BCIBench: A Benchmarking Suite for EEG-Based Brain Computer Interface

***Abstract*— Increased demands for applications of brain computer interface (BCI) have led to growing attention towards their low-power embedded processing architecture design. Most clinical, wellness, and entertainment applications of BCI require wearable and portable devices. Better understanding of application characteristics in terms of computational complexity, memory usage, and power consumption can lead to more effective system designs for future wearable BCIs. In this paper, we introduce BCIBench, a benchmarking suite which includes a wide range of algorithms used for pre-processing, feature extraction and classification in BCI applications. We analyze the architectural characteristics of these algorithms such as performance, parallelism, data-intensiveness and memory behavior. We provide insights into architectural components that can enhance the performance and reduce the power consumption of BCI embedded systems using these applications.**

Keywords—brain computer interface; wearable; low-power; signal processing algorithms; benchmarking;

I. INTRODUCTION

Recently, embedded systems have been deployed in many applications and become an integral part of daily life. Cell-phones, music players, electronics in cars and patient monitoring devices are some examples of the widely used embedded systems. Real-time computing and signal processing are the principal requirements of these systems. On the other hand, they have limited available resources such as memory, computational resources, screen size, key inputs, etc. This is a crucial property to consider during the design, development and assessment of these systems.

Wearable computers are embedded systems positioned on the body that have the capability of sensing, processing, and communication. Wearable computers promise novel uses in healthcare, wellness, and entertainment. Physiological sensors are one category of sensors deployed with wearable computers that can measure blood pressure, blood oxygenation, electrocardiography (ECG), and electroencephalography (EEG). EEG-based brain computer interface (BCI) is an example of wearable computers where EEG sensors provide communication channels to translate brain rhythms of an individual into application-specific signals for external devices. BCI systems allow a person to use mental processes to communicate with external devices without relying on neuromuscular control [1, 2].

Recently, there has been a dramatic growth in BCI research which demonstrates its application potentials from clinical domains to gaming and entertainment applications [2, 3]. There has been also a growing interest in developing real-time wearable embedded BCIs [4]. In BCI systems, signal processing algorithms and several other parameters such as sampling frequency, number of channels, processing window, and number of features are determined by application requirements. Thus, besides managing the limited available resources, it is crucial to use programmable architectures to amortize cost of a design for many related applications while considering the aforementioned parameters.

There are several platforms that can be used to implement a BCI embedded system. In an ASIC design, designers optimize their design based on the target algorithm specification. Microcontroller-based architecture is another technique which requires benchmarks to accomplish the optimization and evaluation of a design. Another promising approach that recently attracts much research interest is using hardware accelerator along with microcontrollers in order to reduce the power consumption. In this approach, a hardware accelerator is dedicated to the power hungry processing blocks and a microcontroller is used to handle the rest of the processing blocks. The hardware accelerator is employed in many research studies such as in seizure detection system [5], or cardiac monitoring system [6]. In order to optimize an implementation design, all the above-mentioned approaches require taking into account the characteristics of the target BCI application.

The ASIC design requires computation and memory behaviors of the application. In microcontroller-based techniques, parallelism, efficient cache configuration and resource usage of the application are required for the design optimization. In the accelerator-based method, detailed characteristics of each individual algorithm in the processing flow are required to identify the bottleneck of power consumption in the processing flow. Thus, regardless of the implementation platform, a suitable benchmarking suite is necessary to address these demands to optimize and evaluate the embedded BCI designs. Benchmarking not only can facilitate architecture customization based on application specifications, but also can empower hardware/software co-design. In the hardware/software co-design methodology, the computations are assigned to various processing elements such as general purpose processors and hardware accelerators. Target objectives on power consumption and throughput requirements will drive decisions on hardware/software co-design.

Currently, a number of benchmarking suites are developed targeting a variety of domains. SPEC [7] is one of the most widely used benchmarking suites that evaluates the performance of general purpose computers. With the extensive growth in the use of embedded systems, several benchmarking suites for embedded systems have been introduced such as Mibench [8]. The authors introduced embedded applications in six different categories and compared several characteristics such as instruction distribution and memory behaviors of their benchmark suite to those of the SPEC benchmarks. There are other domain-specific benchmarking suites developed for of embedded systems such as ImpBench [9] for implantable

architectures, PARSEC [10] for Chip-Multiprocessors (CMPs), MEVBench [11] for mobile computer vision applications.

