An Efficient Model-Compressed EEGNet Accelerator for Generalized Brain-Computer Interfaces with Near Sensor Intelligence

Lichen Feng, Hongwei Shan, Yueqi Zhang, and Zhangming Zhu

***Abstract*—Brain-computer interfaces (BCIs) is promising in interacting with machines through electroencephalogram (EEG) signal. The compact end-to-end neural network model for generalized BCIs, EEGNet, has been implemented in hardware to get near sensor intelligence, but without enough efficiency. To utilize EEGNet in low-power wearable device for long-term use, this paper proposes an efficient EEGNet inference accelerator. Firstly, the EEGNet model is compressed by embedded channel selection, normalization merging, and product quantization. The customized accelerator based on the compressed model is then designed. The multilayer convolutions are achieved by reusing multiplying-accumulators and processing elements (PEs) to minimize area of logic circuits, and the weights and intermediate results are quantized to minimize memory sizes. The PEs are clock-gated to save power. Experimental results in FPGA on three datasets show the good generalizing ability of the proposed design across three BCI diagrams, which only consumes 3.31% area and 1.35% power compared to the one-to-one parallel design. The speedup factors of 1.4, 3.5, and 3.7 are achieved by embedded channel selection with negligible loss of accuracy (-0.80%). The presented accelerator is also synthesized in 65nm CMOS low power (LP) process and consumes 0.23M gates, 24.4ms/inference, 0.267mJ/inference, which is 87.22% more efficient than the implementation of EEGNet in a RISC-V MCU realized in 40nm CMOS LP process in terms of area, and 20.77% more efficient in terms of energy efficiency on BCIC-IV-2a dataset.**

***Index Terms*—Accelerator, brain-computer interfaces, channel selection, near sensor intelligence, product quantization.**

I. Introduction

BRAIN-computer interface (BCI) [1], providing users the communication and control capabilities towards machine through brain, is attracting researchers to explore benefits from it. For example, BCIs designed for seizure prediction can offer alarms for epilepsy patients to improve the quality of life [2]. For individuals with severe physical disabilities, BCIs can be used to control prosthetic limbs [3] or wheelchairs [4] in order to restore movement.

This work was supported in part by the National Natural Science Foundation of China under Grant 62104175, Grant 62021004, and in part by the Fundamental Research Funds for the Central Universities under Grant XJS211111. (Corresponding author: Zhangming Zhu)

The authors are with the Shaanxi Key Laboratory of Integrated Circuits and Systems, School of Microelectronics, Xidian University, Xi'an 710071, China (e-mail: zmyh@263.net).

In addition to the clinical scenarios, recent progresses in wearable platform [5] and low-cost non-invasive electrodes in consumer grade [6] that sample multichannel scalp electroencephalogram (EEG) signal, are pushing BCI paradigms towards enhancing performance on daily life tasks for healthy users [7], such as emotion recognition [8], drowsiness detection [9], motor imagery (MI) [10] and steady-state visual evoked potential (SSVEP) recognition [11] for controlling robots [12, 13], etc. Before building a BCI as one of these, however, a priori knowledge about the expected EEG signal and complex techniques are generally required to extract effective feature, such as Riemannian geometry features [14] and filter-bank common spatial patterns (FBCSP) [15], which complicates the deployment of BCIs.

The recent success of deep learning for neural networks [16] has largely alleviated the need for manual feature extraction. Using paradigm-specific insights in designing the structure, various deep convolutional neural network (CNN) models have been proposed to execute end-to-end learning from raw EEG signal for different BCI applications [17]-[20]. Utilizing the depthwise and separable convolutions, the compact CNN architecture EEGNet [21] with less than 3000 parameters while keeping accurate across paradigms is proposed, but millions of multiply accumulate operations are still required. Due to the lack of enough computational efficiency near electrodes, most traditional BCI systems have to send the sampled EEG signal to a computer or a cloud in a wired or wireless way for the complex offline processing, which leads to issues of processing delay, user comfort, battery life, and privacy concerns for practical application to commercial users [22]. To avoid these issues, efficient implementation of EEGNet for signal processing right at the recording devices to obtain near sensor intelligence is becoming necessary.

A one-to-one hardware implementation of EEGNet has been proposed [23] but still consumes huge amount of resource, which has little possibility to be wearable. Wang et al. [24] presents the scaling down of EEGNet which takes 101ms and 4.28mJ per inference on ARM Cortex-M4F Microcontroller unit (MCU). Schneider et al. [25] have developed the quantized EEGNet (Q-EEGNet), and its optimized mapping on the parallel ultra-low power platform, Mr. Wolf [26], achieving an accuracy of 70.9% on the popular BCI Competition IV-2a dataset [27] and an energy consumption of 0.627mJ per inference with the delay of 5.82ms at the clock frequency of

350MHz. However, Q-EEGNet still has some inefficiencies. On the one hand, it is only verified on MI task, and 4.5 seconds EEG epoch length is adopted ([-0.5, 4] seconds post cue onset) to obtain high accuracy, which will generate long delay (>4 seconds) for the system to offer a judgement and is therefore not practical in use. A commonly used period of 2 seconds ([0.5, 2.5] seconds post cue onset [21, 28, 29]) for the MI task, and datasets for different BCI paradigms remains to be verified after all the optimizations. On the other hand, Q-EEGNet is a software work implemented on the general-purpose platform in essence, the redundant operations such as instruction fetching and decoding are unavoidable.

According to the above discussion, in this paper, a customized integrated-circuit design of EEGNet inference accelerator is proposed for more efficient generalized BCI with near sensor intelligence. The main contributions of this work are summarized as follows.

- The computation complexity of EEGNet is analyzed and reduced through embedded channel selection. The speedup factors of 1.4, 3.5, and 3.7 are achieved on three BCI datasets with negligible loss of accuracies.
- The two normalization layers are merged, and the vector production in the fully-connected (FC) layer is quantized using a hash method, in order to reduce area and energy consumption.
- The presented design reuses the processing elements (PEs) to save area and retain the generalizability across BCI paradigms, and gates clocks of the PEs to save power. The weights and intermediate results are also quantized to minimize memory sizes.
- The proposed accelerator uses merely 3.31% area and 1.35% power compared to the one-to-one parallel design, and 12.78% gate counts and 79.23% energy per inference compared to the implementation of Q-EEGNet in Mr. Wolf MCU.

The remainder of this paper is organized as follows. Section II reviews the related hardware implementations of BCI paradigms. Section III presents the model compression method including embedded channel selection and product quantization for EEGNet. Section IV describes the circuit design of the proposed EEGNet architecture based on the compressed model. Section V shows the implementation results and comparisons with previous works. At last, the conclusion is draw in Section VI.

II. Related Works

Many researches have been proposed to implement various algorithms in hardware in order to improve the efficiency and accuracy, and lower the latency of the BCI systems with near sensor intelligence for healthy users. A summary of them is shown in Table I.

For the BCI paradigm of MI, the method of common spatial pattern (CSP) has been realized in field programmable gate arrays (FPGAs) [30, 31]. The average power consumption of 0.7W with an execution runtime of 430ms is consumed to achieve the accuracy of 78.85% on a 2-class task in [30]. The 2-class and 4-class classification accuracy scores are 80.55% and 67.21%, respectively, with an energy consumption of 0.978mJ per inference [31]. Wang et al. [32] optimize the implementation of the multispectral Riemannian classifier (MRC) in Mr. Wolf [26] in terms of resource usage by using only one temporal window and quantizing the features to mixed-precision representations, which achieves an energy consumption of 1.304mJ and a runtime of 33.39ms per inference. They further improve the classification accuracy of MRC and implement it on a more energy-efficient MCU, Vega [33], obtaining the performance of 198μJ and 16.9ms per classification [34]. An efficient MI-BMInet is further proposed by Wang et al. [35] and implemented in Mr. Wolf. which consumes only 50.10μJ with an inference delay of 5.53ms.

