coefficient for the ith module,
(iv) denum_i is the denominator coefficient for the ith module, and
(v) $y_i[n-1]$ is the output of the ith module at time $n-1$.

As shown in Figure 10(a), the eighth order of a single simple form filter is built using nine "one order" modules and eight registers which are called "partial registers" to store the partial result of each filter order. Once a single filter was implemented, 59 of them were used to filter all channels separately. Due to implementing 59 completely separate filters, this approach consumes the highest silicon area among all three proposed schemes. This approach is shown in Figure 10(b).

The input $x_i[n]$ to each module is the output $y_{i-1}[n]$ from the preceding module, except for the first module where the input is the original signal $x[n]$. The output of the last module $y_8[n]$ is the final output of the eighth-order IIR filter (Figure 10(a)).

Each module's output is stored in a register to create a delay element, representing the $y_i[n-1]$ term for the next clock cycle. This register also serves as the interconnection between consecutive modules, ensuring that the output of one module is fed as the input to the next module in the sequence.

The proposed RAM-based design, using the time-shared technique, shares a single filter among all 59 EEG channels and filters each EEG channel signal one at a time. To perform this idea, a single filter should be capable of storing the history of all channels and calculating filter signals within a sampling interval. Recoding EEG channel history is implemented by replacing each and every partial register with a RAM module. An input is provided to indicate the channel number that the filter should operate on, as well as the address of RAM to obtain the history of that channel. The eighth-order IIR filter is composed of eight cascaded first-order IIR filter stages. Each stage performs the necessary filtering operation and then writes the output to a dedicated section of the RAM. This RAM acts as a buffer, storing the output of each stage before it is read by the subsequent stage in the next clock cycle. By doing so, we eliminate the need for 59 sets of registers shown in Figure 10(b) for each electrode channel, significantly reducing the hardware footprint.

The nature of BCI and EEG signals does not require a sampling rate of more than 1 kHz. This low sampling rate allows us to propose this time-shared RAM-based filter bank. By sharing the "one-order" block of a specific order among all others in filters and replacing all corresponding partial registers with RAM, the silicon area of the filter bank is reduced significantly. This reduction in the area comes from eliminating the other 58 "one order" blocks. On paper, this should reduce silicon area by about 98%. In the result section, the area of these three schemes is presented. The overall structure is shown in Figure 11.

To further reduce silicon area, a third form is presented. In this form, all of the "one-order" blocks within the time-shared filter are replaced with only one "one-order" and all RAM modules are combined together to form a single RAM that contains all of the partial results. Unlike the two previous designs, all coefficients are hardcoded in the filter, in this one, the coefficients are stored in a ROM. This ROM helps us to apply the corresponding coefficient to the "one-order" module. A control unit is implemented to control the filter and maintain its proper functionality. A module named "address gen" calculates proper addressing for state RAM based on the current EEG channel that is being filtered and the current filter order that is being calculated. Figure 12 shows a structural view of this filter. Design shown in Figure 12 leverages a counter, an address generator, a weight ROM containing the numerator (num) and denominator (denum) coefficients of the filter, and RAM to store partial sums. The counter's output corresponds to the channel number, ensuring that the filter processes the signals from each electrode in sequence. The weight ROM stores the filter coefficients, which are then fetched and applied to the single first-order IIR filter module according to the current channel number indicated by the counter. As the filter operates on the signal from each channel, the partial sums-representing intermediate results of the filtering process-are stored in specific addresses within the RAM. These addresses are dynamically generated by the address generator, which takes into account the current channel number and the stage of the filtering process. This design allows for the sequential processing of signals from all 59 channels using just one filter module. The RAM serves as a temporary storage for the intermediate results, which are then sequentially updated as the filter processes each channel's signal. By cycling through all channels and stages of the filter, the system effectively emulates an eighth-order IIR filter for each channel without the need for 59 separate filter instances. By replacing

