Low-Power Manycore Accelerator for Personalized Biomedical Applications

Adam Page
Dept. of Computer Science & Electrical Engineering
University of Maryland, Baltimore County
apage2@umbc.edu

Nasrin Attaran
Dept. of Computer Science & Electrical Engineering
University of Maryland, Baltimore County
attar1@umbc.edu

Colin Shea
Dept. of Computer Science & Electrical Engineering
University of Maryland, Baltimore County
cshea3@umbc.edu

Houman Homayoun
Dept. of Electrical & Computer Engineering
George Mason University
Fairfax, VA
hhomayou@gmu.edu

Tinoosh Mohsenin
Dept. of Computer Science & Electrical Engineering
University of Maryland, Baltimore County
tinoosh@umbc.edu

ABSTRACT

Wearable personal health monitoring systems can offer a cost effective solution for human healthcare. These systems must provide both highly accurate, secured and quick processing and delivery of vast amount of data. In addition, wearable biomedical devices are used in inpatient, outpatient, and at home e-Patient care that must constantly monitor the patient's biomedical and physiological signals 24/7. These biomedical applications require sampling and processing multiple streams of physiological signals with strict power and area footprint. The processing typically consists of feature extraction, data fusion, and classification stages that require a large number of digital signal processing and machine learning kernels. In response to these requirements, in this paper, a low-power, domain-specific manycore accelerator named Power Efficient Nano Clusters (PENC) is proposed to map and execute the kernels of these applications. Experimental results show that the manycore is able to reduce energy consumption by up to 80% and 14% for DSP and machine learning kernels, respectively, when optimally parallelized. The performance of the proposed PENC manycore when acting as a coprocessor to an Intel Atom processor is compared with existing commercial off-the-shelf embedded processing platforms including Intel Atom, Xilinx Artix-7 FPGA, and NVIDIA TK1 ARM-A15 with GPU SoC. The results show that the PENC manycore architecture reduces the energy by as much as 10X while outperforming all off-the-shelf embedded processing platforms across all studied machine learning classifiers.

CCS Concepts

•**Computer systems organization** → **Architectures;** •**Computing methodologies** → *Parallel computing methodologies; Machine learning algorithms;* •**Applied computing** → *Bioinformatics;*

GLSVLSI '16, May 18-20, 2016, Boston, MA, USA

DOI: http://dx.doi.org/10.1145/2902961.2902986

Keywords

Low power; biomedical; manycore; accelerator; digital signal processing; machine learning; FPGA; embedded processors.

Sensor	# Channels	Sampling Freq.	Description
HR	1 - 2	100 Hz	Heart Rate
SpO2	1 - 2	0.2 - 0.5 kHz	Blood Oxygen
ECG	3 - 12	0.2 - 1.0 kHz	Heart Elec Act.
EEG	6 - 64	0.1 - 1.0 kHz	Scalp Elec. Act.
GSR	1 - 4	50 - 100 Hz	Skin Conductance
EMG	4 >	1 - 2 kHz	Muscle Elec. Act.
RESP	1 - 3	50 - 100 Hz	Respiration

Table 1: Example sensors for biomedical applications with typical number of channels and sampling frequencies. Demonstrates the various sampling frequencies and multiple channels.

1. INTRODUCTION

Personalized biomedical applications primarily consist of three basic stages: 1. a sensor front-end to capture and digitize physiological signals, 2. a processing stage to analyze, classify, and potentially store the sensors' data, and 3. a RF module stage to transmit the data, classification, and/or diagnostics to the user or medical personnel [13, 16, 17, 10]. There has been an incredible amount of innovation and improvement in sensor design that has dramatically reduced power while maintaining high accuracy. This is the result of technologies such as MEMS sensors and specialized AFEs targeted for physiological signals, such as Texas Instruments medical AFEs like ADS129x and AFE44xx. There has also been a tremendous amount of work done on wireless RF modules ranging from specialized research modules to commercial modules such as Bluetooth Smart (17.9mA RX, 18.2mA TX, 1uA sleep). Still the relatively high amount of power required to transmit raw or even compressed data, makes it essential to perform local onboard processing [5]. The enterprise of this work is on the processor architecture that addresses the unique challenges and characteristics of biomedical applications. Most previous works have focused on modifications to existing microcontrollers/processors by reducing the instruction set architecture (ISA) or creating an SoC with specialized accelerator cores targeted for particular biomed-

ical applications [7, 4, 15]. These approaches are often very expensive and require long development time to develop specialized chips. Furthermore, these modifications do not target the fundamental characteristics in-common with a majority of biomedical applications. Besides the major restrictions on power and area, the processor must be able to efficiently process several physiological signal streams. Table 1 provides some example sensors with typical number of channels and sampling frequencies. Processing the streams often includes feature extraction, data fusion, and classification stages that consist of both digital signal processing (DSP) and machine learning (ML) kernels that exhibit task-level and data-level parallelism.