TABLE I
Summary of Related Works

Ref.	BCI Paradigm	Method	Plat-form	Acc-uracy	Delay (ms)	Energy (mJ/Inf)
[23]	ERP[1] ERN[2]	EEGNet	FPGA	88.75% 76.33%	0.058 0.115	N.A.[3]
[24]	MI	EEGNet	MCU	65.07%	44	18.1
[25]	MI	Q-EEGNet	MCU	70.9%	5.82	0.627
[30]	MI	CSP	FPGA	76.8%	430	301
[31]	MI	CSP	FPGA	67.2%	10	0.83
[32]	MI	MRC	MCU	75.1%	33.39	1.304
[34]	MI	MRC	MCU	76.4%	16.9	0.198
[35]	MI	MI-BMINet	MCU	76.03%	5.53	0.05
[36]	SSVEP	Phase Encode	FPGA	89.29%	24.67 bpm[4]	N.A.
[37]	SSVEP	Spectral Feature Select	FPGA	91%	47.93 bpm	N.A.

1: Event-Related Potential
2: Error-Related Negativity
3: N.A.: Not Available
4: bpm: bits per minute

For the paradigm of SSVEP, a design based on a low-cost FPGA has been proposed for visual control of four panels very early [36]. Recently, a fully wearable BCI system based on Artix-7 FPGA [37] is proposed, but both the two works lack of resource consumption information.

In terms of the EEGNet model that could generalize across different BCI paradigms, the customized one-to-one FPGA implementation of EEGNet has been proposed [23], which executes an inference within 1ms, but consumes huge amount of resource. Wang et al. [24] reduce the computation complexity by sub-sampling, channel reduction, and narrowing the time window, and the smallest EEGNet takes 101ms and 4.28mJ per inference on ARM Cortex-M4F MCU. Schneider et al. [25] have developed the Q-EEGNet, and its optimized mapping on the ultra-low power MCU Mr. Wolf, achieving an accuracy of 70.9%, and 0.627mJ per inference with the delay of 5.82ms on the popular BCIC-IV-2a dataset [27]. However, these implementations are aiming at MI tasks on the general purpose MCU. Generalized optimization on EEGNet for different BCI applications in customized circuit design remains to be carried out.

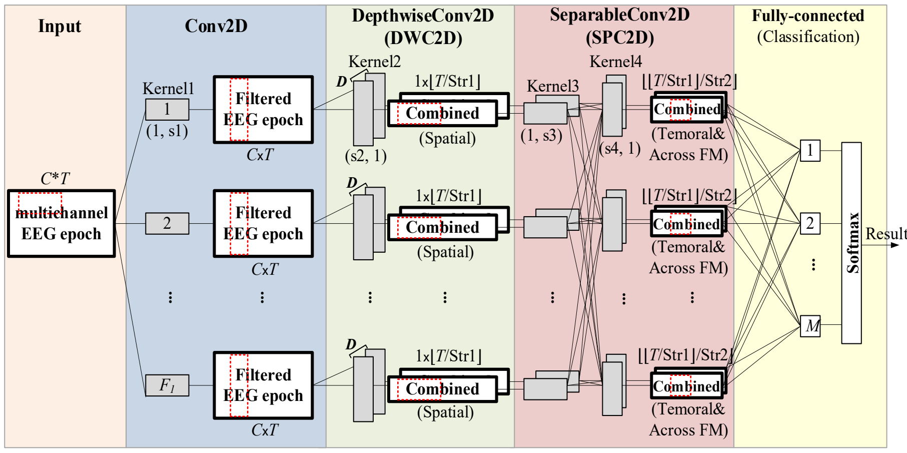

Fig. 1. Overview of EEGNet, where C is the number of electrode channels, T is the number of sample points, F_1 is number of the spatial filters, D is the depth parameter, Str1 is the pooling stride of the pooling layer in DWC2D, Str2 is the pooling stride of the pooling layer in SPC2D, and FM represents the feature map. The kernel sizes of Kernel1, Kernel2, Kernel3, and Kernel4 are (1, s1), (s2, 1), (1, s3), and (s4, 1), respectively. The dashed red rectangles are added to show how the convolutions in each layer works.

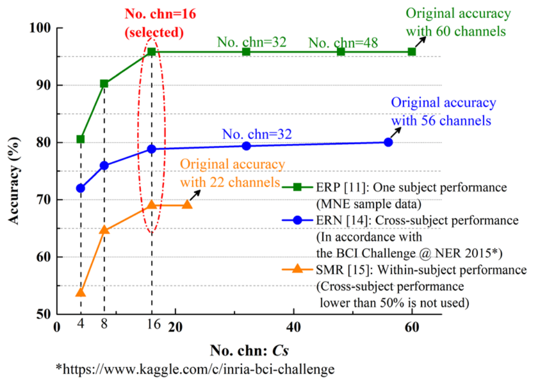

Fig. 2. The inference accuracies on the three datasets of EEGNet (D=2, F_1=8) versus the number of the selected channels. Within subject accuracy is averaged over all subjects.

III. The Compression of EEGNet

A. Overview of EEGNet

As shown in Fig. 1, EEGNet is consisted of five blocks to execute the end-to-end learning and inference from the raw input multichannel EEG epoch.

Firstly, the input multichannel EEG epoch with C channels and T sample points is filtered by F_1 kernels (Kernel1 in Fig. 1 with size (1, s1)) in Conv2D block.

In the subsequent DepthwiseConv2D (DWC2D) block, $D\times F_1$ kernels (Kernel2 with size (s2, 1)) are designed to optimally combine the C electrode channels in spatial domain within each convolutional (conv) channel, where D is the depth parameter controlling the number of spatial filters.

In the SeparableConv2D (SPC2D) block, depthwise convolution (dwconv) and pointwise convolution (pwconv) are executed sequentially in order to obtain the high-level features. Kernel3's of dwconv with size (1, s3) are used to execute the temporal convolution of the spatially-combined output feature maps (FMs) from DWC2D within each conv channel. Kernel4's of pwconv with size (s4, 1) are used to combine the temporally and spatially combined outputs from dwconv across the FMs in all the conv channels.

At last, the FC block with softmax function is adopted as a classifier to tell the category of the input EEG epoch using the generated M products, where M is the number of categories.

B. Embedded Channel Selection

For most EEG-based BCI applications, the number of electrode channel C is relatively large. For example, C=56 for error-related negativity (ERN) classification in P300 speller [38]. F_1 kernels are used to execute filtering of the C electrode channels individually in Conv2D block, and then the C channels are spatially combined to only one dimension right after Conv2D. In this case, the computation complexity of EEGNet is concentrated in Conv2D block, which is highly unbalanced.

The fully parallel implementation of Conv2D leads to huge resource consumption [23], and sequential realization by reusing the filters will lead to long delay. Therefore, channel selection is desired. After reviewing the possible methods [39], and inspired by the norm-based pruning [40], we propose the customized channel selection method for EEGNet, which falls into the category of embedded channel selection method [39].

As illustrated above, Kernel2 in DWC2D block spatially combines the C channels, so the icth weights on jth Kernel2 w_{ic}^{j} (ic=1, 2, …, C; j=1, 2, …, $D\times F_1$) reflects the importance of icth channel. In this case, the l_1-norms of the weights with same ic are calculated for the pretrained network according to (1) and then sorted from largest to smallest for selecting the top Cs channels.

$$l_{1,ic} = \sum_j ||w_{ic}^{j}|| \tag{1}$$

The EEGNet with selected channels are then retrained for good performance. The criterion of l_1-norm is with much lower computation complexity compared to the l_2-norm designed for the MI-BMInet [35] targeting MI task. The retraining process is necessary for keeping accurate across BCI paradigms.