Although a few of the BCI signal processing algorithms were presented as part of larger benchmarks, there is no domain-specific benchmarking suite that covers different components of the BCI signal processing algorithms. Moreover, to identify the challenges and bottlenecks of a BCI application, power and performance analysis of end-to-end signal processing flow is necessary which to our knowledge is not investigated in previous benchmarking suites. Also researchers in the BCI area primarily focus on high-level optimization to improve the system accuracy. Thus, there is a lack of benchmark suites for power and computational analysis of the signal processing algorithms in BCI embedded systems.

In this paper, we introduce BCIBench as a benchmarking suite that includes the most widely used BCI signal processing components for wearable embedded applications. We also include several end-to-end signal processing algorithms such as motor imagery and P300 detector BCIs. To the best of our knowledge, BCIBench is the first benchmarking suite dedicated to BCI. Our key contributions in this paper are listed in the following:

- We introduce a new benchmark suite, which covers a set of applications and signal processing algorithms in the BCI domain.
- We perform a detailed micro-architectural analysis of the benchmark indicating performance and memory characteristics.
- Based on the breakdown analysis of individual blocks in the end-to-end applications, we identify the power hungry blocks for further optimization. This analysis shows how future embedded architectures can better serve BCI applications and enhance their power and performance.

The rest of the paper is organized as follows. In section 2, we briefly describe applications of BCIBench. Benchmark characterization with a detailed analysis is covered in section 3. Section 4 covers breakdown analysis of the BCI applications. A brief discussion presented in section 5. Finally, section 6 concludes the paper.

II. BENCHMARK DETAILS

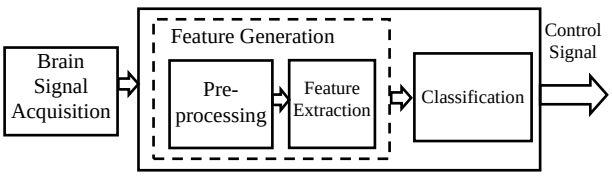

Fig. 1. The overview of a typical BCI application.

BCI applications consist of several signal processing components as shown in Figure 1. First, recorded brain signals are enhanced with filters or re-referencing algorithms. Then discriminative time and frequency features are extracted from the filtered signals. Extracted features are fed to the classifier to produce corresponding output control signal which depends on the specific BCI application. Several studies summarize a comprehensive survey of signal processing methods in BCI applications [12, 13]. Considering the feasibility requirements of low power implementation and limited available resources for wearable embedded BCIs, we selected algorithms with medium to low computational complexity for each of the components. Table I, summarizes the BCIBench programs and applications.

TABLE I. PROGRAMS IN THE BCIBENCH

Pre-processing
Common Average Referencing (CAR)
FIR Filter (FIR)
Surface Laplacian Referencing (Lap)
Feature Extraction
Autoregressive (AR)
Band-Power (BP)
Fast Fourier Transform (FFT)
Higuchi
Hjorth
Wavelet Packet Decomposition (WPD)
Classification
K-Nearest Neighbor (KNN)
Linear Discriminant Analysis (LDA)
Multilayer Perceptron (MLP)
Support Vector Machine (SVM)
End-to-end Applications
BCI1: Motor Imagery BCI
BCI2: VEP BCI
BCI3: Motor Imagery BCI
BCI4: P300 BCI

A. *Pre-Processing*

In most BCI applications, pre-processing is performed on the recorded brain signal prior to the feature extraction due to low signal-to-noise ratio of the signal. In common average referencing (CAR), average value of the samples of all channels at the current time is subtracted from each sample. Laplacian reference adjusts the signal of each channel by removing the average of neighbor channels. Filtering is a very common pre-processing in BCI applications. FIR filters are very popular because of their simple architecture for hardware and software implementation.

B. *Feature Extraction*

Features are the key characteristics of brain data that encode the intent of the user. Typical features of the brain signals include time domain features such as amplitudes or latencies of event-related potentials and frequency domain features such as power spectra and power of different frequency bands. BCIBench includes Autoregressive (AR) algorithm [14], Band-Power (BP) [15], Fast Fourier Transform (FFT), Higuchi Algorithm [16], Hjorth Algorithm [17], and Wavelet Packet Decomposition (WPD). For more details of the algorithms and implementations, please see to the references.