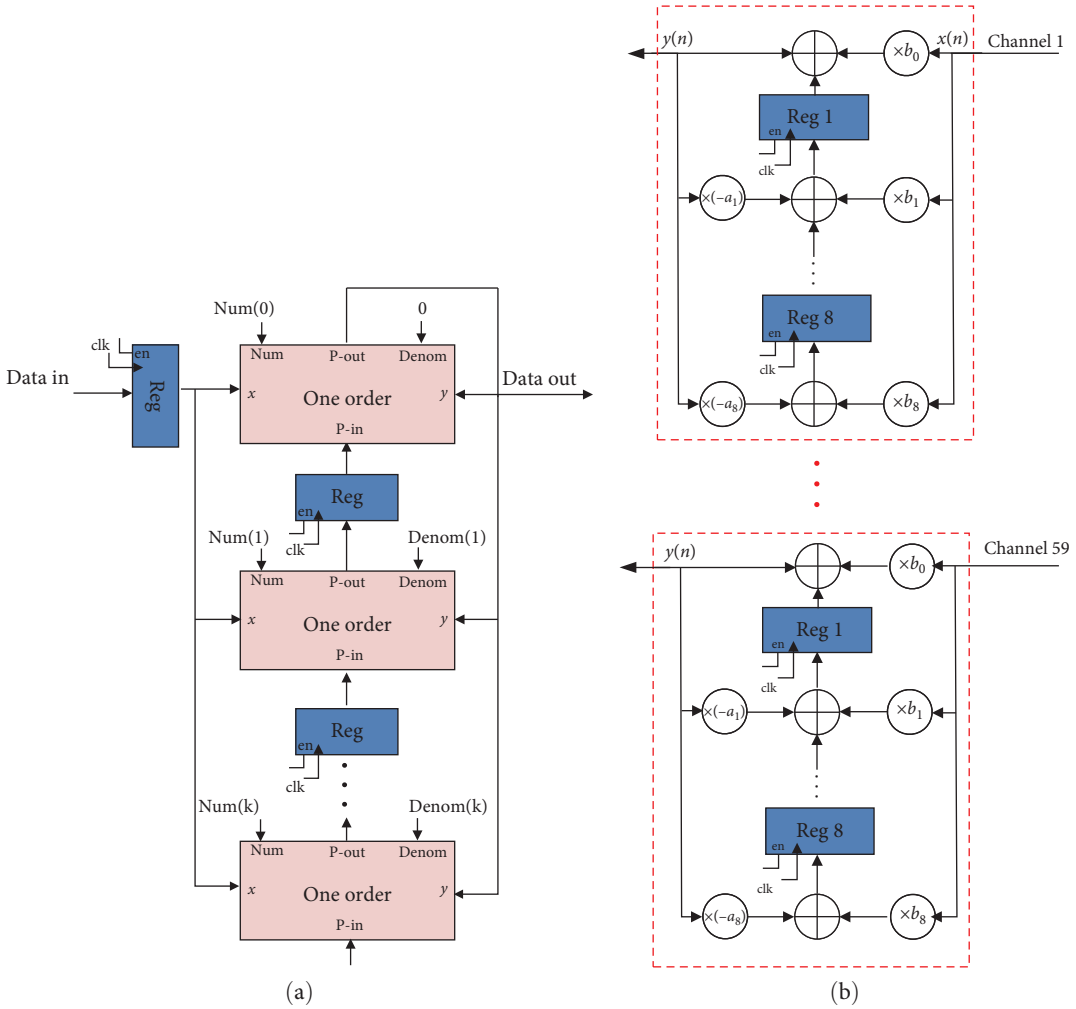

FIGURE 10: Simple form filter: (a) eighth-order IIR filter and (b) filter bank containing 59 eighth-order IIR filter (Filter I).

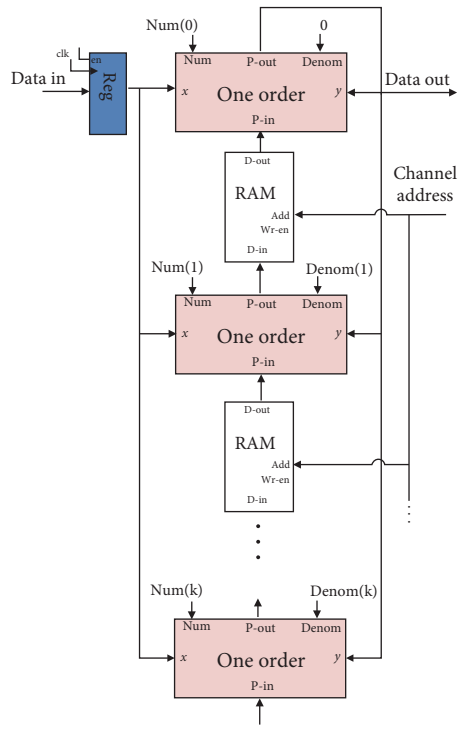

FIGURE 11: RAM-based filter bank (Filter II).

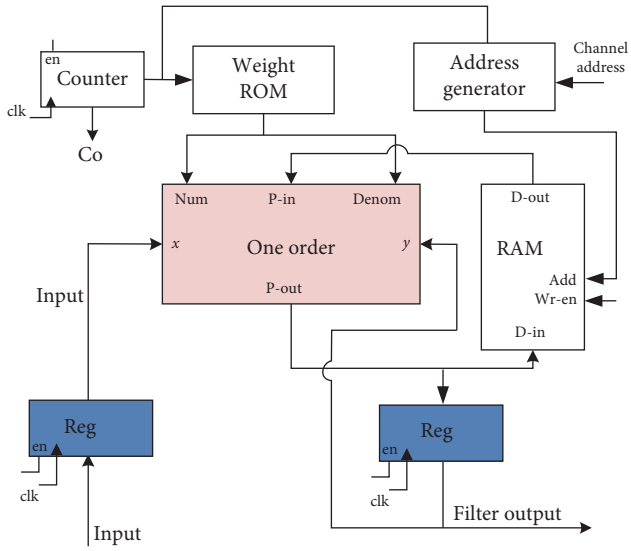

FIGURE 12: RAM-based and multiplier shared filter bank (Filter III).