TABLE II
MODEL PARAMETERS OF THE PRESENTED DESIGN AND Q-EEGNET FOR MI-BCI (FOR SMR DATASET)

Block	Layer	Kernel Number	Kernel Size	Output Map Size
Input	Input			(C, T) Ours:(16, 256); Q-EEGNet:(22, 1125)
Conv2D	Temporal Conv[1]	F_1 Ours:8; Q-EEGNet:8	(1, s1) Ours:(1,32); Q-EEGNet:(1, 64)	(F_1, C, T) Ours:(8, 16, 256); Q-EEGNet:(8, 22, 1125)
	BatchNorm			
DWC2D	Spatial Conv[2]	$F_1 \times D$ Ours:16; Q-EEGNet:16	(s2, 1) Ours:(16, 1); Q-EEGNet: (22, 1)	$(F_1 \times D, 1, T)$ Ours:(16, 1, 256); Q-EEGNet:(16, 1, 1125)
	BatchNorm + Activation Ours:ELU; Q-EEGNet:ReLU			
	AvePool1		(1, Str1) Ours:(1, 4); Q-EEGNet: (1, 8)	$(F_1 \times D, 1, \lfloor T/\text{Str1} \rfloor)$ Ours:(16, 1, 64); Q-EEGNet: (16, 1, 140)
SPC2D	Temporal+Spatial Conv	$F_1 \times D + F_1 \times D$ Ours:16+16 Q-EEGNet: 16+16	(1, s3) + (s4, 1) Ours:(1, 16) + (16, 1) Q-EEGNet: (1, 16) + (16, 1)	$(F_1 \times D, 1, \lfloor T/\text{Str1} \rfloor)$ Ours:(16, 1, 64); Q-EEGNet: (16, 1, 140)
	BatchNorm + Activation: Ours:ELU; Q-EEGNet:ReLU			
	AvePool2		(1, Str2) Ours:(1, 8); Q-EEGNet: (1, 8)	$(F_1 \times D, 1, \lfloor \lfloor T/\text{Str1} \rfloor/\text{Str2} \rfloor)$ Ours:(16, 1, 8); Q-EEGNet: (16, 1, 17)
	Flatten			$(F_1 \times D \times \lfloor \lfloor T/\text{Str1} \rfloor/\text{Str2} \rfloor)$ Ours: 128; Q-EEGNet: 272
FC	Dense	$M \times F_1 \times D \times \lfloor \lfloor T/\text{Str1} \rfloor/\text{Str2} \rfloor$ Ours:512; Q-EEGNet:1088		M Ours: 4; Q-EEGNet: 4

1: Temporal Conv: Horizontal convolution along time steps;
2: Spatial Conv: Vertical convolution along electrode channels or filter channels.

Three datasets, C=56 for ERN classification in P300 speller [38], C=60 for event-related potential (ERP) recognition [41], and C=22 for sensory motor rhythm (SMR) inference in MI task [27] are used to verify the proposed method.

The sample rates are 200Hz for ERN, 151Hz for ERP, and 250Hz for SMR datasets, respectively. Firstly, the EEG signal in ERN dataset is filtered between 1-40Hz, and extracted [0, 1.3]s post feedback as trials. The EEG signal in ERP dataset is selected [0, 1]s post visual/audio stimulation as trials. The EEG signal in SMR dataset is resampled to 128 Hz and extracted [0.5, 2.5]s post cue as trials, and is preprocessed using the same procedure in [17], in order to guarantee a fast response from the system. The parameters of s1, F_1, D, Str1, and Str2 are 32, 8, 2, 4, and 8, respectively, and s2=Cs, M=the No. of categories in each dataset.

As shown in Fig. 2, the channel-selected EEGNet keeps accurate at first, and after the channel number is lower than a certain level, the loss of effective channel leads to the large drop of accuracies. An optimal number of 16 is selected to tradeoff between complexity and accuracy, which leads to the compression ratio of 1.375 (22/16), 3.5 (56/16), and 3.75 (60/16) respectively for the three datasets, and the max loss of accuracy (-0.80% from 80.59% to 79.79%) on ERN dataset is negligible [25]. Table II shows detailed layer sequence, and the model parameters of the presented design and Q-EEGNet in each layer for MI-BCI (SMR dataset). The electrode channel number of our design is decreased from 22 to 16, and the sample number is decreased from 1125 to 256, which leads to a reduction in the parameter and operation numbers.

C. Merging Normalizations

The original equation of normalization [42] is shown in (2).

$$x_{\text{nm}} = \gamma(x_{in} - \mu)/\sigma + \beta \tag{2}$$

where x_{in} is the input value of the norm layer, x_{nm} is the output from the norm layer, μ is the mean and σ is the variance of the mini-batch, respectively, γ is the scale parameter, and β is the shift parameter.

Equation (2) can be transformed to a more compact version as shown in (3).

$$x_{\text{nm}} = a(x_{in} - b) \tag{3}$$

where $a = \gamma/\sigma$ and $b = \mu - \beta\sigma/\gamma$.

The F_1 extracted FMs from Conv2D block with the size of (Cs, T) are required to be normalized before the spatial convolution in DWC2D block, whose $D \times F_1$ output FMs with size of (1, T) are also followed by the normalization [21]. Utilizing this compact version in (3), the two adjacent normalizations and the spatial convolution between them can be expressed as (4).

$$\begin{cases} x_{nm}^{Conv2D}(ic,t,f) = a_1(x_{in}^{Conv2D}(ic,t,f) - b_1) \\ x_{in}^{DWC2D}(t,f,d) = w_{nm}^{DWC2D}(:,d) * x_{nm}^{Conv2D}(:,t,f) \\ x_{nm}^{DWC2D}(t,f,d) = a_2[x_{in}^{DWC2D}(t,f,d) - b_2] \end{cases} \tag{4}$$

where $x_{in}^{Conv2D}(ic,t,f)$ represents the tth sample point of icth electrode channel on the FM in the fth conv channel (t=1, 2, …, T, ic=1, 2, …, Cs, f=1, 2, …, F_1), and $x_{nm}^{Conv2D}(ic,t,f)$ is its normalized value. $x_{in}^{DWC2D}(t,f,d)$, d=1, …, D, denotes the tth spatially combined value within the fth conv channel through the spatial convolution between $w_{nm}^{DWC2D}(:,d)$ and $x_{nm}^{Conv2D}(:,t,f)$ in DWC2D block, and $x_{nm}^{DWC2D}(t,f,d)$ is its normalized value. $a_1 = \gamma_1/\sigma_1$ and $b_1 = \mu_1 - \beta_1\sigma_1/\gamma_1$ are the compact parameters of the normalization before the spatial convolution, and $a_2 = \gamma_2/\sigma_2$ and $b_2 = \mu_2 - \beta_2\sigma_2/\gamma_2$ are the parameters after the spatial convolution.

In this case, the two normalizations with only the linear convolution between them can be merged accordingly [25], i.e., since a_1, b_1, a_2, and b_2 are the already learned parameters during inference, the multiplication in the first normalization can actually be removed as shown in (5).

$$\begin{cases} x_{nm}^{Conv2D,m}(ic,t,f) = x_{in}^{Conv2D}(ic,t,f) - b_1 \\ x_{in}^{DWC2D,m}(t,f,d) = w_{nm}^{DWC2D}(:,d) * x_{nm}^{Conv2D,m}(:,t,f) \\ x_{nm}^{DWC2D}(t,f,d) = a_1 a_2 [x_{in}^{DWC2D,m}(t,d,f) - b_2/a_1] \end{cases} \tag{5}$$

where $x_{nm}^{Conv2D,m}$ and $x_{in}^{DWC2D,m}$ denotes the modified output values from the first normalization and the spatial convolution, respectively.

A multiplier can be saved in terms of area occupation. Considering that there are $Cs \times T \times F_1$ values required to be normalized, a lot of energy is also saved.

D. Product Quantization of Fully-Connected Layer

In the FC layer, outputs from SPC2D block are flattened to calculate the weighted sum, which is practically the product between two high dimensional vectors.