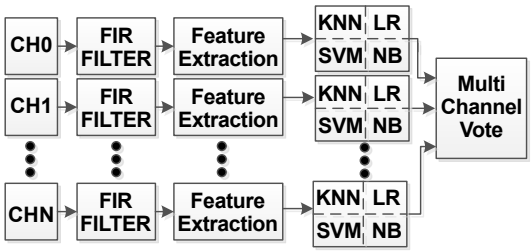

Figure 1: Block diagram of a parallel multi-channel seizure detection application containing feature extraction, classifier, multi-channel vote, and IO interface. The application also shows inherent data-level and task-level parallelism.

2. BIOMEDICAL APPLICATIONS

Among the many commonalities shared between personal biomedical applications, the need to process parallel streams of data in real-time is a dominating feature. The analysis of these multiple streams requires a mix of data-level and task-level parallel computation [3, 6, 2, 1]. In addition, these applications often require a large number of digital signal processing (DSP) and machine learning (ML) techniques. DSP is often used to extract useful representations of the input data while machine learning is needed to perform automated classification for diagnostic and detection purposes. In order to demonstrate these dominant commonalities, we investigated various common DSP and ML kernels. The examined DSP kernels include filtering (FIR), windowing, Fourier transform (FFT), and compressive sensing (CS), while the examined ML kernels include logistic regression (LR), naive Bayes (NB), support vector machine (SVM), and k-nearest neighbor (KNN) [6]. In addition to exploring these various DSP and ML kernels, a case study of seizure detection was also implemented on a number of hardware platforms. The main block diagram of the seizure detection application is presented in Fig. 1 [13, 12]. This case study is an ideal example of a biomedical application that exhibits multiple streams (up to 24 EEG channels) of real-time data that must be processed with DSP and ML kernels. In addition, the multiple streams allow for intuitive parallel processing. Each streams pipelined stages provide task-level parallelism, while within each stage there exists a large degree of potential data-level parallelism. Discussion of the case study along with comparison results are presented in Section 5.

3. PROCESSORS AND PLATFORMS

3.1 Off-the-shelf Processors

In order to study how embedded off-the-shelf processors can efficiently process biomedical applications, we conducted several experiments on two state-of-the-art low power architectures, Atom and ARM, which are substantially different in their core micro-architecture, memory subsystem, and instruction set architecture (ISA). ARM is using ARMs Thumb whereas ATOM is using Intel X86. A recent work has shown that each of these architectures offers a different power and performance trade-off for various applications [15, 11]. Although easy to program, these processors have limited flexibility and parallelism. Therefore, a field-programmable gate array (FPGA) is also explored which provides high flexibility but requires writing low-level logic.

3.1.1 ATOM

The Intel Atom is an ultra low-power processor architecture mainly used in mobile and portable platforms. In this paper we conduct our study on the Intel Edison platform, a system on chip (SoC), containing an IA-32 low-power dual core processor, 1 GB of DDR3, Wi-Fi, Bluetooth, and 4 GB eMMC. The cores are Intel's Silvermont architecture providing a standard x86 architecture, streaming SIMD extension (SSE) and running at a fixed clock of 500MHz.

3.1.2 ARM A15 CPU/NVIDIA K1 GPU

NVIDIA's Jetson TK1 is a System-on-Chip (SoC) combining the Kepler graphics processing unit (GPU) and a 4-Plus-1 ARM processor arrangement. The 4-plus-1 processor configuration, also known as variable Symmetric Multiprocessing (vSMP), consists of five Cortex A15 ARM processors, four high performance and one low power processor. Each ARM A15 CPU has a 32KB L1 data and instruction cache supporting 128-bit NEON™ general-purpose single instruction and SIMD instructions. All processors configuration have shared access to a 2MB L2 cache. The K1 GPU is available to either processor power configuration and consists of a single Streaming Multiprocessor (SMX). The SMX has a CUDA™ compute capability of 3.2, which provides a majority of the architectural benefits of the K20c only scaled down to a single SMX group. The Jetson TK1 has 2GB of DDR3 memory that is shared between the CPU and GPU and is rated up to 933MHz. In this paper we explored different combinations of the four high performance Cortex A15 ARM processors and the GPU for comparison.