C. *Feature Classification*

The last component in BCI applications is the classifier. The classifier usually employs previously recorded data (online or offline) to generate a model. Then, during the real-time process, they use the trained model to classify the unseen input data. BCIBench comprises four different classifiers that are widely used in BCI applications: 1) Support Vector Machine

(SVM), 2) Linear Discriminant Analysis (LDA), 3) K-Nearest Neighbor (K-NN), 4) Multilayer Perceptron (MLP): MLP. For more details, please refer to [18].

D. End-to-end BCI Applications

BCI applications may utilize different algorithms in the pre-processing, feature extraction and classification stages to interpret the brain signals. There are three major applications that are widely used for BCIs:

- SensoriMotor Rhythm (SMR) classification:

  Comprise mu and beta rhythms, which are the oscillations in the brain activities localized in the mu (7-13 Hz) and the beta bands (13 – 30 Hz). The amplitude of these rhythms varies during cerebral activity of the brain. Cerebral activity is related to motor tasks such as moving a cursor on a screen or controlling a robot [19]. In BCIBench, BCI 1 and BCI 3 end-to-end applications are from this category.

- Visual Evoked Potential (VEP) detection:

  VEPs reflect processing of visual information in the brain. For instance, when the user looks at a light flashing with a constant frequency, the flashing frequency is induced in his EEG signal. The goal is to detect this frequency reliably while the user looks at the stimulus. This paradigm can be used for wheelchair or robot control. BCI2 in the BCIBench is a VEP-based system illustrated that presented in [20].

- P300-Evoked Potential detection:

  When infrequent auditory or visual stimuli are combined with routine stimuli, a positive peak in the EEG is evoked at about 300ms, called "P300", which has been first used as a BCI control signal in [21]. BCI4 in BCIBench is a P300-based BCI for an image selection task. Feature extraction includes down sampling and noise removal.

Table II shows preprocessing, feature extraction and classifier that is implemented for each of the BCI applications.

TABLE II. END-TO-END BCI APPLICATIONS IN BCIBENCH.

System	**BCI1**	**BCI2**	**BCI3**	**BCI4**
BCI category	Motor Imagery	VEP	Motor Imagery	P300
Pre-processing	CAR	FIR	Lap	FIR
Feature extraction	BP	FFT	AR	DownSample
Classifier	LDA	LDA	MLP	LDA

III. BENCHMARK CHARACTERISTICS

A. Why BCIBench?

Most clinical, wellness, and entertainment applications of BCI require wearable and portable devices. Therefore, there has been a growing interest towards their low-power embedded architecture design. There are many efforts in developing such devices from g.tec system [22] which is a dry electrode real-time EEG acquisition system to neural dusts [23], which are tiny electronic sensors the size of dust particles scattered onto the cerebral cortex.

As discussed in section II, BCI systems comprise specific signal processing components that may vary for different BCI applications. There are also other parameters such as sampling frequency, number of channels, processing window, and number of features that are determined by application requirements. Better understanding of application characteristics in terms of computational complexity, memory usage, and power consumption can lead to more effective system designs for future wearable BCIs. It is also crucial to use programmable architectures to minimize the cost of a design for many related applications.

Researchers in the BCI area primarily focus on high-level optimization to improve the system accuracy. Thus, there is a lack of benchmark suites for power and computational analysis of the signal processing algorithms in BCI embedded systems to identify the challenges and bottlenecks of a BCI application. Therefore, a domain-specific benchmarking suite that covers different components of the BCI signal processing algorithms is necessary to analyze power and performance of end-to-end signal processing flow which to our knowledge is not investigated in previous benchmarking suites.

B. Experimental Setup

In order to evaluate the components that constitute BCIBench, we use MARSS x86 simulator [24]. MARSS is an open source simulation tool built on QEMU. All components in BCIBench were compiled using GNU gcc compiler suite version 4.5. MARSS provides a cycle-accurate simulation model for Intel Atom core. Each simulated core is configured as 2-wide in-order machine with 16 load/store buffer entries. Each core has a private 32KB, 2-way set associative L1 cache for instruction and data, and shared L2 cache of size 512KB. We use 100MHz as core frequency. All private caches are kept coherent using the write back protocol on an on-chip split-phase bus.