Product quantization (PQ) [43] is a classic vector quantization algorithm for approximating inner products of $\boldsymbol{w}^T \cdot \boldsymbol{x}$, where $\boldsymbol{w} \in \mathbf{R}^N$ are already learned and fixed, $\boldsymbol{x} \in \mathbf{R}^N$ are from training/testing set.

Formally, a quantizer is a function q mapping $\boldsymbol{x}$ to a vector (centroid) $\boldsymbol{c}=q(\boldsymbol{x})\in\boldsymbol{C}=\{\boldsymbol{c}_i;\ i\in\boldsymbol{I}\}$, where the $\boldsymbol{c}$'s are obtained using k-means clustering, and the size of index set $\boldsymbol{I}$ is k: $\boldsymbol{I}=\{0, 1, ..., k\text{-}1\}$. With the learned centroids that guarantee $\boldsymbol{w}^T \cdot \boldsymbol{x} \approx \boldsymbol{w}^T \cdot \boldsymbol{c}$, ($\|\boldsymbol{x}-\boldsymbol{c}\|$ is small enough), a speedup can be obtained by storing the precomputed $\boldsymbol{w}^T \cdot \boldsymbol{c}$ to form the codebook (***CB***) of size k, and then finding the optimal index i of ***CB*** by calculating Euclidean distance between $\boldsymbol{c}_i$ and $\boldsymbol{x}_{new}$ when the new $\boldsymbol{x}_{new}$ comes [43].

To avoid multiplications in calculating Euclidean distance, hashing method is used to find the optimal index i for the new input $\boldsymbol{x}$. The balanced binary regression trees (BBRTs) with fixed structure, 4 levels (1, 2, 4, 8 nodes in each level) and 16 leaves (k=16), are developed as hash quantizers [44] to get the optimal index i through simple comparisons. The input vector $\boldsymbol{x}$ can be further split into m distinct sub-vectors $\boldsymbol{u}_{jx}$, $1 \leq jx \leq m$, of dimension N/m, in order to address the high dimensional problem. The sub-vectors are quantized separately by m distinct sub-quantizers, and the m products between the sub-centroids and sub-weights from $\boldsymbol{CB}_{jx}$ are summed to get the approximate product.

The simulated inference accuracies versus m on the three datasets (ERN [38], ERP [41], SMR [27]) with PQ using BBRT hash quantizers are shown in Fig. 3. When $m \geq 8$, the accuracies with PQ varies slightly, and are close to that without PQ. The value of m determines the number of BBRT quantizers, and each BBRT quantizer is a tree with 15 nodes and a codebook with the depth of 16, hence m=8 is selected to tradeoff between area occupation and accuracy.

E. Analyzation of layer interleaving and reordering

Layer interleaving between Conv2D and DWC2D blocks is proposed to remove the memory required between the two layers in Q-EEGNet [25]. As shown in Fig. 4, only a single column during the temporal convolution in Conv2D block is computed per cycle, followed by computing a single output of the spatial convolution in DWC2D block. The same memory location is reused to store the intermediate result, so the $Cs \times T$ elements required to be stored between the two layers can be reduced to $Cs \times 1$ elements.

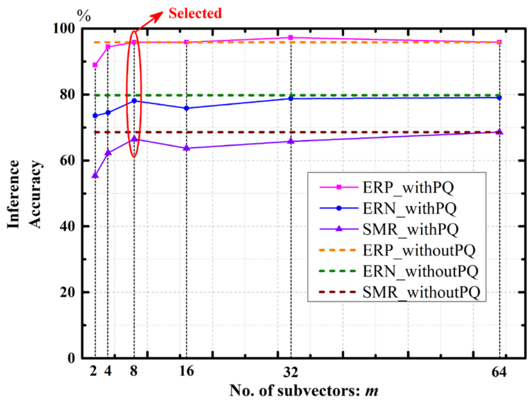

Fig. 3. The inference accuracies on ERP, ERN, and SMR datasets versus the number of sub-vectors (m).

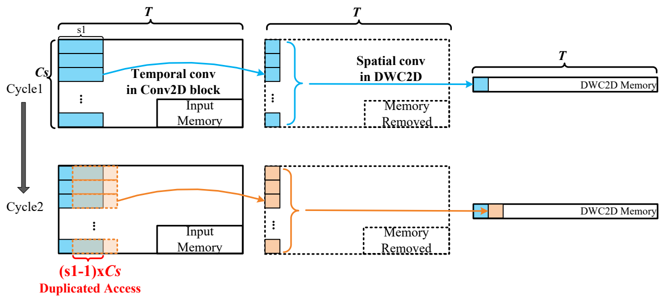

Fig. 4. Analyzation of layer interleaving.

However, $Cs \times$(s1-1) duplicated memory accessing occurs in this way as shown in Fig. 4. Large amount of extra energy is consumed by frequent and duplicated memory accessing, even there are caches in MCU. In terms of power consumption, traditional realization with shifting registers storing the adjacent elements for Conv2D is preferred for the customized circuit implementation.

Both spatial convolution in DWC2D block and separable convolution in SPC2D blocks are followed by batch normalization (BN), nonlinear activation, and average pooling layers. Through replacing the Exponential Linear Unit (ELU) activation by Rectified Linear Unit (ReLU), Q-EEGNet calculates average pooling before BN and activation, so the number of computation can be reduced.

However, this technique only works for 4.5 seconds EEG epoch to guarantee the negligible impact of -0.1% on the classification accuracy which is not practical in use as discussed above. Replacing ELU by ReLU decreases the accuracy by 3% for 2 seconds EEG epoch, which is not negligible. For the nonlinear exponential function in ELU, reordering will generate a complex equation and is not beneficial for hardware realization.

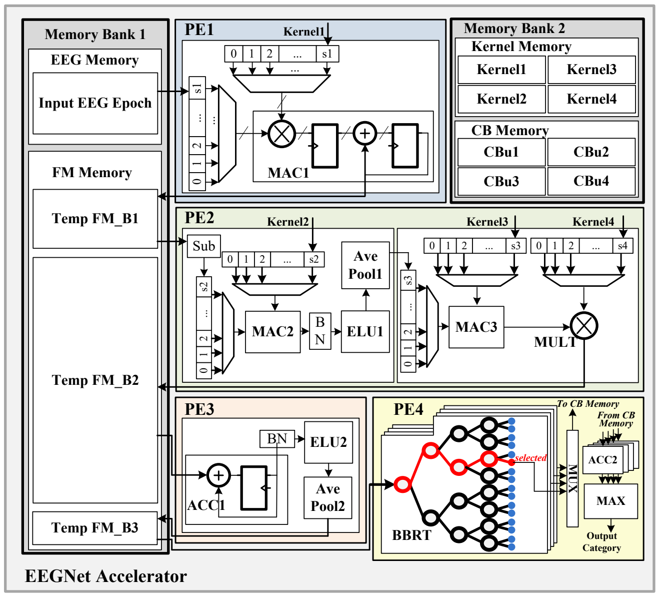

Fig. 5. Overview of the proposed efficient EEGNet accelerator.

IV. The Design of the Efficient EEGNet Accelerator

The overview of the proposed efficient EEGNet accelerator is shown in Fig. 5, which is composed of four processing elements (PE1~PE4) and two memory banks (Memory Bank 1 and 2). Memory Bank 1 is consisted of EEG memory storing the input EEG epoch for classification, and FM memory storing FMs, including Temp FM_B1~FM_B3. Memory Bank 2 is consisted of Kernel Memory storing the pre-learned kernels used in the three blocks, and CB memory storing the pre-learned codebooks for PQ. Details are shown as follows.