3.1.3 FPGA

Field-programmable gate arrays (FPGA) are highly flexible allowing on-the-fly configuration to optimize bit resolution, clock frequency, and parallelization for a given application. In addition, modern FPGAs provide accelerators to boost the performance for operations such as multipliers, generic DSP cores, and embedded memories. The main disadvantages of FPGAs, however, are that they have substantially higher leakage power and require writing low level logic blocks. In this paper, we utilize Xilinx low-power Artix-7 FPGA platform (xc7a200tlfbg484). A very efficient implementation of the seizure detection application with different machine learning classifiers has previously been implemented [13]. This implementation is optimized on the bit-width resolution in addition to the level of parallelism and pipelining to meet the application deadline.

3.2 Domain-Specific PENC Manycore Accelerator

PENC manycore accelerator has a homogeneous architecture that consists of in-order processors with a 6 stage pipeline, a RISC-like DSP instruction set and a Harvard memory model [1, 2, 6, 9, 8]. The core operates on a 16-bit data-path with minimal instruction and data memory suitable for task-level parallelism. Furthermore, the cores have a low complexity, limited instruction set to further reduce area and power footprint. A processor can support up to 128 instructions, 128 data memory, and provides 16 quick-access registers. In the network topology, a cluster consists of three

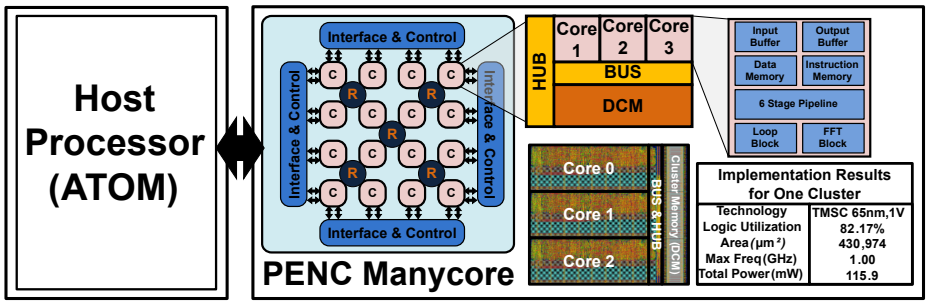

Figure 2: Block diagram of the PENC manycore acting as a coprocessor to the Intel Atom host. The manycore consists of 16 clusters (192 cores) routed through hierarchical routers. On the right, there is a detailed view and ASIC layout of a cluster which contains 3 lightweight cores, a shared bus and memory.

cores that can perform intra-cluster communication directly via a bus and inter-cluster communication through a hierarchical routing architecture. Each cluster also contains a shared memory. Figure 2 shows the block diagram of a 16 cluster version of the design, highlighting the processing cores in a cluster. Each core and router was synthesized and placed and routed in a 65 nm CMOS technology. Our manycore development environment and simulator provides cycle accurate results including completion time, instructions, and memory usage per core. The simulator also breaks down the instructions into groups such as ALU, branch, and communication.

This manycore architecture is ideally suited for most biomedical applications as it addresses the inherent characteristics and challenges. As previously discussed, biomedical applications process a number of physiological signals. Each signal can be processed in parallel in different designated clusters. Dynamic voltage and frequency scaling (DVFS) of the clusters can be used to align the sensor computation completion time, which can dramatically reduce energy usage. The lightweight cores also help to ensure that all used cores are fully utilized. While the lightweight cores are ideal for DSP kernels that require minimal static data [1, 2], ML kernels often require larger amounts of memory for their model data. This is addressed with the cluster-level shared memory that is interfaced to the bus. The shared memory can be accessed within the cluster using the bus and from other clusters through the router. The next section provides empirical results showing how these manycore features are well suited for biomedical applications.