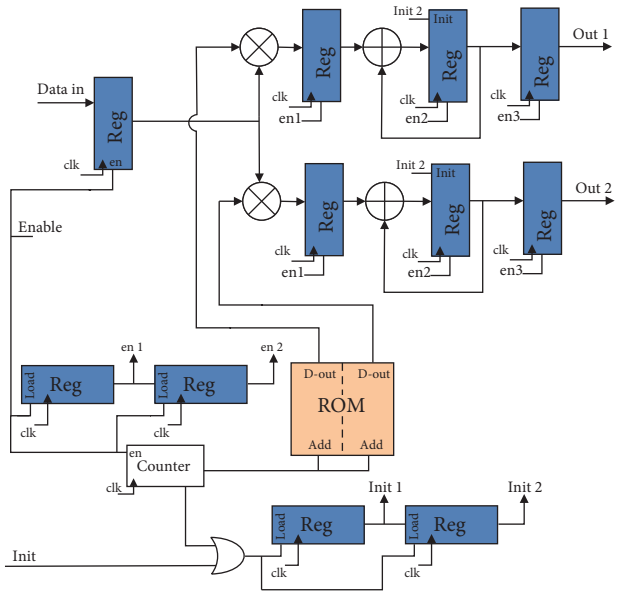

FIGURE 13: CSP block schematic.

nine "one order" modules with only one, theoretically, this filter bank should have saved nearly 99.8% and 89% silicon area compared to filter bank designs one and two, respectively.

4.2. Feature Extraction. Two separate pipelines are implemented to calculate two classes simultaneously. This decision eliminates the need of storing redundant data. Required coefficients are stored in a ROM block. By entering new data, the counter increases by one and applies a new coefficient to the MAC unit. After one data of each EEG channel is applied to the CSP unit, it automatically generates a pair of outputs and informs the next unit. The block diagram of the CSP block is shown in Figure 13. In the training phase for applying the CSP algorithm to the data in an MI BCI, the weights are first stored in a ROM. The dimensions of the weight matrix are 2×59, indicating that there are two sets of weights for each of the 59 channels. The CSP module produces two outputs, which are generated in parallel. This is achieved by processing the input from each channel in turn. As each channel's data is inputted into the system, a counter retrieves the corresponding weight from the ROM and multiplies it with the input. The result of this multiplication is then accumulated with the previous values. This operation is controlled by a signal named init2. After the data from all 59 channels have been processed, which takes 59 clock cycles, a signal named en3 is activated. Once en3 is active, the output is ready and can be passed on to the next stage of processing. The output of the counter is a number corresponding to the channel number. The relevant coefficients are transferred from the weight ROM to Module 1. Based on the output of the counter, the previous sum and the subsequent sum are stored in the corresponding memory address, which is generated by the address generator. To elaborate, as the data from each channel are processed by Module 1, the counter indicates the channel number, and the corresponding weights are fetched from the weight ROM. These weights are then used to multiply the input signal for that specific channel. The product of this multiplication is added to the cumulative sum of the previous products. The address generator plays a crucial role here. It generates the specific memory addresses where the sums (both the previous and the subsequent ones) are to be stored. This ensures that the data are organized correctly in memory, facilitating efficient retrieval, and processing in subsequent stages.

In this paper, we present a robust method for the real-time calculation of variance and average within a fixed-size sliding window of samples. This technique is particularly advantageous for BCI applications where the sampling rate is relatively low, such as in the case of electroencephalogram (EEG) signal processing. The proposed method ensures computational efficiency and accuracy, enabling the variance to be updated incrementally with each new sample, thereby eliminating the need for recalculations from the entire dataset. The methodology comprises the following steps:

(1) Initial conditions: We start with a window of 400 samples, each with an initial variance denoted as V_I and an initial average denoted as M_I.

(2) Updating for new sample: When a new sample, Data_{new}, enters the window and an old sample, Data_{old}, exits, we update the sum and sum of squares to reflect this change.

(3) Sum of samples: The sum of samples is updated by adding the new sample and subtracting the old sample, ensuring that we are always considering the current set of 400 samples:

$$\text{Updated} \sum_{k=1}^{400} x_k = (400 \times M_I) + \text{Data}_{\text{new}} - \text{Data}_{\text{old}}. \tag{2}$$

(4) Sum of squares of samples: The sum of the squares of samples is updated in a similar fashion:

$$\text{Updated} \sum_{k=1}^{400} x_k^2 = (400 \times (V_I + M_I^2)) + \text{Data}_{\text{new}}^2 - \text{Data}_{\text{old}}^2. \tag{3}$$

(5) New average calculation: The new average, M_f, is then calculated using the updated sum of samples:

$$M_f = \frac{1}{400} \times \text{Updated sum of samples} = \frac{1}{400} \sum_{k=1}^{400} x_k. \tag{4}$$

This calculation gives us the true average of the current window, not a partial average.

(6) New variance calculation: The new variance, V_f, is calculated using the updated sum and sum of squares:

$$V_f = \frac{1}{400} \sum_{k=1}^{400} x_k^2 - \left(\frac{1}{400} \sum_{k=1}^{400} x_k \right)^2. \tag{5}$$

This formula correctly computes the variance based on the true average of the current window.

(7) Final variance formula: The final variance formula provided in the paper is derived from the previous steps and accurately calculates the new variance based on the true average of the current window of samples:

$$V_f = V_I + M_I^2 + \left(\frac{\text{Data}_{\text{new}}^2 - \text{Data}_{\text{old}}^2}{400} \right) - M_f^2. \tag{6}$$

The sequence of operations detailed above facilitates a continuous and real-time update of the variance, which is critical for adaptive filtering and signal analysis within BCI systems. The efficiency of this method lies in its ability to maintain accurate variance calculations with minimal computational overhead, thereby making it suitable for applications with stringent resource constraints.

Implementation results show that in comparison to the main formula, the recursive variance formula leads to reduction of energy consumption by 96%, respectively.