BCIBench applications are designed to have the parametric property which enables us to run them with any desired configuration. These parameters vary according to the BCI application. For example, clinical epilepsy detection needs 128 channels with 1 KHz sampling frequency while EEG headsets which are used for gaming have 1-4 channels with 128 samples per second. We set the application parameters for our analysis according to Table III. In a BCI application, a trial is a window of time that is required to process brain signals to detect a desired pattern. In this work, each EEG trial has one second of data for 8 channels. With the configuration described in Table III, each trial has 4096 samples. All simulations are repeated for 100 trials of input data to ensure that a stable behavior in simulator is obtained.

TABLE III. BCIBENCH APPLICATION PARAMETERS.

Parameter	Pre-processing	Feature extraction	Classifier
sampling frequency	512	512	NA
#of channels	8	8	NA
processing window	1	1	NA
# of features	NA	40	40
# of training vector	NA	NA	100

In the previous benchmarking suites, all the analysis belongs to running algorithms as well as loading input data to the main memory. This results in incorrect information because I/O file transfer has excessive latency and a significant part of the execution time goes to reading input data files. To resolve this issue, we modified MARSS to collect statistics of any

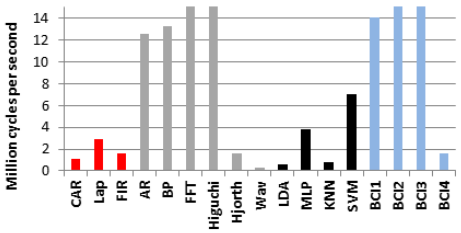

Fig. 4. Excecution time

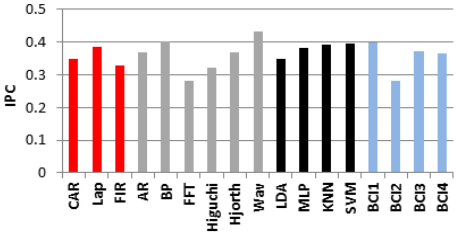

Fig. 5. IPC value

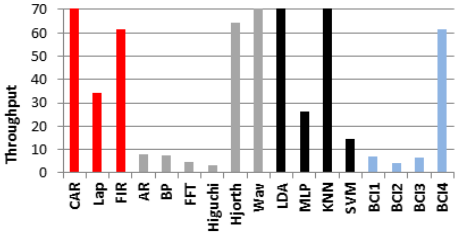

Fig. 6. Throughput

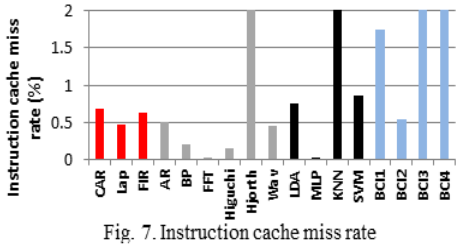

Fig. 7. Instruction cache miss rate

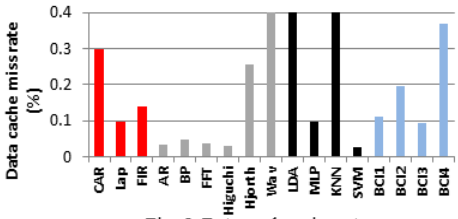

Fig. 8. Data cache miss rate

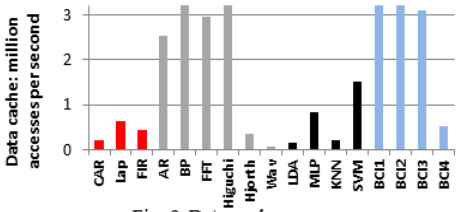

Fig. 9. Data cache access

region of interest in the implementation. This enables us to start collecting statistics after the input data is completely loaded into the main memory.

C. *Data Set*

Providing data sets corresponding to the applications of a benchmark suite is extremely valuable. In this work, instead of using standard data sets, we use real EEG data which was recorded from a state-of-the-art EEG machine during the P300 spell checker, motor imagery and VEP paradigms.

D. *Computational Complexity Analysis*

In ultra-low power design methodology, voltage scaling is a popular approach to reduce energy consumption. In this approach, the supply voltage adjusts according to the peak frequency of the application. Execution time in terms of the number of cycles per second as the peak frequency is a good representation of an algorithm or application requirement. Figure 4 shows the number of cycles per second for BCIBench applications. It also illustrates the difference between the computational complexities of various BCI signal processing components (e.g., filters). In our previous work, we investigated high-level optimization considering computational cost as well as classification accuracy in feature selection process [25].

