Proceedings of the
8th Graz Brain-Computer Interface Conference 2019 DOI: 10.3217/978-3-85125-682-6-33

HETEROGENEOUS REAL-TIME MULTI-CHANNEL TIME-DOMAIN FEATURE EXTRACTION USING PARALLEL SUM REDUCTION ON GPU

J. Arnin, D. Kahani, H. Lakany, B.A. Conway[1]

[1] Centre of Excellence in Rehabilitation Engineering, Department of Biomedical Engineering, University of Strathclyde, Glasgow, United Kingdom

E-mail: jetsada.arnin@strath.ac.uk, b.a.conway@strath.ac.uk

ABSTRACT: Online BCI has become a fascinating field of research nowadays. One of the main challenges in this field is to reduce the latency caused by the computational complexity of the signal processing algorithms. This issue leads to difficulty in processing real-time data. Usually, a trade-off needs to be considered between the number of input samples and precision of the processing algorithms. In this paper, heterogeneous computing concept is investigated to alleviate the computational complexity occurred in real-time processing. An OpenCL was utilized to implement signal processing algorithms in parallel. Feature extraction methods including band power and statistical moments were selected to examine the power of heterogeneous computing using parallel sum reduction. As a result, varying the number of work-group sizes which is an essential parameter of parallel processing provided dissimilar computing times. Also, running at a higher sampling rate yielded a higher benchmark ratio between sequential and parallel. However, system optimization is still necessary when processing BCI in real time.

INTRODUCTION

Processing signal in real-time brain-computer interface (BCI) could usually encounter many difficulties ranging from hardware level to software level. One of the most challenging issues is the system latency [1] which may raise a major problem because this could lead to missing some important data such as an EEG component or an event under half-second. To minimize the latency by only optimizing sequential algorithms to reduce the processing time may not be enough to capture those components. The most common solution on the hardware level is to increase the speed of data transmission, buffering a sufficient amount of incoming data, and using fast processing units [2]. This may cost developer much money and a comprehensive technique.

Nowadays, high-performance computing (HPC) technology plays an important role in solving complex computational problems such as simulation, modeling, and analysis [3-4]. This technology does not only focus on developing faster hardware but the algorithms also [5]. According to the HPC, its concept is based on parallel computing for running application efficiently, reliably, and quickly [6]. To understand the concept of parallel computing, any sequential task can be split into a section which each is run separately on hardware acceleration. There is a lot of hardware acceleration available on the market that has a reasonable price such as consumer graphics processing unit (GPU) and a user-friendly field-programmable gate array (FPGA) [7]. These hardware units are programmable with their specific languages that may take much time to learn in a programming language. To resolve this issue, an open computing language (OpenCL) has been developed to overcome cross-platform programming [8]. It means that any hardware acceleration unit can be executed with one-time coding. So far, the OpenCL platform has become an industry standard for programming those hardware units [9]. In addition, certainly understanding heterogeneous computing concept which is the use of parallel processing techniques is essential and required when programming in OpenCL [10].

To demonstrate the heterogeneous computing concept for real-time signal processing, implementing in major processing steps and their bottlenecks was discussed in this paper. One of the most challenging BCI problems is running feature extraction algorithms in real time [11]. Since the number of processing channel is always much more than one or two, some complex features such as independent component analysis (ICA), autoregressive (AR) model, and discrete wavelet transform (DWT) are mostly implemented in an offline BCI [12]. However, these time-consuming features can be used in real-time processing with optimization that may limit the performance of the algorithms [13]. As an advantage of heterogeneous computing, these algorithms can be broken down into a smaller part and compute each part concurrently then concatenate to a final solution.

In this paper, we applied the commercial and open-source OpenCL technology into real-time signal processing which time-domain feature extraction methods including selective band power and statistical moments were selected to evaluate the computing performance. The system includes both EEG simulation and signal processing module. The proposed module offers up to 32 channels for real-time signal processing based on the heterogeneous computing concept. The archive EEG dataset was used to test the computing performance in real time with different sampling frequency acquired. The parallel computing time was compared to sequential processing approach as well.

 DOI: 10.3217/978-3-85125-682-6-33

MATERIALS AND METHODS

Latency analysis: Regarding the cause of computational latency in real-time processing, this can be divided into four categories, i.e., signal acquisition hardware, data transmission, types of application, and processing algorithms. Table 1 shows the comparison of the latency causes in terms of delay and versatility. The comparison was on the basis of cost-effectiveness and current technology. According to Table 1, the main cause of latency in a real-time BCI is computational complexity of the signal processing algorithms. To overcome this bottleneck, using different approach such as parallel processing instead of traditionally sequential method could more reduce the latency.