The RTL design for the variance calculation module, as illustrated in Figure 14, is optimized to perform the computation within eight clock cycles. The output from the CSP module, once available, is sequentially fed into the variance module. For the sake of clarity, only one of the dual variance modules operating in tandem is shown in the figure. The computation process initiates in the first clock cycle by enabling the load registers for the new value (nv) and the old value (ov), alongside the push and pop commands of the first-in-first-out (FIFO) queue. This setup permits the accommodation of the incoming new data into nv and the displacement of the oldest data into ov. Concurrently, the register m4 is loaded with the product of the prior mean multiplied by 400, while the multiplexer mx2 is engaged to capture the preceding mean for subsequent

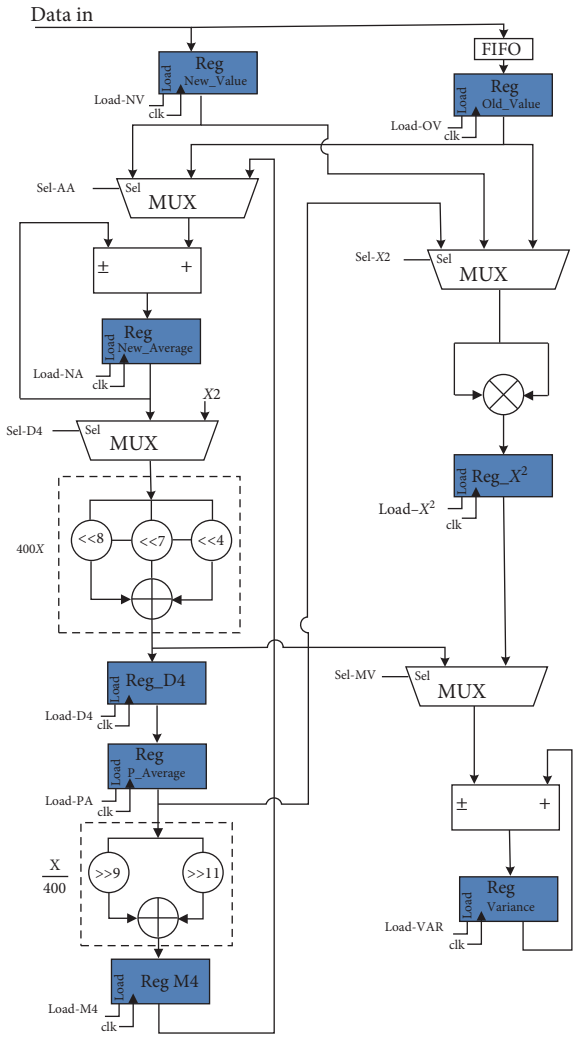

FIGURE 14: Variance block schematic.

operations. Subsequently, in the second clock cycle, the multiplexer mvv selects the content of register x2, triggering the load registers for both the variance and the cumulative sum. The variance is updated by incorporating the square of the prior mean. Concurrently, the multiplexer maa retrieves the value from register m4, which is then loaded into the register designated for the new mean. As the third clock cycle commences, the multiplexer maa selects the new data residing in nv, adding it to the existing sum in ma, which is then stored back in ma. In parallel, the multiplexer mx2 captures the new data value, directing it to register x2 to compute and store its square. In the fourth clock cycle, the multiplexer maa is set to select the value of the outgoing data, and through the adder/subtracter, the old data value is subtracted from the cumulative sum in ma. Meanwhile, the multiplexer md4 selects the value from register x2, and the load register d4 is engaged. This action results in the storage of the new squared value divided by 400 in d4, while simultaneously computing the square of the old value divided by 400. During the fifth clock cycle, the multiplexer mvv selects the value in d4, and with the activation of the variance register load and the add operation, the variance is augmented by the

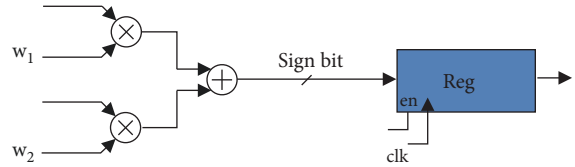

Figure 15: Block diagram of SVM.

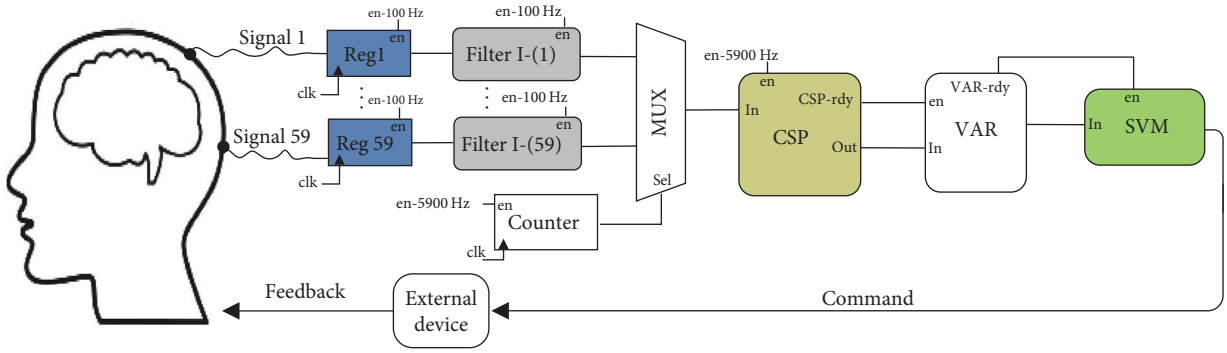

Figure 16: Block diagram of design I.