Instruction level parallelism (ILP) is a measure of "how many" operations can be performed simultaneously and is represented using instructions per cycle (IPC). This parameter depends on several factors such as the issue width of the processor, branch prediction accuracy, number of resources, cache configuration (size, number of read/write ports, miss rate, etc.), and instruction dependencies. Figure 5 shows the IPC of the BCIBench applications for the simulated core. It shows the variation of the IPC values among BCIBench applications. The variations show the architectural implications of the instruction dependency and branch prediction accuracy for each program.

Overlap processing is very popular in real-time analysis. In this method, the processing window should be able to slide a variable amount from the current window. For example, if we assume 80% overlapping, five windows will process during one second. Thus, it is desirable to measure the throughput of a BCI signal processing flow. Figure 6 shows the throughput for BCIBench algorithms and applications. For an end-to-end application, throughput is measured as the number of trials that can be processed per second, and for a single algorithm it shows how frequently the core can process that algorithm in one second. In the BCI domain, throughput requirement depends on the application. For instance, gaming and entertainment applications require high throughput while the spell checker application has lower throughput requirement.

E. *Memory Characteristics*

Embedded systems typically use Reduced Instruction Set Architecture (RISC) processors which require higher memory due to low code density and load/store architecture of RISC processors. In an embedded system, memory system occupies a large portion of power budget and area. Therefore, in order to properly design low-power wearable systems, the memory demands of application must be served efficiently. Access time and energy per access are two key metrics in memory system of any application. However, in low-power wearable applications, data rate is low and due to power constraints, energy per access is more important. In a memory system, cache miss translates to a larger energy per access due to accessing slower and larger memories in the higher level. Figure 7 shows the miss rate for instruction cache. The instruction cache miss rate depends on the instruction sequence, e.g., number of jump and branches, branch predication accuracy and cache configuration. AR and BP algorithms have similar number of instructions, but due to the lower branch prediction accuracy in the AR, it has a higher instruction miss rate. Figure 8 shows the miss rate for L1 data cache. Data cache miss rate mainly depends on how the program accesses the data. Programs which access data with spatial and temporal locality have lower data cache miss rate. FIR and CAR are both preprocessing algorithms that operate on a matrix of data (rows correspond to various channels and columns correspond to samples acquired from each channel). FIR algorithm is applied to each channel individually, so it benefits from a suitable locality in data accesses. In CAR algorithm, it requires the current sample from all channels (i.e. accesses to the data matrix is column based and not row based). Therefore, data cache miss rate is higher than that of the FIR.

TABLE IV. BREAKDOWN ANALYSIS OF END-TO-END BCI APPLICATIONS.

Application	Algorithm	Power (mW)	Cycles per sample	Cycles per second (Millions)	Instruction per sample	Instruction per second (Millions)	L1D_acces per sample	Response time(Second)	Throughput	INT_ALU utilization (%)	FP_ALU utilization (%)	AGU utilization (%)	INT_ALU usage (Thousands)	FP_ALU usage (Thousands)	AGU usage (Thousands)
BCI1	CAR	2.76	188	0.77	64	0.26	36	0.0077	130	0.475	0.089	0.482	131	4	133
	BP	46.82	3175	13	1280	5.24	863	0.13	8	6.906	2.856	12.514	1903	131	3448
	LDA	0.01	1	0.003	1	0.001	1	0.0001	10000	0.001	0.001	0.002	0.4	0.05	0.5
	Total	**49.59**	**3364**	**13.77**	**1345**	**5.5**	**900**	**0.1377**	**7**	**7.382**	**2.946**	**12.998**	**2034**	**135**	**3581**
BCI2	FIR	4.33	293	1.20	101	0.41	102	0.012	83	0.183	0.317	0.681	85	25	317
	FFT	79.43	5387	22	1505	6.17	703	0.22	5	5.551	18.634	6.059	2583	1445	2820
	LDA	0.01	1	0.003	1	0.001	1	0.0001	10000	0.001	0.001	0.001	0.4	0.05	0.5
	Total	**83.77**	**5681**	**23.2**	**1607**	**6.58**	**806**	**0.232**	**4**	**5.735**	**18.952**	**6.741**	**2668**	**1470**	**3137**
BCI3	Lap	9.59	650	2.66	251	1.03	139	0.027	38	1.663	0.082	1.761	499	4	528
	AR	44.30	3004	12.3	1101	4.51	601	0.123	8	6.783	4.499	7.798	2035	225	2339
	MLP	0.11	7	0.03	1	0.01	1	0.0003	3333	0.018	0.006	0.018	5	0.3	5
	Total	**54**	**3661**	**14.99**	**1353**	**5.55**	**741**	**0.1503**	**7**	**8.464**	**4.587**	**9.577**	**2539**	**229**	**2873**
BCI4	FIR	4.33	293	1.20	101	0.41	102	0.012	83	3.152	5.464	11.737	85	25	317
	DS	0.52	35	0.143	14	0.06	8	0.0014	698	1.023	0.001	1.088	28	0.005	29
	LDA	0.02	1	0.003	1	0.001	1	0.0001	10000	0.024	0.019	0.036	0.6	0.09	1
	Total	**4.87**	**329**	**1.35**	**116**	**0.47**	**111**	**0.0134**	**74**	**4.199**	**5.484**	**12.861**	**114**	**25**	**347**

