A Domain-Specific System-On-Chip Design for Energy Efficient Wearable Edge AI Applications

Yigit Tuncel
tuncel@wisc.edu
UW-Madison
USA

Anish Krishnakumar
anallamurkri@wisc.edu
UW-Madison
USA

Aishwarya L. Chithra
chithra@wisc.edu
UW-Madison
USA

Younghyun Kim
ykim722@wisc.edu
UW-Madison
USA

Umit Ogras
uogras@wisc.edu
UW-Madison
USA

ABSTRACT

Artificial intelligence (AI) based wearable applications collect and process a significant amount of streaming sensor data. Transmitting the raw data to cloud processors wastes scarce energy and threatens user privacy. Wearable edge AI devices should ideally balance two competing requirements: (1) maximizing the energy efficiency using targeted hardware accelerators and (2) providing versatility using general-purpose cores to support arbitrary applications. To this end, we present an open-source domain-specific programmable system-on-chip (SoC) that combines a RISC-V core with a meticulously determined set of accelerators targeting wearable applications. We apply the proposed design method to design an FPGA prototype and six real-life use cases to demonstrate the efficacy of the proposed SoC. Thorough experimental evaluations show that the proposed SoC provides up to 9.1× faster execution and up to 8.9× higher energy efficiency than software implementations in FPGA *while maintaining programmability.*

CCS CONCEPTS

• **Computer systems organization → System on a chip**.

KEYWORDS

Domain-specific system-on-chip, RISC-V, Hardware accelerator, Low-power, Energy efficiency, Wearable computing, Healthcare

ACM Reference Format:
Yigit Tuncel, Anish Krishnakumar, Aishwarya L. Chithra, Younghyun Kim, and Umit Ogras. 2022. A Domain-Specific System-On-Chip Design for Energy Efficient Wearable Edge AI Applications. In *ACM/IEEE International Symposium on Low Power Electronics and Design (ISLPED '22), August 1–3, 2022, Boston, MA, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3531437.3539711

1 INTRODUCTION

Wearable devices have many promising applications ranging from activity recognition to early diagnosis and prognosis of diseases through continuous monitoring of health-related signals [10]. The emergence of new device form factors, such as electronic patches and devices embedded into clothing, further amplifies the potential impact of wearables on everyday life. However, widespread adoption of wearable devices has so far been hindered due to frequent maintenance and charging requirements despite their potential [3].

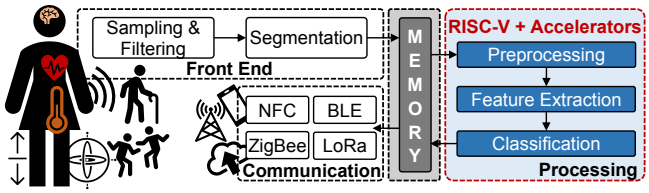

Figure 1: Main steps in a typical wearable edge AI app.

Wearable devices collect data from various sensors, such as accelerometers and biopotentials, to enable sophisticated health monitoring applications, as illustrated in Figure 1. The decadal plan from the Semiconductor Research Corporation highlights that the collective data acquisition rate of sensors will be as high as 10^{20} bits/sec by 2032 [16]. Offloading the sensor data to another device or cloud is prohibitive since wireless data transfer is power-hungry [14, 16]. Therefore, wearable devices must be capable of implementing complex artificial intelligence (AI) and signal processing algorithms at the edge, which requires powerful microprocessors. At the same time, wearable devices also have severely limited battery and energy harvesting capacity due to small form factor requirements. For example, the Fitbit Inspire 2 consumes ~1.5 mW on average, while wearable energy harvesting potential hardly reaches 100 µW, leading to a substantial (>15×) gap between the required and available energy budget [10, 18]. Therefore, there is a strong need for approaches that enable practical wearable devices that satisfy both computational power and energy requirements.

The energy gap can be bridged by two complementary efforts: reducing the energy consumption requirements and increasing the harvested energy. This paper focuses on the former; we propose a domain-specific system-on-chip (SoC) for wearable edge AI-based health monitoring devices that satisfy two competing requirements: programmability and high energy efficiency. General-purpose microprocessors have been favorable for implementing wearable applications since they provide the flexibility to implement a wide range of applications. Indeed, most commercial hardware platforms employ Arm or RISC-V cores as the primary processor [5, 13, 14]. However, the software (SW) programmability comes at the expense of low energy efficiency. Custom hardware (HW) accelerators address this problem and provide 2–3 orders of magnitude higher energy efficiency by achieving faster operation and lower power consumption [1, 4, 9]. However, their benefits diminish if programming the SoC requires significant human effort to support rapidly evolving applications [7, 16]. Hence, there is a strong need for SoCs that can approach the energy efficiency of custom designs while preserving programmability.