value in d4. This step also involves storing the new mean value in register d4. In the sixth clock cycle, the value in d4, now representing the new mean, is preserved in the pa register upon activation. Concurrently, the multiplexer md4 selects the value from register x2. The seventh clock cycle sees the multiplexer mx2 selecting the value in the pa register, preparing to load the new mean squared into register x2 in the following cycle. At the same time, the multiplexer mvv selects the value in d4, and with the subtraction operation from ba and the variance register load, the variance is adjusted by subtracting the square of the old value divided by 400. Finally, in the eighth clock cycle, the new mean squared, held in register x2, is selected by the multiplexer mvv. The subtract operation from va is then activated, and the variance register is loaded, culminating in the storage of the final variance value in the variance register during the subsequent clock cycle. The "variance output ready" signal is then dispatched to the subsequent processing block.

4.3. Classification. As mentioned before, the linear SVM classifier is used for classification. Its linearity causes a significant reduction in hardware. The block diagram of this module is shown in Figure 15. The critical path consists of a multiplier and an adder. This module should calculate the output data with a higher frequency than the sampling rate.

4.4. Proposed BCI Systems. All mentioned blocks make up the proposed BCI system. The proposed system begins to process after receiving digitalized signals and makes a decision after filtering data and extracting features. In the following, the proposed systems are investigated in detail.

In a simple form system shown in Figure 16, one separate filter has been considered for each channel. This filter, which works with the sampling rate, prepares its output after one clock. The filtered data are applied to CSP to reduce feature size to two features. This block receives 59 data from channels at 59 consecutive clocks, then, prepares two outputs and activates the variance module.

The variance block receives input data by receiving the start signal and calculates the variance of new data and 399 prior data. This process takes 18 clocks. After preparing the output, the variance block sends its output to the SVM block which classifies data at one clock. A timer block is considered to generate activation signals of the filter and CSP blocks.

The designed filter named Filter II is used in the second proposed system, which is shown in Figure 17. The second proposed system differs from the filter block's first one. In the way that just one filter has been used for all channels. Filter II should be activated at a rate of 59 times the sampling rate to calculate the filtered data of all channels without losing data. The filter block informs CSP by preparing filtered data. The rest of the second system works similarly to the first one. The built-in timer unit is responsible for generating filter activation signals.

The third proposed system, shown in Figure 17, uses Filter III, which makes it different from other systems. In this way, 10 clocks are needed to produce the output of each channel. Other units and connections of this system are like the second system.

In all three structures, digital clock manager module is used to set different frequencies of different parts of the circuit. The task of this module is to receive the working frequency of the circuit (in this research, 1 MHz was chosen) and make activation signals of different modules using the working frequency. The design of this module is such that different frequencies can be applied as input. The system does not operate at a fixed, predetermined frequency. In this research, the lowest frequency at which the circuit works is 5,900 Hz, and the highest frequency of the filter is 130 MHz, so any frequency between these two numbers can be applied to the circuit as a working frequency. Therefore, the system

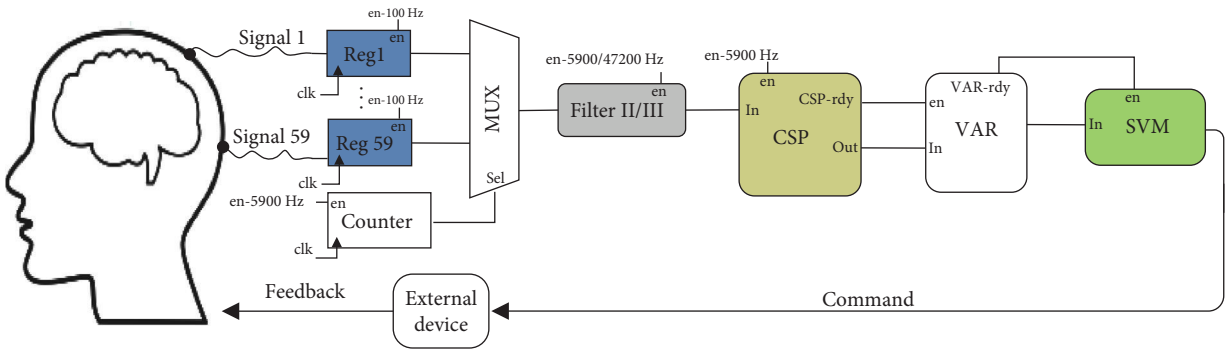

Figure 17: Block diagram of design II/III.

Table 3: Latency of modules and proposed systems.

Modules and systems	Latency
Filter I	1 clk
Filter II	$1 \times N$ clk
Filter III	$(2 + \text{order}) \times N$ clk
CSP	$3 \times N$ clk
Variance	10 clk
SVM	1 clk
Prop. design I	$12 + 3 \times N$ clk
Prop. design II	$4 \times N + 11$ clk
Prop. design III	$(2 + \text{order}) \times N + 3 \times N + 11$ clk

has the ability to create a compromise between response delay and power consumption by changing the frequency. To reduce the response delay, the frequency can be increased, but it should be noted that with increasing frequency, power consumption increases.