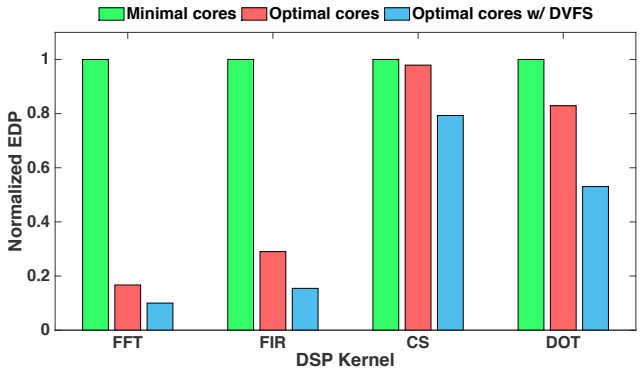

Figure 3: Mappings of DSP kernels including fast Fourier transform (FFT), FIR filter (FIR), compressive sensing (CS), and dot-product (DOT). First mapping uses the minimum cores needed. The second mapping utilizes maximum cores to leverage parallelism. The third is the same as second but with scaled frequency and voltage (DVFS).

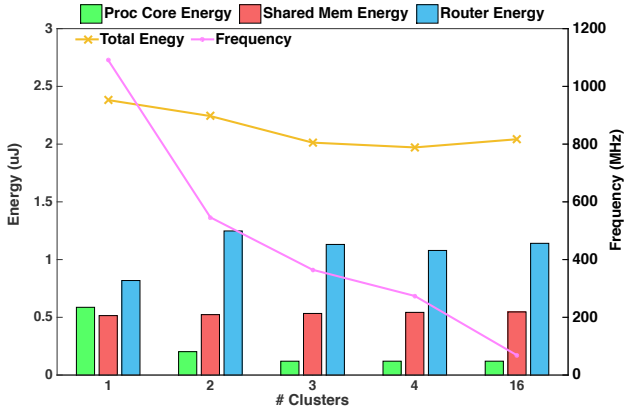

Figure 4: Comparison of different mappings of k-nearest neighbor (KNN) kernel on PENC manycore with voltage and frequency scaling to meet deadline. Additional clusters can be utilized to exploit KNN parallel structure allowing to reduce frequency and voltage. Energy dissipation (w/ core, router and memory breakdown), frequency, and voltage values are shown.

4. MANYCORE ANALYSIS

In order to demonstrate the manycore's effectiveness at targeting low-power biomedical applications, experiments were performed that highlight the unique characteristics of these applications. Specifically, the experiments map various DSP and ML kernels with performance measured in energy, execution time, and memory demands.

4.1 DSP Kernel Mapping with DVFS

In the first experiment, various digital signal processing kernels were mapped onto the manycore. The DSP kernels include fast Fourier transform (FFT), finite impulse response filter (FIR), compressive sensing (CS), and dot-product operation (DOT). An initial mapping was performed that used the minimum number of cores to act as a baseline. A second mapping was then performed that used the optimal amount of cores. This was done by selecting the best from a number of implementations. The final mapping is equivalent to the second mapping but scales the voltage and frequency to meet the execution time of the first mapping. Figure 3 shows all three of these mapping for the four DSP kernels with their corresponding energy-delay product (EDP). The plot shows that the manycore is able to efficiently parallelize all of the kernels and is able to achieve an energy reduction of up to 80%.

4.2 ML Kernel Mapping with DVFS

Many digital signal processing kernels require very little static and dynamic memory. For example, an 128-point FFT requires

around 512 words of memory assuming twiddle factors are precomputed and the input is complex. On the other hand, many machine learning kernels can often require storing a large volume of model data. For example, k-nearest neighbor essentially requires storing all of the training data. This could correspond to thousands of values needing to be stored. This is accommodated for by having cluster-level shared memory accessible through the cluster's bus. Similar to the mappings of the DSP kernels, the mapping of a ML kernel onto the manycore is performed. The k-nearest neighbor algorithm with 3,000 model data is mapped using between 1 and 16 clusters. The results are depicted in Fig. 4. As can be seen, increasing the number of clusters to map KNN allows the frequency to be dramatically reduced. The optimal mapping is obtained using 4 clusters which was able to reduce energy by 14% and execution time by 30% compared to single cluster.

5. CASE STUDY COMPARISON

5.1 Seizure Detection Overview

Epilepsy is a common neurological disease that affects approximately 2.2 million Americans. According to a recent Institute of Medicine report, epilepsy is the 4th most common neurological disorder in the US with roughly 1 in 26 people being diagnosed with epilepsy in their lifetime [1]. The ability to monitor epileptic patients in an ambulatory setting is a crucial tool that has significant medical, psycho-social, cost, and safety advantages. For example, this device could help determine the minimal effective dosage as well as alert medical personnel when a seizure is detected to reduce the occurrences of sudden unexpected death in epilepsy (SUDEP).