Table 1: Delay and versatility in signal processing

Category	Delay	Versatility
Signal acquisition	Low	Low
Data transmission	Medium	Low
Types of application	Medium	Low
Processing algorithms	High	High

EEG dataset: The archive EEG, a collection of 32-channel data from 14 subjects (7 males, 7 females), provided by Swartz Center for Computational Neuroscience [14] was used to evaluate the performance of our parallel design. According to the data, the participants were asked to perform a go-nogo categorization task and a go-no recognition task on natural photographs displayed every 20 milliseconds. The experiment ran a total of 2500 trials on each participant. Note that as the archive data was sampled at 1000 Hz with a specific amplifier but for the purpose of full usage, the data were regenerated and rectified to appropriately match with the voltage range of the analog output device.

Simulation system: This study was developed based on Qt platform (Qt 5.12 LTS) using C++ programming language which can integrate the OpenCL and related libraries together. The simulation system consists of a signal generator and signal acquisition. The archive dataset was generated waveform through the 32-channel analog output device (NI PCIe-6738) and then fed back into the 32-channel analog input device (NI PCIe-6343) using RG58 50-Ohm coaxial cables. For the output device, each channel was generated at the sampling frequency of 1kHz according to the dataset. Note that the output resolution is 16 bits with voltage range of ±10V. For signal acquisition, the sampling rate was varied, including 128, 256 512, 1024, and 2048 Hertz.

Heterogeneous signal processing: According to the general signal processing pipeline, it is frequently processed in sequential approach. This may result in a delay when loads of processing steps are added. Using heterogeneous computing concept in this problem can decrease the latency dramatically. In this paper, the calculation part of the feature extraction on each channel was processed separately and concurrently by multiple compute units. Fig. 1 demonstrates the overall system of the heterogeneous feature extraction which each channel processes simultaneously.

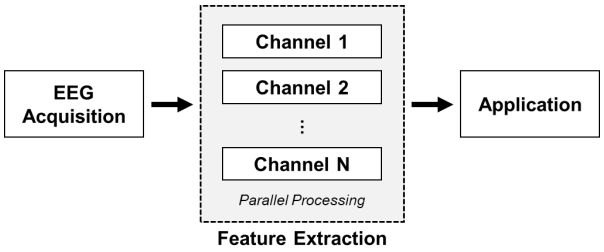

Figure 1: The overall system of heterogeneous feature extraction for real-time BCI.

OpenCL initialization: As an advantage of the cross-platform parallel programming, the graphics card is the easiest unit to be used with OpenCL. The AMD graphic cards (Radeon™ Pro WX 7100) were used to deploy the computation based on the OpenCL 2.0 which the shared virtual memory technique was introduced [15]. The shared virtual memory can reduce the latency of transferring data between the host and devices. Setting up the number of work items manually split into global and local to yield the best computing result. Fig. 2 shows an overview of the structure of the OpenCL 2.0 platform used in the study.

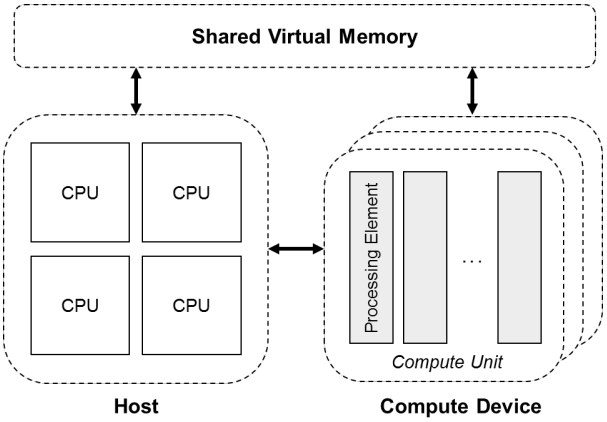

Figure 2: The overview of the structure of the OpenCL 2.0 platform.