5. Hardware Simulation

Table 3 shows the calculation of response delay for each module and system. According to the reported frequencies after synthesis, to compare the minimum period, it is necessary to pay attention to the maximum frequency of each module of the system.

The accuracy and power consumption are the two key parameters that show the success degree for the performance of an embedded BCI system. Belwafi et al. [2] categorized the amount of accuracy and power consumption into three levels and estimated that a system with more than 75% accuracy and less than 1 W power consumption performs as a successful one. In addition, it is necessary to reduce the latency for real-time and wearable systems. Table 4 compares the performance of the three proposed systems with the previous works. Our proposed designs achieve 77% accuracy, which is almost similar to [8, 11, 40] and is considered a successful system according to Belwafi et al. [2]. Although Belwafi et al. [7] and Feng et al. [39] offered the highest accuracy among the compared designs, their system does not satisfy the power consumption and timing constraints. Compared to Belwafi et al.'s [8, 11, 40] study, our system is, respectively, 3%, 5%, and 6% less accurate, which is not a significant amount to consider. While Belwafi et al. [8] and Malekmohammadi et al. [11] reduced their power consumption to less than 1 W, which is considered a reasonable power consumption, their latency is 430 and 83 ms, respectively, which is far more than the latency of our system. Our best design achieves similar power consumption to [40] with four times less latency as well as without reduction in the number of channels, which makes it suitable for wearable and real-time BCI applications.

As for the hardware resources used on the FPGA, Table 5 compares the number of LUT, DSP, and registers used in the three proposed systems and two previous ones. The lowest LUT consumption is associated with the system proposed in Malekmohammadi et al.'s [11] study because of using a large number of DSPs (reported 24), while the best-proposed design employs only 10 DSPs. Among the three proposed systems, the effects of the developed techniques are visible. Design III has 95% and 45% reduction in LUT usage compared to Design I and Design II, respectively. According to the Vivado report, the number of registers used in Design I and Design II remains unchanged since only partial registers are replaced with RAM. Using internal RAMs of FPGA in design III leads to reduction in the number of registers. The reason behind the reduction in the number of registers in this work compared to the other works is the technique used in the variance calculation.

According to the results of the area consumption on FPGA in the proposed designs, the filter in design I occupies the largest area of the chip. By applying the first and second techniques in design II and design III, the area decreases 90% and 99%, respectively. Since the implementation of the rest of the elements is the same, there was an insignificant change in the occupied area.

Most of the embedded BCIs are implemented on FPGAs or microcontrollers. The best solution to minimize the size of the system besides achieving the lowest power consumption is exporting the system as an ASIC. Therefore, all the proposed designs are implemented using 45 nm CMOS Technology. It can be observed in Table 6 that designs II and III are 37.5 times more power efficient than design I. Also, the consumed area of design III has decreased by 66% compared to design I. Therefore, the reported methods make design III suitable for low-power wearable devices. it takes 0.63 ms to response after one trial (400 data) and consumes 0.00 4 W power, so our best design consumes 2 μj energy.

Table 4: Comparison of different parameters of proposed designs and previous designs.

	Data	Sampling rate (Hz)	Algorithms	Platform	Accuracy	Power (W)	Latency (ms)
[7]	MI, SSEVP, P300	N/I	Adaptive filter, CSP, MD	Stratix-IV	94.47	1.067	394
[8]	MI	256	WOLA filter bank, CSP, MD	Stratix-IV	80.2	0.67	430
			Surface laplacian				
[11]	MI	128	SCSSP, MI, LDA, SVM	Virtex-6 FPGA	80.55	0.09	83
[39]	SSEVP	200	FIR filter, averaging method	Cyclone II EP2C35 DSP	90.62	27	2,000
[40]	MI	100	CNN	ARM Cortex-M4 and M7	82.51	0.01	2.95
Prop. design I	MI	100	IIR, CSP, VAR, SVM	Virtex-7 xcvu-65 FPGA	77.45	0.026	0.13
Prop. design II	MI	100	IIR, CSP, VAR, SVM	Virtex-7 xcvu-65 FPGA	77.45	0.011	0.19
Prop. design III	MI	100	IIR, CSP, VAR, SVM	Virtex-7 xcvu-65 FPGA	77.45	0.011	0.63

Table 5: Comparison of hardware resource usage.

	LUT	DSP	Flip flop
Prop. design I	157,556	6	28,570
Prop. design II	13,268	6	28,502
Prop. design III	6,900	10	14,158
[8]	12,025	8	16,556
[11]	2,663	24	2,988

Table 6: Comparison of ASIC result.

	Power (W)	Area (mm^2)	Delay (ns)
Prop. design I	0.15	0.753124	2.37
Prop. design II	0.004	0.275067	2.27
Prop. design III	0.004	0.250889	2.27

6. Conclusion

In this paper, we proposed efficient hardware for an embedded MI-BCI system. First, we designed a BCI system to classify two MI tasks by comparing various algorithms for preprocessing, feature extraction and classification. The best algorithms were IIR filter for preprocessing, CSP for feature extraction, and SVM as a classifier in terms of hardware cost and reasonable accuracy (77.4%). To design components in RTL, we used methods to reduce the area, power consumption, and latency without losing accuracy. In preprocessing, EEG signals from each channel should be applied to an IIR filter, which is area- and power-consuming, so we used time-shared and RAM-based techniques to reduce the power consumption and the area of filter bank. In designing other components, we used pipelining and hardware-friendly equations to meet the constraints. We implemented our designs on a Virtex-7 FPGA as a prototyping platform achieving 0.01 W power consumption and 1.52% utilization. Finally, to achieve wearable embedded BCI constraints and minimize the area, we implemented our designs using 45 nm CMOS technology which consumed 0.004 W power with 0.25 mm^2 area.