As mentioned earlier, in voltage scaling which is a promising method for power reduction, it is important to determine the activity level of the unit of interest to adjust the supply voltage accordingly. In the cache design, the average number of accesses per second is a measure of the activity level. Therefore, we measured data cache accesses per second for BCIBench applications as shown in Figure 9.

IV. ARCHITECTURE IMPROVEMENT

Regardless of the implementation platform, a benchmark suite can be used in order to optimize a design in terms of power and performance. In the design of wearable computers, a new trend that has recently become popular is to use a microcontroller along with a hardware accelerator to meet power constraints. In this method, the block which consumes the majority of power in the signal processing flow is detected. Then, a special purpose hardware accelerator is used to implement the block to reduce the application power consumption. Thus, it requires a breakdown analysis of individual algorithms in the processing flow. BCI signal processing flow may change by changing the target application. Therefore, developing a hardware accelerator for a specific algorithm restricts the design only to the underlying application. This motivates the designers to use reconfigurable hardware accelerator. These accelerators include some general functional units (FUs) and several specific FUs that can be configured using configurable interconnections.

According to Figure 1, a BCI application comprises three major blocks, including pre-processing, feature extraction and classification. We analyze each individual block of the BCI end-to-end applications for power, performance and memory behaviors. For this purpose, we use metrics such as the number of instructions, throughput, resource usage, and resource utilization. Table IV shows the breakdown analysis for the end-to-end BCI applications. According to the Intel document [27], Atom core consumes 3.6W power for 1GHz clock. We calculate power consumption of each algorithm according to its execution time ((cycles_per_second/1e+9)*3.6). Information reported in Table IV can be used for variety of purposes. For instance, one important application parameter is the sampling frequency that needs to be carefully selected based on the requirements of the algorithms. In Table IV, we investigate cycles per sample and instructions per sample as the complexity measures of processing one input sample. Therefore, one can use these measures to estimate computational complexity of each block of the BCI application for different sampling frequency, as well as different number of channels, processing window, etc. These measures can also be used to calculate the peak computational requirements of each algorithm to adjust the proper supply voltage. Table IV also shows power and performance bottlenecks of the signal processing flow. As shown for BCI1, 2, and 3, feature extraction (i.e. BP, FFT, and AR) is the most computationally expensive block while in BCI4, preprocessing (i.e. FIR) is the bottleneck. This investigation is a guideline for designers to be able to reduce power consumption of specific blocks.

Response time is the amount of time that the core takes to complete the processing. Number of accesses to L1 data cache per second is an estimate of frequency of usage for L1 data cache. Resource utilization is the percentage of the time that a functional unit is actually occupied, as compared with the total time that the unit is available for use (Atom core has two integer ALUs, two Floating-point units and two address generation units (AGUs)). The last three columns show the absolute usage of those six ALUs. As we can see from Table IV, the utilization of AGUs is 30% larger than that of floating-point ALUs and 65% larger than the utilization of integer ALUs that shows high memory activity in BCI applications. Therefore, a functional unit for address generation purpose can improve both the performance and power consumption of the architecture. Another observation is that the maximum utilization of functional units of Atom core by BCI applications is 18%. Therefore, embedded BCI applications do not require complex architectures with several ALUs. Therefore, architectures with single ALU in the data path will results in higher utilization and very low power consumption. An example of those architectures that might be potential architectures for BCI can be found in [28].