Feature extraction: Many familiar feature extraction methods have been developed for processing BCI such as discrete Fourier transform (DFT) or power spectral density (PDS) and wavelet transform (WT) based which are based on frequency analysis. These methods are considered to be the most effective techniques for dealing with time-varying EEG signals. Regarding the time-domain analysis, the most commonly used method for EEG feature extraction is selective band power which is the average power of a signal in a specific frequency range. In this paper, the band power was used as a feature for event detection. Besides the band power, another time-domain method named statistical moments is also used to evaluate. The statistical moments are specific quantitative measurements in time domain

 DOI: 10.3217/978-3-85125-682-6-33

analysis. The general formula of the n-th order statistical moments was described in the literature [16-17] which mean, variance, skewness, and kurtosis are mostly used for feature extraction. In this paper, the first moment and second moment which are mean and variance were calculated concurrently on each channel.

Parallel sum reduction: To achieve the highest performance from the heterogeneous computing concept, as the calculation of band power and statistical moments mostly uses a summation, this can be managed by using a parallel sum reduction technique [18]. This technique maximizes the performance of compute unit by copying values from global memory into a local memory of the same work-group. Then each work-group processes its local work-item concurrently that is partitioning the whole summation into a small summation and finalize when all work-items finished their own tasks. Fig. 3 shows the concept of parallel sum reduction which introduces the use of local memory to store each element concurrently and then reduce to half by using a stride. Note that in the OpenCL 2.0 parallel sum reduction is integrated into a workgroup function so there is no need to write a nested loop to calculate the summation.

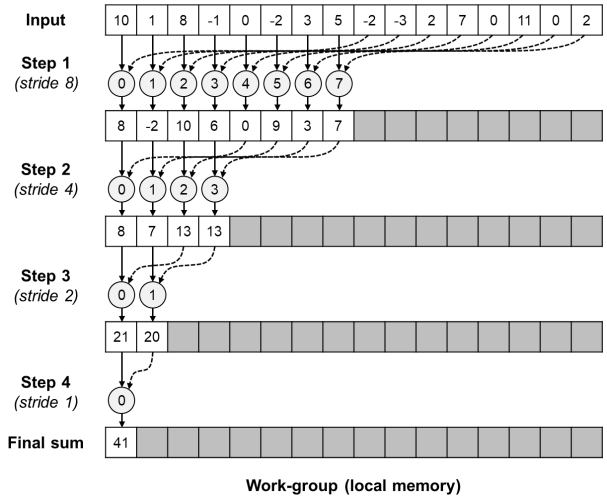

Figure 3: The concept of parallel sum reduction.

Performance improvement: In order to improve the computational speed, fine-tuning parameters for parallel sum reduction is required. As the appropriate number of input data for sum reduction should be a power of 2, we decided the local work-item at 16, 32, 64, and 128, respectively. Conversely, using sum reduction inside the work-group function of the OpenCL 2.0, the input number is not necessary to follow the power of 2.

Benchmarking: The average computation time of a full command execution running on the GPU each local work-item are recorded for 100 times and compared to the result from sequential computing. Note that the full execution starts from transferring data from host to device, processing the kernel, and transferring data back to the host. This study ran on 64-bit Window 10 OS, with 32-GB DDR4 and Intel Xeon E5-1630 v4.

RESULTS

Feature extraction methods including band power and statistical moments were examined in real-time signal processing using the OpenCL platform. The execution time and benchmarking of sequential processing and parallel processing approach were reported in this section. Fig. 4 illustrates the execution time in microsecond when the band power feature was performed. Regarding the result, the execution time was related to the number of a processed sample which higher number required more processing time. With a modification of the number of work-group sizes, a large number of work-group size provided the better performance which the computing time was reduced. In addition, Fig. 5 shows the benchmarking of sequential processing and parallel processing on the same compute device. The ratio was calculated from the execution time of the sequential approach divided by the execution parallel approach. According to the benchmarking result, the higher the sampling rate set, the higher the ratio received. Apart from the band power, using statistical moments also provided likely an identical result. Fig. 6 presents its execution time at different numbers of input. Interestingly, adjusting work-group size had an impacted on speed especially at 256Hz and 512Hz. Fig. 7 also showed the ratio as explained previously.

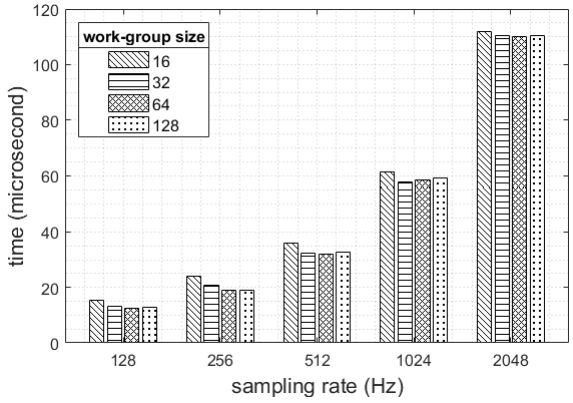

Figure 4: Execution time of band power feature.