A. Structure of PE1~PE3

PE1 executes the convolution towards the raw multichannel EEG signal in Conv2D block. To minimize the resource occupation, PE1 is consisted of two shift-registers and a multiplying-accumulator (MAC1) to accomplish convolution, and PE1 is reused *Cs* times to accomplish convolutions sequentially towards the input EEG epoch with *Cs* channels. The same Kernel1 from Memory Bank 2 is buffered in the shift register during the *Cs* times temporal convolution. By using the shift register, the input EEG data is read only once in each conv channel to avoid duplicated access. The results are stored in temporary feature map memory of block 1 (Temp FM_B1) in Memory Bank 1.

PE2 calculates the spatial convolution of the *Cs* channels in DWC2D block by reusing MAC2, and the temporal convolution in SPC2D block with the same logic as PE1 by reusing MAC3. ELU1 and average pooling (Ave Pool1) after the spatial convolution in DWC2D block are also accomplished in PE2. The results are stored in Temp FM_B2 in Memory Bank 1. According to the above discussion about merging normalization, only a subtract operation is designed before spatial convolution, and the BN circuit is the normal operation with scaling and bias. In this way, a multiplier (for scaling before spatial convolution) is dismissed. A multiplier is designed to execute part of pointwise convolution in SPC2D block and will be described in detail in next section.

PE3 combines the $D\times F_1$ feature maps across the F_1 conv channels by reusing an accumulator (ACC1). The results after BN, ELU2, and Ave Pool2 are stored in Temp FM_B3 as a flatten operation before the FC layer.

B. Reusing PEs and Decomposing Pointwise Conv

As shown in Fig. 1, the computations in the F_1 conv channels are independent before the pointwise convolution in SPC2D block. Therefore, PE1 and PE2 processing temporal convolutions in Conv2D block and spatial convolutions in DWC2D block can be reused F_1 times to achieve the F_1 filtering and the consecutive steps within this conv channel toward the raw EEG signal. In this way, only one group of PE1 and PE2 is required, and Temp FM_B1 only stores FM in one conv channel so as to minimize memory depth.

The input epoch is processed sequentially in the PEs, therefore, the calculation in the PEs are independent, and their processing time can be adjusted adaptively. Within each PE, by changing the reused times of MAC, BN, and ELU circuits, the design can fit the different epoch lengths. Specifically, one need to adjust the boundary value of the counters in the PEs.

On the other hand, PE1 is reused *Cs* times to accomplish convolutions sequentially towards the input EEG epoch with *Cs* channels. PE2 calculates the spatial convolution of the *Cs* channels in DWC2D block, which is practically adding *Cs* weighted products (using Kernel2 with size (s2, 1) where s2=Cs). PE3 and PE4 are not related to the channel number *Cs* (also named the electrode number). Hence, the variation of *Cs* only influences the reused times of PE1, and size of Kernel2, which can be realized by adjusting the boundary value of the counters in PE1, and the length of the shift-register for MAC2.

With the kernel size of ($D\times F_1$, 1), the pointwise convolution can be expressed by (2).

$$FM_o(zo,t) = \sum_{zi=1}^{F_1\times D} FM_i(zi,t)W(zo,zi) \quad (2)$$

where $FM_o(zo, t)$ represents the *t*th sample of the output feature map in the *z*th output conv channel, $FM_i(zi, t)$ is the *t*th sample of input feature map FM_i in the *k*th input conv channel, and $W(zo, zi)$ represents *zi*th weight on the *zo*th pointwise conv kernel.

FM_i is independent of *zo*, so we can split the multiplication and accumulation operations in pointwise convolution. The $\lfloor T/\mathrm{Str1}\rfloor$ samples can be multiplied by $s4=D\times F_1$ kernels (Kernel4) immediately after the depthwise convolution in SPC2D before involving all the outputs of F_1 channels, which can be reused and hence added to PE2 (MULT shown in PE2), and the accumulation without need of kernel values is placed in PE3 (ACC1 shown in PE3).

C. Structure of PE4

As illustrated in Section II. *D*, the vector products between the flattened *N*-dimensional vectors and the *N*-dimensional weight vectors are quantized in the FC layer. Hence, only four BBRTs, a multiplexer (MUX), four accumulators (ACC2), and a MAX unit are designed in PE4.

There are four BBRTs since the maximum number of

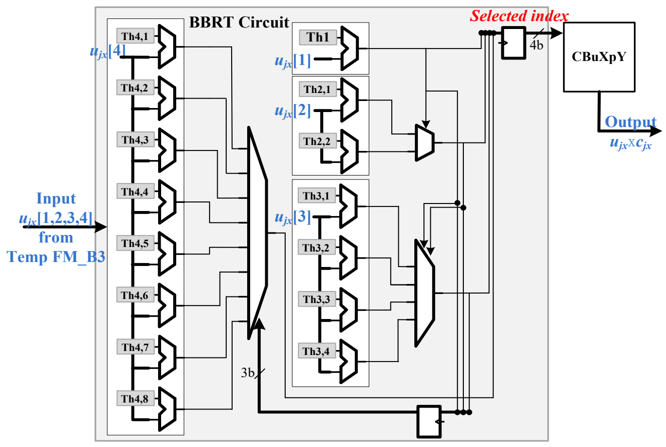

Fig. 6. The BBRT-based product calculation circuit structure of in PE4.

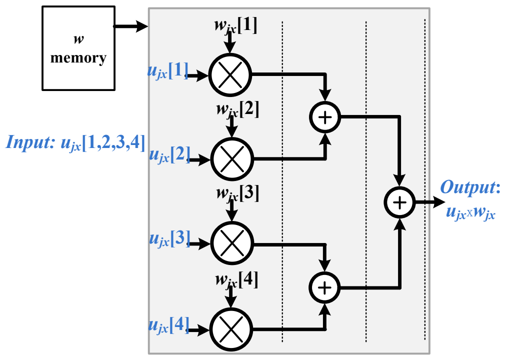

Fig. 7. The traditional MAC-based product circuit with same speed of BBRT-based.

categories of the three datasets is four. The CB memory storing the pre-calculated products between the sub-centroids and sub-weights is then also divided into four units (CBu1, CBu2, CBu3, and CBu4 shown in Fig. 5).

For each category, the N-dimensional vector is divided into 8 sub-vectors according to simulated results shown in Fig. 3, so CBu1~4 are divided into 8 partitions, respectively, and each partition stores a codebook with the depth of 16 according to BBRT algorithm [44].

For the 4-class tasks (ERP and SMR), four BBRTs are used 8 times to get the indexes for reading the 8 optimal products from CBuXpY, X=1, 2, 3, 4, and Y=1, 2, …, 8. The 8 products are summed together using ACC2 in 8 clock cycles, and the four sums are compared in MAX unit to tell the category of the input EEG epoch. For the 2-class tasks, PE4 follows the same principle, but only two BBRTs, two ACC2's, CBu1, and CBu2 are used.

The detailed circuit structure of BBRT is shown in Fig. 6, which is only consisted of 15 threshold registers, 15 comparators, a 2-way multiplexer, a 4-way multiplexer, and an 8-way multiplexer. The four compared results are merged as the index of CBuXpY. Due to the outstanding attribute of BBRT, only four elements of the sub-vector are required by the 4-level tree. In this case, the depth of Temp FM_B3 is only 4×4×8. This is a compression of Temp FM_B3 compared to

TABLE III
Comparisons Between BBRT- and MAC-Based Product Circuit

Performance	BBRT-Based	MAC-Based
Clock Frequency	100MHz	100MHz
LUT	30	77
Flip Flop	12	104
BRAM	2.5	2
Power Consumption	31mW	32mW

4×N when N>32 (which is true for the three datasets using 2 second EEG epochs).

For enough timing margin, two registers are inserted to divide the comparison results into two pipeline stages. Therefore, two clocks are used for the trees to obtain the index. Another clock is used to read the stored product from CB memory. To maintain same speed, traditional MAC-based calculation needs 4 multipliers and 3 adders as shown in Fig. 7 (assuming that the weights are read before multiplication).