Parallelization and multi-core processing is an effective approach to reduce the power consumption for several applications [29]. However, for wearable embedded BCI

systems with low operating frequency, multi-core architecture may increase power consumption. Because, a small portion of power consumption goes to the processing in these applications. Thus, multi-core architecture which requires extra complexity for dispatching processing among cores and handles communications of cores, increases the power consumption. Also, due to low operating frequency of these applications, the operation frequency of cores will be consequently low that extra leakage power will dominate dynamic power reduction.

Our observations illustrate the implementation of BCI signal processing algorithms with Atom architecture. However, the benchmarks provided in this study may be analyzed in conjunction with other architectures of interest. We will continue to expand our investigations with other open-source low-power cores. The BCIBench suite will be available on our website and a link will be provided in the camera-ready version.

V. CONCLUSION

Wearable BCI applications are gaining popularity due to the advances in creating small electronic circuits and processing units, as well as easy-to-wear EEG electrodes. In this work, we presented BCIBench as a novel benchmarking suite for wearable BCI applications. BCIBench provided a range of BCI algorithms for pre-processing, feature extraction and classification to evaluate BCI wearable embedded architectures. We evaluated memory behavior and performance of BCIBench using MARSS simulator for Intel Atom core. Our analysis was a guideline for designers to be able to design efficient memory system and reduce power consumption of specific blocks. For further architecture improvement to meet power constraints of low power BCIs, we analyzed individual blocks of the signal processing flow. One can use these measures to estimate computational complexity of each block of the BCI applications with different parameters. It can also be used to measure the peak computational requirements of each algorithm to adjust the proper supply voltage.

VI. ACKNOWLEDGMENT

This work was supported in part by the TerraSwarm Research Center, one of six centers supported by the STARnet phase of the Focus Center Research Program (FCRP) a Semiconductor Research Corporation program sponsored by MARCO and DARPA.

REFERENCES

[1] G. Schalk, K. Miller, N. Anderson, J. Wilson, M. Smyth, J. Ojemann, D. Moran, J. Wolpaw, and E. Leuthardt, “Two-dimensional movement control using electrocorticographic signals in humans,” Journal of neural engineering, vol. 5, p. 75, 2008.

[2] B. Rebsamen, C. Teo, Q. Zeng, M. Ang Jr, E. Burdet, C. Guan, H. Zhang, and C. Laugier, “Controlling a wheelchair indoors using thought,” IEEE Intelligent Systems, pp. 18–24, 2007.

[3] A. Nijholt, D. P.-O. Bos, and B. Reuderink, “Turning shortcomings into challenges: Brain–computer interfaces for games,” Entertainment Computing, vol. 1, no. 2, pp. 85–94, 2009.

[4] Chin-Teng Lin et. al, “Development of Wireless Brain Computer Interface With Embedded Multitask Scheduling and its Application on Real-Time Driver's Drowsiness Detection and Warning,” Biomedical Engineering, IEEE Transactions on , vol. 55, no. 5, pp.1582,1591, 2008

[5] N. Verma et al., “A micro-power eeg acquisition soc with integrated feature extraction processor for a chronic seizure detection system,” JSSC, vol. 45, no. 4, pp. 804–816, 2010.

[6] M. Shoaib, N. Jha, and N. Verma, “A low-energy computation platform for data-driven biomedical monitoring algorithms,” in DAC. IEEE, 2011, pp. 591–596.

[7] B. Case, “Spec2000 retires spec92,” The Microprocessor Report, vol. 9, 1995.

[8] M. Guthaus, J. Ringenberg, D. Ernst, T. Austin, T. Mudge, and R. Brown, “Mibench: A free, commercially representative embedded benchmark suite,” in WWC. IEEE, 2001, pp. 3–14.

[9] C. Strydis, C. Kachris, and G. Gaydadjiev, “Impbench: A novel benchmark suite for biomedical, microelectronic implants,” in SAMOS. IEEE, 2008, pp. 82–91.