In our previous work, a flexible seizure detection hardware system was implemented to detect the onset of a seizure by analyzing multiple channel, scalp-based EEG data in real time [13, 12, 5]. The developed system is capable of processing up to 24 channels of EEG electrodes that are digitized using specialized AFE ICs. Each stream of EEG sensor data is sampled at a rate of 256 Hz with 16-bit resolution. The processing consists of 4 main stages as previously shown in the diagram in Fig. 1. The EEG sensor data is first passed through a filter to remove high frequency and DC components. A feature extraction stage is then used to convert windows of time-series data into 5 temporal features per EEG channel. Each channel's features are then classified using one of four classifiers: k-nearest neighbor (KNN), support vector machine (SVM), naive Bayes (NB), or logistic regression (LR). A final stage is then used to produce a final decision based on a multi-channel voting scheme. For our study, the windows consist of 256 samples (1 second) with 50% overlapping windows. This means that a window will contain half-second of new data, which gives a 500 ms deadline to process each window. Figure 5 depicts the task graph for each classifier variant of the seizure detection system. The task graphs highlight the parallelism that exists within the stages and between the EEG channels.

5.2 Platform Implementation

In this comparison, three platforms were use to implement and evaluate 5 processor combinations. Great care was taken to ensure optimal implementation and consistency across all combinations.

5.2.1 Intel Edison

The Intel Edison platform was used to gather results for the Intel Atom processors and is the purposed interface with the PENC. When using the Atom processors, great care was done to efficiently utilize the low power x86 cores. The software was written as a single program, multiple thread (SPMT) execution model. With in each thread the math operations were optimized for single instruction, multiple data (SIMD) execution. This was accomplished by explicit exploitation of parallelism through multithreading across the EEG channels, where the total number of channels is divided equally between available cores. The GCC/G++ compiler was passed architecture appropriate flags to increase the compilers effort on performance such as -O3, mtune=native, and specification of the floating point unit of SSE 4.2. To enhance the SIMD further, Intel® Performance Primitives (IPP) were used when the compiler could not vectorize or correctly map the functions to SIMD instructions. When the manycore is interfaced as an accelerator, the Edison is used as the host to perform data marshalling. In this case, the system toggles between active mode to transfer window of EEG data and sleep mode otherwise.

5.2.2 NVIDIA Jetson TK1

The NVIDIA® Jetson TK1 platform was used to obtain results for the ARM A15 CPU processors as well as with the embedded GPU as an accelerator. The application development and testing is similar to [14]. The same application architecture used for the Intel Edison was employed for the ARM A15 CPUs. With a combination of SPMT for partitioning groups of channel processing between the multiple core configurations. Each thread also targets the use of SIMD through ARM's NEON architecture. As there is no commercial optimizations for ARM's Neon SIMD, these functions are accelerated using the ARM Ne10 project. When using the GPU, the CPU cluster only has two cores active. One is used to dispatch kernels to the GPU and perform data marshalling. Another A15 core was used to service the OS. The application architecture utilizes existing CUDA libraries, cuBLAS and CUDA Math Libraries, for the KNN, SVM, NB, and LR kernels. In order to extract the maximum amount of parallelism, by allowing for kernel overlap, from the implementations, each operation within uses up to sixteen CUDA streams.

5.2.3 Xilinx Artix-7

For the Artix-7 FPGA, the entire system was implemented in Verilog. Xilinx's proprietary synthesizer was used to generate the register transistor language (RTL) as well perform the optimal translation onto the Artix-7 FPGA. When synthesizing, a balance design approach was used to balance area, power, and latency. The classifier model data was also stored using the FPGAs block random access memories (BRAM). A number of designs were explored that looked to optimize in the form bit resolution, parallelism, and pipelining. Further details can be found in [12].