The performances of two kinds of PE4 based on the two methods are compared in Table III. At the same clock frequency of 100MHz, the two circuits consume power at the same level, but the BBRT-based only need 39% LUTs and 11% Flip Flops of MAC-based. Counting the compression of Temp FM_B3 memory, we can conclude that BBRT-based product circuit is with good improvement in terms of area.

D. *Softmax to Max*

In the original EEGNet models shown in Fig. 1, softmax is applied to the outputs of FC layer to obtain the category with maximum possibility according to (3)

$$S(FC_{oi}) = \frac{e^{FCoi}}{\sum_{oj=1}^{M} e^{FCoj}} \quad (3)$$

where FC_{oi} is the ith output from FC layer, and M is the total number of FC layer output samples.

The softmax function is involved with complex exponential function, and only for the error back propagation when the target function is chosen as cross-entropy during learning. When inferencing, the softmax function would not change the relative order of the inputs, hence, only the maximum is designed to find in the proposed design (MAX shown in Fig. 5).

E. *Timing Arrangement*

To minimize resource consumption, PE1~4 are reused at different stage. Therefore, the timing arrangement of them is quite important in the proposed design.

As shown in Fig. 8, the processing time of PE1~4 are annotated as T_PE1~4, respectively. PE1 and PE2 work alternately F_1 times to obtain the results in the independent F_1 channels before PE3. PE3 and PE4 are used only once per inference.

The number of output samples from PE1 is annotated as n0. For the 16 channel EEG epoch with T sample points and Kernel1 with size of s1, n0=16×T and T_PE1=16×T×s1. The n0 samples are stored in Temp FM_B1.

The number of output samples from MAC2, AvePool1, and MAC3 are annotated as n1, n2, and n3, respectively. For the 16 channel filtered T samples, MAC2 using Kernel2 of size s2

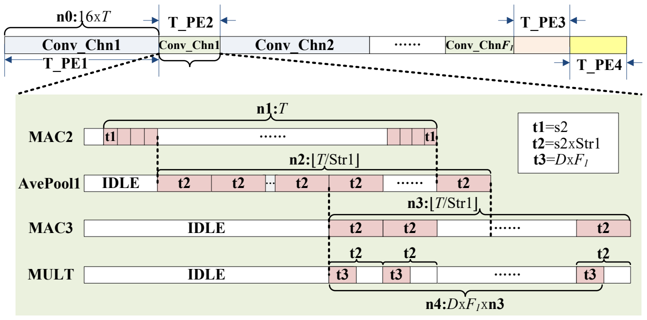

Fig. 8. Timing arrangement of PE1~4.

consumes t1=s2 clock cycles to generate one spatially combined sample, hence MAC2 outputs n1=T samples using s2 × T clock cycles. AvePool1 generates an output after receiving Str1 outputs from MAC2, so n2=⌊T/Str1⌋ and t2=s2×Str1. MAC3 generates the first temporally combined sample after receiving s3/2 outputs from AvePool1, and generates n3=n2=⌊T/Str1⌋ samples in total. For each output from MAC3, D×F_1 products are calculated by MULT in D×F_1 clock cycles and stored in Temp FM_B2. Therefore, the depth of Temp FM_B2 is n4=D × F_1 × ⌊T/Str1⌋. T_PE2 can be estimated as (s2×T+s3×s2×Str1) clock cycles, which is an order of magnitude smaller than T_PE1 due to the spatial combination in DWC2D block.

According to the above analyzation, MAC2, AvePool1, MAC3, and MULT units can be pipelined without storing the intermediate results, so as to save area.

PE3 reads the content in Temp FM_B2 to generate target M × 4 × 8 elements required by PE4, and PE4 executes classification based on BBRT. It can be inferred that T_PE3 and T_PE4 are of the same order of magnitude as T_PE2.

According to above discussion, in most of the processing time, only PE1 is working and PE2~4 is waiting, so the technique of clock gating is utilized to power off PE2~4 when PE1 is activated so as to save power. The embedded channel selection technique which shorten T_PE1 further shows its advantage in the proposed sequential design.

V. Implementation Results

A. Determining Fixed-Point Bit-width

The presented system is coded in Verilog HDL in the signed fixed-point arithmetic. The Memory Banks occupy large area compared to the relatively simple logic circuits, hence, the bit-widths of the variables in the four modules that determine the storing bit-widths of Memory Banks are analyzed in detail.

First, the bit-width of Temp FM_B1 is determined by the bit-width of Kernel1, since MAC operation is conducted between input EEG epoch and Kernel1. The inference accuracies on the three datasets versus Kernel1 bit-widths are shown in Fig. 9. With less than 0.3% decrease on accuracy, the bit-width of Kernel1 is selected to be 8. The increase of accuracy on SMR dataset with the decrease of the bit-width might due to the discard of redundant fractional bits of Kernel1. Second, the bit-width of Temp FM_B2 is determined by the bit-width of Kernel4, since the product between Kernel4 and

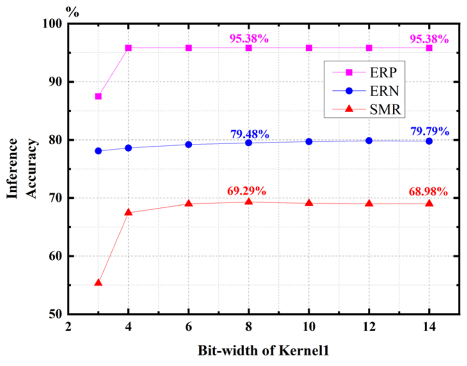

Fig. 9. The inference accuracies on ERP, ERN, and SMR datasets versus the bit-widths of Kernel1.

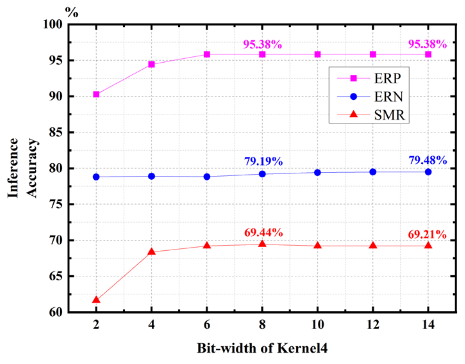

Fig. 10. The inference accuracies on ERP, ERN, and SMR datasets versus the bit-widths of Kernel4, with the bit-width of Kernel1 fixed to 8.

the output from MAC3 is stored in Temp FM_B2. With the bit-width of Kernel1 fixed as 8, the inference accuracies versus the bit-width of Kernel4 are shown in Fig. 10. With less than 0.3% decrease on accuracy, the bit-width of Kernel4 is selected to be 8. The bit-width of Temp FM_B1 and Temp FM_B2 are determined to be 16 accordingly, with the extension of integer bits induced by accumulation.

The bit-width of Temp FM_B3 is determined by the PQ algorithm. A quantization to 8 bits is designed in the original BBRT, so the bit-width of Temp FM_B3 is 8 [44].

B. Performance Comparison and Discussion

Firstly, the proposed design is implemented in a Kintex-7 FPGA. The performance in terms of resource utilization, processing delay, and power consumption of the presented design is shown in Table IV. The comparison between the previous work, the design without model compression, and the design with model compression is also shown in Table IV. The 10Watts power consumption of [23] is estimated according to previous works [45], [46] occupying comparable amount of resource on similar FPGA platforms.