Data Availability

No underlying data were collected or produced in this study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] A. Vinod and K. P. Thomas, "Neurofeedback games using eeg-based brain–computer interface technology," in *Signal Processing and Machine Learning for Brain–Machine Interfaces*, vol. 114 of *IET Control, Robotics and Sensors Series*, pp. 301–329, The Institution of Engineering and Technology, London, United Kingdom, 2018.

[2] K. Belwafi, S. Gannouni, and H. Aboalsamh, "Embedded brain computer interface: state-of-the-art in research," *Sensors*, vol. 21, no. 13, Article ID 4293, 2021.

[3] C. T. Lin, F. C. Lin, S. A. Chen, S. W. Lu, T. C. Chen, and L. W. Ko, "Eeg-based brain–computer interface for smart living environmental auto-adjustment," *Journal of Medical and Biological Engineering*, vol. 30, no. 4, pp. 237–245, 2010.

[4] K. Khurana, P. Gupta, R. C. Panicker, and A. Kumar, "Development of an FPGA-based real-time p300 speller," in *22nd International Conference on Field Programmable Logic and Applications (FPL)*, pp. 551–554, IEEE, 2012.

[5] L. Jiang, E. Tham, M. Yeo, and O. G. Phu, "iPhone-based portable brain control wheelchair," in *2012 7th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, pp. 1592–1594, IEEE, Singapore, 2012.

[6] G. Li and W.-Y. Chung, "Combined EEG-Gyroscope-tDCS brain machine interface system for early management of driver drowsiness," *IEEE Transactions on Human-Machine Systems*, vol. 48, no. 1, pp. 50–62, 2018.

[7] K. Belwafi, F. Ghaffari, R. Djemal, and O. Romain, "A hardware/software prototype of EEG-based BCI system for home device control," *Journal of Signal Processing Systems*, vol. 89, no. 2, pp. 263–279, 2017.

[8] K. Belwafi, O. Romain, S. Gannouni, F. Ghaffari, R. Djemal, and B. Ouni, "An embedded implementation based on adaptive filter bank for brain–computer interface systems," *Journal of Neuroscience Methods*, vol. 305, pp. 1–16, 2018.

[9] A. Palumbo, F. Amato, B. Calabrese et al., "An embedded system for EEG acquisition and processing for brain computer interface applications," in *Wearable and Autonomous Biomedical Devices and Systems for Smart Environment*, vol. 75 of *Lecture Notes in Electrical Engineering*, pp. 137–154, Springer, Berlin, Heidelberg, 2010.

[10] R. Karkon, S. M. R. Shahshahani, and H. R. Mahdiani, "A Custom Hardware CCA Engine for Real-time SSVEP-based BCI Applications," in *2020 20th International Symposium on Computer Architecture and Digital Systems (CADS)*, pp. 1–6, IEEE, Rasht, Iran, 2020.

[11] A. Malekmohammadi, H. Mohammadzade, A. Chamanzar, M. Shabany, and B. Ghojogh, "An efficient hardware implementation for a motor imagery brain computer interface system," *Scientia Iranica*, pp. 26–94, 2019.

[12] N. Padfield, J. Zabalza, H. Zhao, V. Masero, and J. Ren, "EEG-based brain–computer interfaces using motor-imagery: techniques and challenges," *Sensors*, vol. 19, no. 6, Article ID 1423, 2019.

[13] M. A. L. Nicolelis, "Brain–machine interfaces to restore motor function and probe neural circuits," *Nature Reviews Neuroscience*, vol. 4, no. 5, pp. 417–422, 2003.

[14] R. Aler, I. M. Galván, and J. M. Valls, "Applying evolution strategies to preprocessing EEG signals for brain–computer interfaces," *Information Sciences*, vol. 215, pp. 53–66, 2012.

[15] J. P. Donoghue, "Connecting cortex to machines: recent advances in brain interfaces," *Nature Neuroscience*, vol. 5, no. S11, pp. 1085–1088, 2002.

[16] A. Widmann, E. Schröger, and B. Maess, "Digital filter design for electrophysiological data—a practical approach," *Journal of Neuroscience Methods*, vol. 250, pp. 34–46, 2015.

[17] W. Yi, S. Qiu, H. Qi, L. Zhang, B. Wan, and D. Ming, "EEG feature comparison and classification of simple and compound limb motor imagery," *Journal of NeuroEngineering and Rehabilitation*, vol. 10, no. 1, pp. 1–12, 2013.

[18] R. Aler, I. M. Galván, and J. M. Valls, "Evolving spatial and frequency selection filters for brain–computer interfaces," in *IEEE Congress on Evolutionary Computation*, pp. 1–7, IEEE, Barcelona, Spain, 2010.