5.3 Experimental Setup

Consistency had to be ensured when collecting power and timing results of the platforms. The execution times for all implementations were found by averaging thousands of processed windows of EEG data. For the Intel Edison and Jetson TK1 platforms, the power consumption of the entire system was captured using TI INA219 voltage and power meter, which is shown in Fig. 6. As previously discussed, power and cycle accurate results for the manycore were obtained in simulation. The power metrics for the manycore are based on Cadence projections of the standard-cell layout of the cores and routers. The required power of the manycore when processing was added to the Intel Edison power which consisted of both active and sleep states. For the Artix-7 FPGA implementation, the power and timing results were obtained using Xilinx XPower and Timing analyzer, respectively. Since this only includes power dissipation of the chip, the power usage of the Intel Edison is also added when the Atom processor is idle.

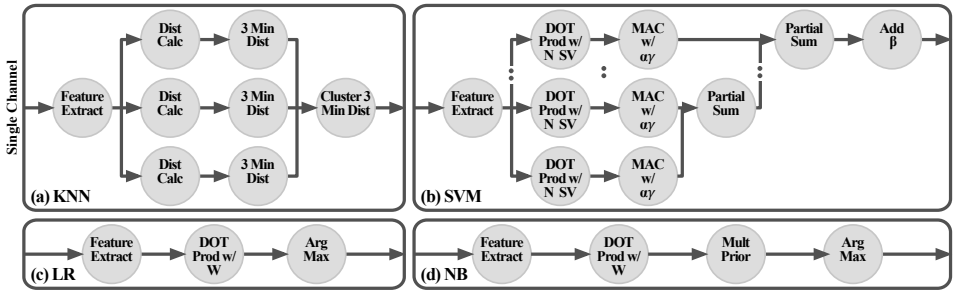

Figure 5: Task graphs of each classifier variation of the seizure detection case study. The task graphs highlight the task-level parallelism and interconnect between feature extraction and classification stage for single channel.

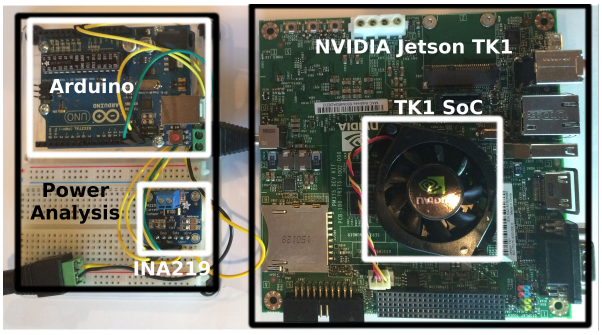

Figure 6: Experimental setup to obtain power and execution time measurements for TK1 using TI INA219 and Arduino. For this platform both ARM A15 cores & embedded GPU are targeted. The embedded GPU is used as an accelerator.

Design	KNN	SVM	NB	LR
Logic Slices	3788	4281	2987	2583
Memory (Kb)	2916.4	828.4	0.26	0.30
Max Freq (MHz)	131	152	101	134
Latency (cycle)	644,622	108,665	264	242
Nominal Freq (KHz)[1]	1286.01	217.33	0.53	0.48
Dynamic Power (uW)[1]	2697.76	192.96	0.171	0.098
Leakage Power (mW)	122	122	122	122
Dynamic Energy (uJ)[2]	1348.9	96.5	0.09	0.05
Total Energy (mJ)	62.35	61.10	61.00	61.00

Table 2: Artix-7 performance for seizure detection case study consisting of 5 features/channel and classifier. 1) The power results are for Nominal frequency to meet 0.5 sec interval, 2) The FPGA has large leakage power dominant compared to dynamic power, both dynamic and total results are presented.

5.4 Results

The FPGA platform is the most similar to the manycore in that it allows for the most flexibility in scaling frequency and targeting the data-level and task-level parallelism. Unfortunately, FPGA suffers from having very high leakage power. The results of the FPGA are summarized in Table 2. The leakage power is comparable to the dynamic power and therefore causes the total energy usage of the system to become much larger. Figure 7 compares the energy dissipation of the seizure detection application across four different processor combinations: Intel Atom, NVIDIA TK1, Artix-7 FPGA and Atom+PENC manycore. For the TK1, using the embedded GPU as an accelerator improved efficiency over using just the ARM A15 processors. Therefore, the combination is only provided for this platform. For all four classifier variants, the manycore accelerator consumes roughly 2-3X less energy than the