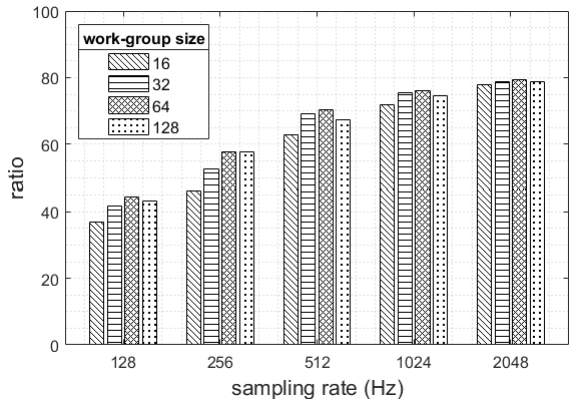

Figure 5: Benchmarking of band power feature.

 DOI: 10.3217/978-3-85125-682-6-33

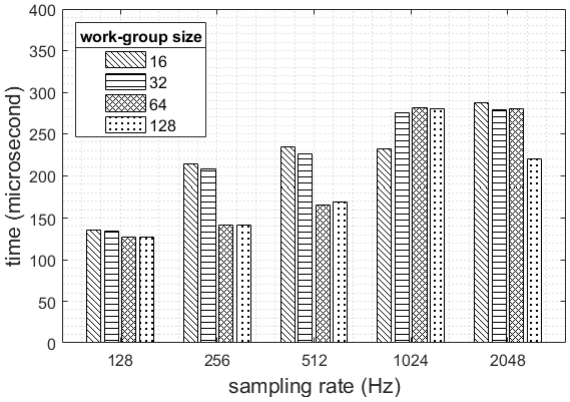

Figure 6: Execution time of statistical moments feature.

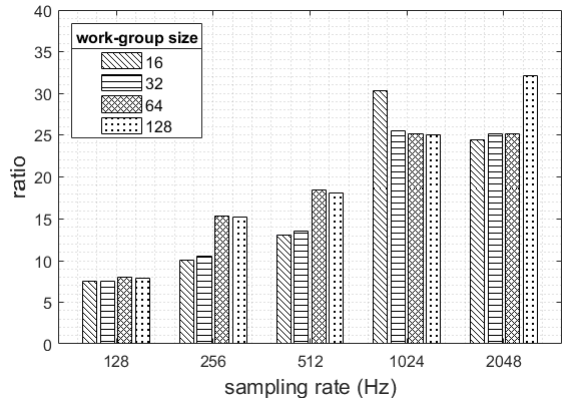

Figure 7: Benchmarking of statistical moments feature.

DISCUSSION

According to the results, the latency of feature extraction was relatively small and appropriately enough for real-time processing. For example, at 1kHz of the sampling rate, band power feature was about 0.06 milliseconds while the statistical moments method was roughly 0.25 milliseconds. Fascinatingly, running data at 2kHz with band power method achieved the execution time about 0.11 milliseconds which is much higher than running at 1kHz (nearly double time) whereas statistical moments method yielded the slightly identical result at 1kHz. It is to be observed that running on different processors and environments may yield different computing times.

Remarkably to the benchmarking results, the reduction technique provided a higher ratio when the size of the input was larger. Besides, increasing the number of work-group size from 16 to 64 provided almost the higher ratio for both features. While setting a work-group size at 128, the ratio dropped slightly. This is because it is allowed enough times for the compute unit to initiate internal parameters (hardware level) and then process data continuously and efficiently as discussed in the previous study [19]. Therefore, a tradeoff between the number of input samples and the acceptable latency in the system should be considered. Note that the GPU used in the study has the maximum work-group size of 256 work-items on each dimension.

Regarding the sum reduction, it can be applied to other time-consuming algorithms such as frequency-domain analysis like DFT and WT. These multiple computing steps can be separated into multiple kernels and directly execute from device side without any request commands from the host side as introduced in the new features of OpenCL 2.0. This technique has also been implemented into the statistical moments which there were two kernels, i.e., one for mean calculation and another for variance calculation, running concurrently using shared virtual memory. Not only running on the GPU, by using the same OpenCL program the project can be run on other systems such as FPGAs. This is expeditious and required only minor parameters adjustment.