Compared to the one-to-one parallel design in [23], the proposed design with model compression only needs 3.31% (4512/136416) area, and 1.35% power consumption (0.135/10). Compared to the implementation without model compression,

TABLE IV
COMPARISON WITH PREVIOUS FPGA WORK AND THE IMPLEMENTATION WITHOUT AND WITH MODEL COMPRESSION

Performance	Tsukahara et al. [23]	Without compression	With compression
Platform	Xilinx VC707	Xilinx KC705	Xilinx KC705
LUT	207896 for ERP 191331 for ERN	2771	**2724**
Flip Flop	147427 for ERP 136416 for ERN	4604	**4512**
DSP	2492 for ERP 2276 for ERN	21	21
BRAM	180 for ERP 168 for ERN	63	**19.5**
Freq.	100MHz	100MHz	100MHz
Power	~10W	136mW	135mW
Delay	58us for ERP 115.85us for ERN	26.7ms for ERP 43.0ms for ERN 16.6ms for SMR	**7.2ms** for ERP **12.4ms** for ERN **12.2ms** for SMR

TABLE V
COMPARISON WITH Q-EEGNET IN MR. WOLF AND THE PROPOSED DESIGN WITH MODEL COMPRESSION ON SMR DATASET

Performance	Q-EEGNet implemented in Mr. Wolf [25]	Proposed design with compression
Epoch length	4.5 seconds	**2 seconds**
Accuracy	**70.9%**	68.80%
Process	40nm CMOS LP	65nm CMOS LP
Gate counts (NAND2)	1.8 M	**0.23 M**
Memory (kB)	**35.41**(SISD)/68.15(SIMD)	49.88
Time /Inference (ms)	31.98(SISD)/28.67(SIMD)	**24.4**
Energy /Inference (mJ)	0.375(SISD)/0.337(SIMD)	**0.267**
Frequency	50MHz	50MHz
Power (mW)	11.75	**10.97[*] (clock gated for PE2~4)** **24.51[*] (clock not gated for PE2~4)**

SISD: Single Instruction Single Data;
SIMD: Single Instruction Multiple Data
*: Synthesized result

the proposed design achieves the acceleration with the speeding up factor of 1.4 (16.6/12.2), 3.5 (43/12.4), and 3.7 (26.7/7.2), respectively, on the three datasets.

The design with model compression is then implemented in 65nm CMOS Low Power (LP) process, and synthesized for evaluating the performance. The resource utilization, including area and power, of the proposed design, and the performance comparison with Q-EEGNet implemented in Mr. Wolf MCU is shown in Table V.

As can be observed, the presented accelerator design achieves the accuracy of 68.80% on SMR dataset for motor imagery task, which is 2% lower than that of Q-EEGNet but responds 2.25× faster (2/4.5). There is a drop of accuracy from 69.44% in Fig. 10 to 68.80%, because 69.44% in Fig. 10 is the result with other layers in float-point, and 68.80% is the result with other layers in fixed-point. There is a loss of accuracy due to float-to-fixed conversion.

In terms of area, the proposed accelerator only consumes 0.23M equivalent gate counts which is 87.22% less than that of Mr. Wolf MCU (0.23/1.8), since it is a customized design with PE reusing and without instruction fetching and decoding or buses. The 49.88kB memory occupation of the design is calculated by multiplying the number of variables (including input EEG epoch, feature maps, kernels, codebooks, and two tables in ELU module using table driven method) and their bit-widths, which is between the minimum 35.41kB of the single instruction single data implementation of Q-EEGNet and 68.15kB of single instruction multiple data (SIMD) implementation.

Thanks to the small area occupation and clock gating of PE2~4, the accelerator only consumes 10.97mW (synthesized result from Design Compiler). The power consumption without using clock gating is also listed to show the effectiveness of this technique. The estimated power consumption by synthesizing is generally smaller than the measured result. We have added the toggle rates of the signals according to the simulation result to minimize this gap into same order of magnitude. Considering we are using the 65nm CMOS process which has two generational gaps in terms of processing node compared to 40nm CMOS of Mr. Wolf, the power efficiency of the proposed design is guaranteed.

In terms of processing speed, due to the deployed channel selection technique and short epoch, the accelerator could do one inference in 24.4ms, which outperforms the fastest Q-EEGNet implementation of 28.67ms using SIMD.

The low power consumption and the fast processing speed lead to the energy of 0.267mJ per inference which is 20.77% more efficient than 0.337mJ/inference of Q-EEGNet.

VI. CONCLUSION

An efficient EEGNet inference accelerator based on the compressed EEGNet model and quantized fully-connected layer is proposed in this paper. The EEGNet model is compressed by embedded channel selection and normalization merging. The fully-connected layer which calculates vector products is quantized for efficiency improvement. Processing elements in the presented design is reused to saves area, and clock-gated to save power. The accelerator design is implemented in FPGA for three datasets, and synthesized using 65nm CMOS Low Power (LP) process. Experimental results in FPGA show that by reusing the processing elements and clock-gating, 3.31% area and 1.35% power compared to the one-to-one parallel design can be achieved. The model compression further offers the speedup factors of 1.4, 3.5, and 3.7 compared to the non-compressed design on the three datasets. Synthesized results in 65nm CMOS LP process show that the designed accelerator only need 12.78% gate counts 79.23% energy per inference compared to the implementation of Q-EEGNet in Mr. Wolf MCU on BCIC-IV-2a dataset, which is more suitable for lightweight BCIs with near-sensor intelligence.

REFERENCES

[1] J. R. Wolpaw, N. Birbaumer, D. J. McFarland, G. Pfurtscheller, and T. M. Vaughan, “Brain-computer interfaces for communication and control,” *Clin. Neurophysiol.*, vol. 113, no. 6, pp. 767-791, 2002.

[2] J. Yang and M. Sawan, “From seizure detection to smart and fully embedded seizure prediction engine: a review,” *IEEE Trans. Biomed. Circuits Syst.*, vol. 14, no. 5, pp. 1008-1023, 2020.

[3] A. B. Schwartz, X. T. Cui, D. J. Weber, and D. W. Moran, “Brain-controlled interfaces: movement restoration with neural prosthetics,” *Neuron*, vol. 52, no. 1, pp. 205-220, 2006.

[4] Y. Yu, *et al.*, “Self-Paced Operation of a Wheelchair Based on a Hybrid Brain-Computer Interface Combining Motor Imagery and P300 Potential,” *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 25, no. 12, pp. 2516–2526, 2017.

[5] O. Valentin *et al.*, “Validation and benchmarking of a wearable EEG acquisition platform for real-world applications,” *IEEE Trans. Biomed. Circuits Syst.*, vol. 13, no. 1, pp. 103–111, 2019.

[6] P. Sawangjai, S. Hompoonsup, P. Leelaarporn, S. Kongwudhikunakorn, and T. Wilaiprasitporn, “Consumer Grade EEG Measuring Sensors as Research Tools: A Review.” *IEEE Sensors J.*, vol. 20, no. 8, pp. 3996-4024, 2020.

[7] P. Arico, N. Sciaraffa, and F. Babiloni, “Brain-computer interfaces: Toward a daily life employment,” *Brain Sci.*, vol. 10, no. 3, 2020.

[8] J. Cheng, M. Chen, C. Li, Y. Liu, R. Song, A. Liu, and X. Chen, “Emotion recognition from multi-channel EEG via deep forest,” *IEEE J. Biomed. Health Inform.*, vol. 25, no. 2, pp. 453–464, 2020.

[9] U. Budak, *et al.*, “An effective hybrid model for EEG-based drowsiness detection,” *IEEE Sensors J.*, vol. 19, no. 17, pp. 7624–7631, 2019.

[10] Z. Qiu, B. Z. Allison, J. Jin, *et al.*, “Optimized motor imagery paradigm based on imagining Chinese characters writing movement,” *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 25, no. 7, pp. 1009–1017, 2017.

[11] W. Dang, *et al.*, “MHLCNN: Multi-Harmonic Linkage CNN Model for SSVEP and SSMVEP Signal Classification,” *IEEE Trans. Circuits Syst. II: Express Brief*, vol. 69, no. 1, pp. 244-248, 2022.

[12] Y. Liu *et al.*, “Motor-Imagery-Based Teleoperation of a Dual-Arm Robot Performing Manipulation Tasks,” *IEEE Trans. Cogn. Develop. Syst.*, vol. 11, no. 3, pp. 414-424, 2019.