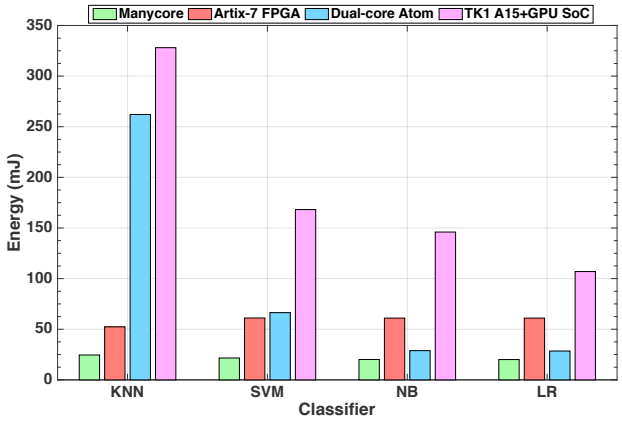

Figure 7: Comparison of different platforms' energy usage to process 1-sec window for the seizure detection case study. All four classifier variants are included. For PENC manycore, the Intel Edison platform power is included to perform data marshalling.

custom FPGA implementation. Besides consuming less energy, the manycore also has the advantage of faster development time compared to FPGA by enabling designs to be implemented in Assembly (and later C) with the ability to rapidly simulate designs on existing hardware. When compared to the TK1 and standalone Atom, the Atom+manycore is able to reduce energy usage by up to 10X. The greatest reduction in energy occurs for the KNN and SVM variants which contain significantly more computation and parallelism as highlighted in the task graphs. For the NB and LR variants, the manycore provides minimal reduction in energy, roughly 1.2X. This is due to the fact that NB and LR have very little computation which is negated by transfer time and long idle times for all processor combinations when not processing. This is corroborated by the fact that the Atom is able to have better energy efficiency than the TK1 with the GPU accelerator for these cases.

The execution time to process one window of EEG data for the seizure detection system is shown in Fig. 8. For all classifiers, the PENC manycore is able to meet the 500 ms deadline along with all other platforms. The manycore is faster than the Intel Atom processor and TK1 ARM+GPU SoC while also dissipating the least amount of energy. The biggest reduction in execution time is seen in KNN-based seizure detection which contains maximum amount of computation and parallelism. For this variant, the manycore is 10X, 15X, and 7X faster than FPGA, standalone Atom, and NVIDIA TK1 implementations, respectively. Only in the NB and LR variants does the FPGA implementation obtain lower execution times, however, the FPGA requires twice the amount of energy.

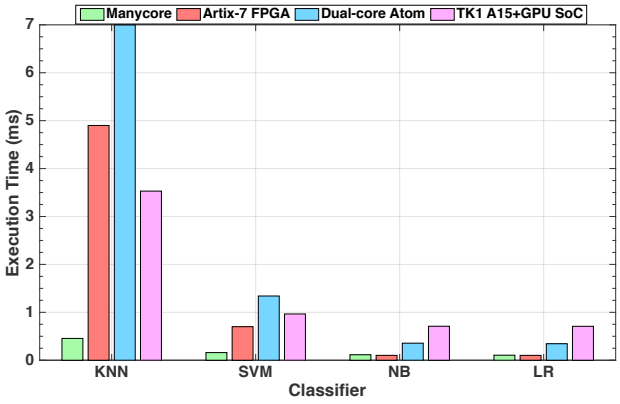

Figure 8: Comparison of different platforms' execution time to process 1-sec window for the seizure detection case study. All four classifier variants are also included. The seizure application assumes 50% overlapping windows which allots 500ms to process each window.

6. CONCLUSION

This paper explores the choice of embedded architectures for energy-efficient processing of biomedical applications. Biomedical applications share strong commonalities requiring sampling from a number of physiological signals and processing that contains various digital signal processing and machine learning kernels. The software as well as hardware implementations of machine learning biomedical kernels are compared. For the choice of software, state-of-the-art commercial off-the-shelf embedded processing platforms such as ARM and Atom are compared with the hardware implementation of these kernels on embedded low-power FPGA. To further push the energy-efficiency, a custom lightweight, symmetric GALS manycore architecture is proposed that enables not only exploiting the task-level and data-level parallelism that exists in biomedical kernels, but also perform dynamic voltage and frequency scaling to dramatically reduce the energy usage. By using the optimal number of cores with DVFS, we were able to reduce energy usage by up to 80% and 14% for DSP and ML tasks, respectively, relative to using the minimal number of cores. When compared to other commercial off-the-shelf platforms for the case study of seizure detection, the PENC manycore is able to reduce energy by up to 10X while having comparable execution time.