[19] C. Y. Chen, C. W. Wu, C. T. Lin, and S. A. Chen, "A novel classification method for motor imagery based on brain–computer

interface," in *2014 International Joint Conference on Neural Networks (IJCNN)*, pp. 4099–4102, IEEE, Beijing, China, 2014.

[20] C. Lindig-Leon and L. Bougrain, "A multi-label classification method for detection of combined motor imageries," in *2015 IEEE International Conference on Systems, Man, and Cybernetics*, pp. 3128–3133, IEEE, 2015.

[21] Y. Chang, "Architecture design for performing grasp-and-lift tasks in brain–machine-interface-based human-in-the-loop robotic system," *IET Cyber-Physical Systems: Theory & Applications*, vol. 4, no. 3, pp. 198–203, 2019.

[22] H. Yang, S. Sakhavi, K. K. Ang, and C. Guan, "On the use of convolutional neural networks and augmented CSP features for multi-class motor imagery of EEG signals classification," in *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pp. 2620–2623, IEEE, Milan, Italy, 2015.

[23] S. Aggarwal and N. Chugh, "Signal processing techniques for motor imagery brain computer interface: a review," *Array*, vol. 1-2, Article ID 100003, 2019.

[24] L. Q. Thang and C. Temiyasathit, "Increase performance of four-class classification for motor-imagery based brain–computer interface," in *2014 International Conference on Computer, Information and Telecommunication Systems (CITS)*, pp. 1–5, IEEE, Jeju, Korea (South), 2014.

[25] H. Wang and A. Bezerianos, "Brain-controlled wheelchair controlled by sustained and brief motor imagery BCIs," *Electronics Letters*, vol. 53, no. 17, pp. 1178–1180, 2017.

[26] H. K. Lee and Y. S. Choi, "A convolution neural networks scheme for classification of motor imagery EEG based on wavelet time-frequecy image," in *2018 International Conference on Information Networking (ICOIN)*, pp. 906–909, IEEE, Chiang Mai, Thailand, 2018.

[27] J. Zhang, C. Yan, and X. Gong, "Deep convolutional neural network for decoding motor imagery based brain computer interface," in *2017 IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC)*, pp. 1–5, IEEE, Xiamen, China, 2017.

[28] S. Sakhavi, C. Guan, and S. Yan, "Parallel convolutional-linear neural network for motor imagery classification," in *2015 23rd European Signal Processing Conference (EUSIPCO)*, pp. 2736–2740, IEEE, 2015.

[29] W. Ko, J. Yoon, E. Kang, E. Jun, J.-S. Choi, and H.-I. Suk, "Deep recurrent spatio-temporal neural network for motor imagery based BCI," in *2018 6th International Conference on Brain–Computer Interface (BCI)*, pp. 1–3, IEEE, Gangwon, Korea (South), 2018.

[30] B. Blankertz, G. Dornhege, M. Krauledat, K.-R. Müller, and G. Curio, "The non-invasive berlin brain–computer interface: fast acquisition of effective performance in untrained subjects," *NeuroImage*, vol. 37, no. 2, pp. 539–550, 2007.

[31] A. K. Das and S. Suresh, "An effect-size based channel selection algorithm for mental task classification in brain computer interface," in *2015 IEEE International Conference on Systems, Man, and Cybernetics*, pp. 3140–3145, IEEE, 2015.

[32] A. Jiang, J. Shang, X. Liu, Y. Tang, H. K. Kwan, and Y. Zhu, "Efficient CSP algorithm with spatio-temporal filtering for motor imagery classification," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 28, no. 4, pp. 1006–1016, 2020.

[33] J. Wang, Z. Feng, X. Ren, N. Lu, J. Luo, and L. Sun, "Feature subset and time segment selection for the classification of EEG data based motor imagery," *Biomedical Signal Processing and Control*, vol. 61, Article ID 102026, 2020.

[34] J.-S. Bang, M.-H. Lee, S. Fazli, C. Guan, and S.-W. Lee, "Spatio-spectral feature representation for motor imagery classification using convolutional neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 7, pp. 3038–3049, 2022.

[35] R. T. Schirrmeister, J. T. Springenberg, L. D. J. Fiederer et al., "Deep learning with convolutional neural networks for EEG decoding and visualization," *Human Brain Mapping*, vol. 38, no. 11, pp. 5391–5420, 2017.

[36] K. Zhang, G. Xu, Z. Han et al., "Data augmentation for motor imagery signal classification based on a hybrid neural network," *Sensors*, vol. 20, no. 16, Article ID 4485, 2020.

[37] J. Yang, Z. Ma, J. Wang, and Y. Fu, "A novel deep learning scheme for motor imagery EEG decoding based on spatial representation fusion," *IEEE Access*, vol. 8, pp. 202100–202110, 2020.

[38] S. Kumar, A. Sharma, and T. Tsunoda, "Brain wave classification using long short-term memory network based OPTICAL predictor," *Scientific Reports*, vol. 9, no. 1, Article ID 9153, 2019.

[39] Z. Feng, L. Zeng, H. Wu, F. Tian, and Q. He, "Design of an online brain-computer interface system based on field programmable gate array," *Journal of Physics: Conference Series*, vol. 1624, Article ID 042061, 2020.

[40] X. Wang, M. Hersche, M. Magno, and L. Benini, "MI-BMInet: an efficient convolutional neural network for motor imagery brain–machine interfaces with EEG channel selection," *IEEE Sensors Journal*, 2022.