With regard to the latency analysis, some limitations can be resolved but have to tradeoff between time- and cost-effectiveness. Furthermore, an OpenCL library for signal processing could help researchers to gain the most benefit from heterogeneous computing because in this study we have developed all steps from the beginning including setting up complicated parameters such as initializing a platform and a context.

CONCLUSION

This study presents the use of heterogeneous computing technique to implement into BCI processing. As a result, speeding up the computation by using a parallel processing scheme is possible and flexible for real-time computing. To reduce the system latency, optimization of both hardware and software should be considered when using in such real-time applications. Lastly, an OpenCL library could help researchers to reduce the developing time for the BCI applications which is the next step of our work.

ACKNOWLEDGEMENTS

The authors would like to acknowledge the support from Scottish Government Health Directorates and the Royal Thai Government scholarship. The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

REFERENCES

[1] Xu R, Jiang N, Lin C, Mrachacz-Kersting N, Dremstrup K, Farina D. Enhanced low-latency detection of motor intention from EEG for closed-loop brain-computer interface applications. IEEE Transactions on Biomedical Engineering. 2014;61(2):288-96

[2] Oweiss KG. A systems approach for data compression and latency reduction in cortically controlled brain machine interfaces. IEEE Transactions on Biomedical Engineering. 2006;53(7):1364-77

[3] Gulo CA, Sementille AC, Tavares JM. Techniques of medical image processing and analysis accelerated by high-performance computing: A systematic literature review. Journal of Real-Time Image Processing. 2017;16:1-8
[4] Xu J, Huang E, Chen CH, Lee LH. Simulation optimization: A review and exploration in the new era of cloud computing and big data. Asia-Pacific Journal of Operational Research. 2015;32(03):1550019
[5] Wang T, Kemao Q. Parallel computing in experimental mechanics and optical measurement: A review (II). Optics and Lasers in Engineering. 2018;104:181-91
[6] Pratx G, Xing L. GPU computing in medical physics: A review. Medical physics. 2011;38(5):2685-97
[7] Brodtkorb AR, Hagen TR, Sætra ML. Graphics processing unit (GPU) programming strategies and trends in GPU computing. Journal of Parallel and Distributed Computing. 2013;73(1):4-13
[8] Munshi A, Gaster B, Mattson TG, Ginsburg D. OpenCL Programming Guide, Pearson Education (2011)
[9] Stone JE, Gohara D, Shi G. OpenCL: A parallel programming standard for heterogeneous computing systems. Computing in science & engineering. 2010;12(3):66
[10] Gaster B, Howes L, Kaeli DR, Mistry P, Schaa D. Heterogeneous computing with openCL: revised openCL 1, Newnes (2012)
[11] Tangermann M, et al. Review of the BCI competition IV. Frontiers in neuroscience. 2012;6:55
[12] Lotte F, et al. A review of classification algorithms for EEG-based brain–computer interfaces: a 10 year update. Journal of neural engineering. 2018;15(3):031005
[13] Al-Fahoum AS, Al-Fraihat AA. Methods of EEG signal features extraction using linear analysis in frequency and time-frequency domains. ISRN neuroscience. 2014
[14] Delorme A, Rousselet GA, Mace MJ, Fabre-Thorpe M. Interaction of top-down and bottom-up processing in the fast visual analysis of natural scenes. Brain research Cognitive brain research. 2004;19(2):103-13
[15] Kaeli DR, Mistry P, Schaa D, Zhang DP. Heterogeneous Computing with OpenCL 2.0, Morgan Kaufmann (2015)
[16] Soliman SS, Hsue SZ. Signal classification using statistical moments. IEEE Transactions on Communications. 1992;40(5):908-16
[17] Hjorth B. EEG analysis based on time domain properties. Electroencephalography and clinical neurophysiology. 1970;29(3):306-10
[18] Zaki MJ, Parthasarathy S, Ogihara M, Li W. Parallel algorithms for discovery of association rules. Data mining and knowledge discovery. 1997;1(4):343-73
[19] Fang J, Varbanescu AL, Sips H. A comprehensive performance comparison of CUDA and OpenCL, in Proc. International Conference on Parallel Processing, 2011, 216-225