7. ACKNOWLEDGEMENT

This material is based upon work supported by the National Science Foundation under Grant No. 00008999 and 00010145.

8. REFERENCES

[1] J. Bisasky, D. Chandler, and T. Mohsenin. A many-core platform implemented for multi-channel seizure detection. In *Circuits and Systems (ISCAS), IEEE International Symposium on*, pages 564–567, May 2012.

[2] J. Bisasky, H. Homayoun, et al. A 64-core platform for biomedical signal processing. In *Quality Electronic Design (ISQED), 2013 14th International Symposium on*, pages 368–372, March 2013.

[3] H. Ghasemzadeh and R. Jafari. Ultra low-power signal processing in wearable monitoring systems: A tiered screening architecture with optimal bit resolution. *ACM Trans. Embed. Comput. Syst.*, 13(1):9:1–9:23, Sept. 2013.

[4] S.-Y. Hsu, Y. Ho, et al. A sub-100uw multi-functional cardiac signal processor for mobile healthcare applications. In *VLSI Circuits (VLSIC), 2012 Symposium on*, pages 156–157, June 2012.

[5] A. Jafari and T. Mohsenin. A low power seizure detection processor based on direct use of compressively-sensed data and employing a deterministic random matrix. In *IEEE Biomedical Circuits and Systems (Biocas) Conference*, Oct 2015.

[6] M. Khavari Tavana, A. Kulkarni, et al. Energy-efficient mapping of biomedical applications on domain-specific accelerator under process variation. In *Proceedings of the 2014 International Symposium on Low Power Electronics and Design*, ISLPED '14, pages 275–278, New York, NY, USA, 2014. ACM.

[7] C. Kim et al. Ulp-srp: Ultra low-power samsung reconfigurable processor for biomedical applications. *ACM Trans. Reconfigurable Tech. Syst.*, 7:22:1–22:15, Sept. 2014.

[8] A. Kulkarni, A. Jafari, et al. Sketching-based high-performance biomedical big data processing accelerator. In *Circuits and Systems (ISCAS), 2016 IEEE International Symposium on*, 2016.

[9] A. Kulkarni, Y. Pino, M. French, and T. Mohsenin. Real-time anomaly detection framework for many-core router through machine learning techniques. In *ACM Journal on Emerging Technologies in Computing (JETC)*, New York, NY, USA, 2016. ACM.

[10] K. H. Lee and N. Verma. A low-power processor with configurable embedded machine-learning accelerators for high-order and adaptive analysis of medical-sensor signals. *IEEE Journal of Solid-State Circuits*, 48:1625–37, July 2013.

[11] P. Otto, M. Malik, N. Akhlaghi, R. Sequeira, H. Homayoun, and S. Sikdar. Power and performance characterization, analysis and tuning for energy-efficient edge detection on atom and arm based platforms. In *Computer Design (ICCD), 2015 33rd IEEE International Conference on*, pages 479–482, Oct 2015.

[12] A. Page, S. Pramod, et al. An ultra low power feature extraction and classification system for wearable seizure detection. In *Engineering in Medicine and Biology Society (EMBC), 2015 37th Annual International Conference of the IEEE*, Sept 2015.

[13] A. Page, C. Sagedy, et al. A flexible multichannel eeg feature extractor and classifier for seizure detection. *Circuits and Systems II: Express Briefs, IEEE Transactions on*, 62(2):109–113, 2015.

[14] A. Page, C. Shea, and T. Mohsenin. Wearable seizure detection using convolutional neural networks with transfer learning. In *Circuits and Systems (ISCAS), 2016 IEEE International Symposium on*, 2016.

[15] A. Venkat and D. M. Tullsen. Harnessing ISA diversity: Design of a heterogeneous-isa chip multiprocessor. In *ACM/IEEE 41st International Symposium on Computer Architecture, ISCA 2014, Minneapolis, MN, USA, June 14-18, 2014*, pages 121–132, 2014.

[16] S. Viseh, M. Ghovanloo, and T. Mohsenin. Towards an ultra low power on-board processor for tongue drive system. *Circuits and Systems II: IEEE Transactions on, accepted*, 62(2):174–178, Feb 2015.

[17] J. Yoo, L. Yan, et al. An 8-channel scalable eeg acquisition soc with patient-specific seizure classification and recording processor. *Solid-State Circuits, IEEE Journal of*, 48(1):214–228, 2013.