[10] C. Bienia, S. Kumar, J. Singh, and K. Li, “The parsec benchmark suite: Characterization and architectural implications,” in PACT. ACM, 2008, pp. 72–81.

[11] J. Clemons, H. Zhu, S. Savarese, and T. Austin, “Mevbench: A mobile computer vision benchmarking suite,” in IISWC., 2011, pp. 91–102.

[12] A. Bashashati, M. Fatourechi, R. Ward, G. Birch, “A survey of signal processing algorithms in brain–computer interfaces based on electrical brain signals,” Journal of Neural engineering, vol. 4, p. R32, 2007.

[13] P. Brunner, L. Bianchi, C. Guger, F. Cincotti, and G. Schalk, “Current trends in hardware and software for brain–computer interfaces (bcis),” Journal of Neural Engineering, vol. 8, p. 025001, 2011.

[14] C. Anderson, E. Stolz, and S. Shamsunder, “Multivariate autoregressive models for classification of spontaneous electroencephalographic signals during mental tasks,” Biomedical Engineering, IEEE Transactions on, vol. 45, no. 3, pp. 277–286, 1998.

[15] D. Allen and C. MacKinnon, “Time-frequency analysis of movement-related spectral power in EEG during repetitive movements: A comparison of methods,” Journal of neuroscience methods, vol. 186, no. 1, pp. 107–115, 2010

[16] T. Higuchi, “Approach to an irregular time series on the basis of the fractal theory,” Physica D: Nonlinear Phenomena, vol. 31, no. 2, pp. 277–283, 1988.

[17] B. Hjorth, “Time domain descriptors and their relation to a particular model for generation of eeg activity.” Computerized EEG analysis, pp. 3–8, 1975.

[18] F. Lotte, M. Congedo, A. Lécuyer, F. Lamarche, and B. Arnaldi, “A review of classification algorithms for EEG-based brain–computer interfaces,” Journal of neural engineering, vol. 4, p. R1, 2007.

[19] C. Neuper, G. Müller-Putz, R. Scherer, and G. Pfurtscheller, “Motor imagery and eeg-based control of spelling devices and neuroprostheses,” Progress in brain research, vol. 159, pp. 393–409, 2006.

[20] X. Gao, D. Xu, M. Cheng, and S. Gao, “A bci-based environmental controller for the motion-disabled,” Neural Systems and Rehabilitation Engineering, IEEE Transactions on, vol. 11, no. 2, pp. 137–140, 2003.

[21] L. Farwell, E. Donchin, “Talking off the top of your head: toward a mental prosthesis utilizing event-related brain potentials,” Electroencephalography and clinical Neurophysiology, vol. 70, no. 6, pp. 510–523, 1988.

[22] g.tec - guger technologies. http://www.gtec.at - GugerTechnologies

[23] S. Dongjin, M.C. Jose , M.R. Jan , Z. Elad, M Michel, “Neural dust: an ultrasonic, low power solution for chronic brainMachine interfaces”, arXiv:1307.2196 [q-bio.NC].

[24] A. Patel, F. Afram, S. Chen, and K. Ghose, “Marss: A full system simulator for multicore x86 cpus,” in DAC. ACM, 2011, pp. 1050–1055.

[25] A. Ahmadi, O. Dehzangi, and R. Jafari, “Brain-computer interface signal processing algorithms: A computational cost vs. accuracy analysis for wearable computers,” in BSN. IEEE, 2012, pp. 40–45.

[26] G. Reinman and N. Jouppi, “Cacti 2.0: An integrated cache timing and power model,” WRL Research Report, vol. 7, 2000.

[27] I. Inc. Atom processor, in download.intel.com/embedded/processors/prodbrief/324100.pdf.

[28] N. Ickes, Y. Sinangil, F. Pappalardo, E. Guidetti, and A. P. Chandrakasan, “A 10 pj/cycle ultra-low-voltage 32-bit microprocessor

system-on-chip," in ESSCIRC (ESSCIRC), 2011 Proceedings of the. IEEE, 2011, pp. 159–162.

[29] Y. Tanabe, M. Sumiyoshi, M. Nishiyama, I. Yamazaki, S. Fujii, K. Kimura, T. Aoyama, M. Banno, H. Hayashi, T. Miyamori, "A 464gops 620gops/w heterogeneous multi-core soc for image-recognition applications," IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2012, pp. 222–223.