[13] L. Shao, *et al.* “EEG-controlled wall-crawling cleaning robot using ssvep-based brain-computer interface,” *J. Healthcare Eng.*, vol. 2020, pp. 1–11, 2020.

[14] M. Congedo, A. Barachant, and R. Bhatia, “Riemannian geometry for EEG-based brain-computer interfaces; a primer and a review,” *BrainComputer Interf.*, vol. 4, pp. 1–20, 2017.

[15] K. Ang, Z. Chin, H. Zhang, and C. Guan, “Filter Bank Common Spatial Pattern (FBCSP) in Brain-Computer Interface,” *Proc. Int. Joint Conf. Neural Networks*, pp. 2391-2398, June 2008.

[16] K. He, X. Zhang, S. Ren and J. Sun, “Deep Residual Learning for Image Recognition,” *IEEE Conf. Computer Vision Pattern Recog.* (*CVPR*), 2016, pp. 770-778.

[17] 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.

[18] P. Autthasan *et al.*, “MIN2Net: End-to-End Multi-Task Learning for Subject-Independent Motor Imagery EEG Classification,” in *IEEE Trans. Biomed. Eng.*, early access, 2021.

[19] Z. Gao, T. Yuan, X. Zhou, C. Ma, K. Ma, and P. Hui, “A Deep Learning Method for Improving the Classification Accuracy of SSMVEP-Based BCI,” *IEEE Trans. Circuits Syst. II: Express Brief*, vol. 67, no. 12, pp. 3447-3451, 2020.

[20] O. B. Guney M. Oblokulov, and H. Ozkan, “A Deep Neural Network for SSVEP-Based Brain-Computer Interfaces,” *IEEE Trans. Biomed. Eng.*, vol. 69, no. 2, pp. 932-944, 2022.

[21] V. J. Lawhern, A. J. Solon, N. R. Waytowich, S. M. Gordon, C. P. Hung, and B. J. Lance, “EEGNet: A Compact Convolutional Network for EEG-based Brain-Computer Interfaces,” *J. Neural Eng.*, vol. 15, no. 5, pp. 056013.1-056013.17, 2018.

[22] C. Beach, E. Balaban, and A. J. Casson, “Chapter 14 - Edge algorithms for wearables: An overview of a truly multi-disciplinary problem,” in *Wearable Sensors*, E. Sazonov, Ed., 2nd ed. Oxford, U.K.: Academic Press, 2021, pp. 379–414.

[23] A. Tsukahara, Y. Anzai, K. Tanaka, Y. Uchikawa, “A design of EEGNet-based inference processor for pattern recognition of EEG using FPGA,” *IEEJ Trans. Electronics. Inform. Syst.*, vol. 104, pp. 53-64, 2021.

[24] X. Wang *et al.*, “An accurate EEGNet-based motor-imagery brain-computer interface for low-power edge computing,” in *Proc. IEEE Int. Symp. Med. Meas. Appl.*, 2020, pp. 1–6.

[25] T. Schneider *et al.*, “Q-EEGNet: An energy-efficient 8-bit quantized parallel EEGNet implementation for edge motor-imagery brain-machine interfaces,” in *Proc. IEEE Int. Conf. Smart Comput.*, 2020, pp. 284–289.

[26] A. Pullini, D. Rossi *et al.*, “Mr.Wolf: An Energy-Precision Scalable Parallel Ultra Low Power SoC for IoT Edge Processing,” *IEEE J. Solid-State Circuits*, vol. 54, no. 7, pp. 1970–1981, 2019.

[27] M. Tangermann, K. R. Müller, A. Aertsen, *et al.*, “Review of the BCI Competition IV,” *Front. Neurosci.*, vol 6, pp. 55, 2012.

[28] S. Sakhavi, C. Guan, and S. Yan, “Parallel convolutional-linear neural network for motor imagery classification,” in *Proc. EUSIPCO*, 2015, pp. 2786–2790.

[29] N. Lu, T. Li, X. Ren, and H. Miao, “A deep learning scheme for motor imagery classification based on restricted Boltzmann machines,” *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 25, no. 6, pp. 566–576, Jun. 2017.

[30] K. Belwafi *et al.*, “An embedded implementation based on adaptive filter bank for brain-computer interface systems,” *J. Neurosci. Methods*, vol. 305, pp. 1–16, 2018.

[31] A. Malekmohammadi *et al.*, “An efficient hardware implementation for a motor imagery brain computer interface system,” *Scientia Iranica*, vol. 26, pp. 72–94, 2019.

[32] X. Wang *et al.*, “Mixed-precision quantization and parallel implementation of multispectral Riemannian classification for brain-machine interfaces,” in *Proc. IEEE Int. Symp. Circuits Syst.*, 2021, pp. 1–5.

[33] D. Rossi *et al.*, “A 1.3TOPS/W, 32GOPS fully integrated 10-core SoC for IoT end-nodes with 1.7uW cognitive wake-up from MRAM-based state-retentive sleep mode,” in *Proc. IEEE Int. Solid- State Circuits Conf.*, 2021, vol. 64, pp. 60–62.

[34] X. Wang *et al.*, “Sub-100μW Multispectral Riemannian Classification for EEG-Based Brain–Machine Interfaces,” *IEEE Trans. Biomed. Circuits Syst.*, vol. 15, no. 6, pp. 1149-1160, 2021.

[35] X. Wang, *et al.*, “MI-BMInet: An Efficient Convolutional Neural Network for Motor Imagery Brain--Machine Interfaces with EEG Channel Selection,” *arXiv preprint*, arXiv: 2203.14592v2, 2022.

[36] K.-K. Shyu, *et al.*, “Development of a low-cost FPGA-based SSVEP BCI multimedia control system,” *IEEE Trans. Biomed. Circuits Syst.*, vol. 4, no. 2, pp. 125–132, 2010.

[37] B-S. Lin, *et al.*, “Design of SSVEP Enhancement-Based Brain Computer Interface,” *IEEE Sensor J.*, vol. 21, no. 13, 2021.

[38] P. Margaux, M. Emmanuel, D. Sebastien, *et al.*, “Objective and Subjective Evaluation of Online Error Correction during P300-Based Spelling,” *Adv. human-computer interaction*, pp. 33.1-33.13, 2012.

[39] M. Z. Baig, N. Aslam, and H. Shum, “Filtering techniques for channel selection in motor imagery EEG applications: a survey,” *Artificial Intelligence Review*, vol. 53, no. 3, 2020.

[40] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, “Pruning filters for efficient convnets,” in *Proc. Int. Conf. Learn. Representations* (*ICLR*), 2017, pp. 1–13.

[41] A. Gramfort, M. Luessi, E. Larson, *et al.*, “MEG and EEG data analysis with MNE-Python,” *Front. Neuroscience*, vol. 7, pp. 267, 2013.

[42] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 448–456.

[43] H. Jégou, M. Douze and C. Schmid, "Product Quantization for Nearest Neighbor Search," in *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 1, pp. 117-128, 2011.

[44] D. Blalock, J. Guttag, “Multiplying Matrices Without Multiplying”, *arXiv preprint*, arXiv:2106.10860, 2021.

[45] J. Li, K.-F. Un, W.-H. Yu, P.-I. Mak, and R. P. Martins, “An FPGA-Based Energy-Efficient Reconfigurable Convolutional Neural Network Accelerator for Object Recognition Applications,” *IEEE Trans. Circuits Syst. II: Express Brief*, vol. 68, no. 9, pp. 3143-3147, 2021.

[46] J. Qiu, J. Wang, S. Yao S, *et al.*, “Going deeper with embedded FPGA platform for convolutional neural network,” in *Proc. ACM/SIGDA Int. Symp. Field-Programmable Gate Arrays.* ACM, 2016, pp: 26-35.