ISLPED '22, August 1–3, 2022, Boston, MA, USA

ACM ISBN 978-1-4503-9354-6/22/08…$15.00
https://doi.org/10.1145/3531437.3539711

To our knowledge, this paper presents the first domain-specific SoC that targets wearable applications through a systematic analysis. We identify the most frequently used preprocessing, feature extraction, and classification tasks in wearable health monitoring applications by analyzing 282 recently published articles in this domain. We judiciously selected five tasks among them with the highest execution time and energy consumption. Then, we designed HW accelerators to accelerate them. Thus, the SoC integrates a general-purpose RISC-V core with *five HW accelerators for frequently used compute-intensive tasks*. The RISC-V core enables programmability while the HW accelerators boost the energy efficiency of the most frequently used tasks. The proposed SoC can implement any application and accelerate hundreds of applications that use a subset of the most frequent tasks we identified. As a case study, we implement six representative real-world wearable application use cases to demonstrate the capabilities of the proposed SoC. Extensive evaluations on Xilinx ZCU102 FPGA demonstrate that the proposed SoC boosts the energy efficiency of wearable health monitoring applications by up to 8.9× compared to the reference SW implementations running only on the RISC-V core. Furthermore, our domain-specific SoC design approach preserves the programming flexibility and facilitates adding new HW accelerators and implementing new applications. Our major contributions are:

- Identifying the most commonly used tasks in wearable health monitoring literature and designing HW accelerators for them,
- An energy-efficient domain-specific SoC for health monitoring that integrates our HW accelerators and RISC-V cores,
- Demonstrating the proposed SoC using six representative end-to-end use cases,
- Extensive power, performance, and energy evaluations with a prototype on the Xilinx ZCU102 FPGA.

2 RELATED WORK

State-of-the-art wearable edge AI applications are typically implemented in SW on low-power microcontroller units (MCUs). For example, Samie et. al present an IMU-based human activity recognition (HAR) implementation on a TI CC1350 MCU (housing an Arm Cortex M3 core) [14], with average power and energy consumptions of 9.45 mW and 142.48 µJ per classification. Similarly, the authors in [13] implement a stress detection application that uses electrocardiography (ECG) and galvanic-skin-response data on a custom SoC with a RISC-V core. Another study presents a prototype device that collects and processes electroencephalography (EEG) signals to extract evoked potentials [5]. They use an STM32F412 MCU with an Arm Cortex M4 core that consumes ~18 mW of power. The power consumption of SW implementations motivates researchers to seek more energy-efficient solutions.

ASIC solutions provide much higher energy efficiency than general-purpose cores. For example, authors in [4] present a 30.6 µW dedicated HAR HW accelerator implemented with TSMC 65nm LP library. Similarly, another study demonstrates an ECG peak detector that consumes ~30 µW power in ASIC [9]. However, the increased energy efficiency comes at the expense of flexibility, making custom designs suitable to only a handful of applications.

Domain-specific SoCs support a wide range of applications by combining general-purpose cores with a targeted set of HW accelerators. Using domain-specific accelerators has been recognized as a promising way to boost energy efficiency and performance [1]. For example, recent work focused on accelerating the localization and perception functions in the autonomous driving domain, demonstrating up to 34× higher power efficiency than Intel i7 [8]. Similarly, significant benefits are demonstrated for other domains, including speech recognition and deep learning [19]. However, there is no domain-specific HW platform for resource-constrained wearable edge devices, despite their critical need for energy efficieny.

3 PROPOSED DOMAIN-SPECIFIC SOC DESIGN

To overcome the shortcomings of SW implementations and custom designs, we present a domain-specific SoC for wearable devices. The proposed design addresses the challenges mentioned above by integrating a programmable subsystem for flexibility and extensible HW accelerator cluster for improving energy efficiency and performance, as illustrated in Figure 2. Since different applications in a domain use various tasks for data processing, the proposed design features a programmable subsystem built around a general-purpose RISC-V core. In addition, we integrate an extensible HW accelerator cluster to boost the energy efficiency of frequently used compute-intensive tasks in wearable applications.

3.1 Programmable Subsystem

The programmable subsystem is based on the open-source PULPissimo microcontroller architecture [15]. The PULPissimo platform is used for three reasons. It (1) allows for end-to-end customization thanks to its open-source availability; (2) incorporates low-power components (e.g., RISC-V core and interconnects) and DMA for low-energy data movement; and (3) is silicon-proven. It utilizes a RISC-V processor, an in-order single-issue 4-pipeline stage 32-bit core. The RISC-V core can be enhanced at design time to include floating-point unit (FPU), DSP, and SIMD extensions. The PULP infrastructure also provides a software development kit (SDK) with pre-built APIs and libraries to utilize the on-chip computing resources. The SoC also includes an L2-memory subsystem that allows different cores to share data, as shown in Figure 2. A peripheral AXI bus interconnects the RISC-V core, peripherals, and the control plane of the HW accelerator cluster. The control plane uses the simple advanced peripheral bus (APB) protocol suitable for rare and sporadic transactions. The tightly coupled data memory (TCDM) on-chip network enables high-bandwidth streaming data communication between the RISC-V core, L2 memory, and the HW

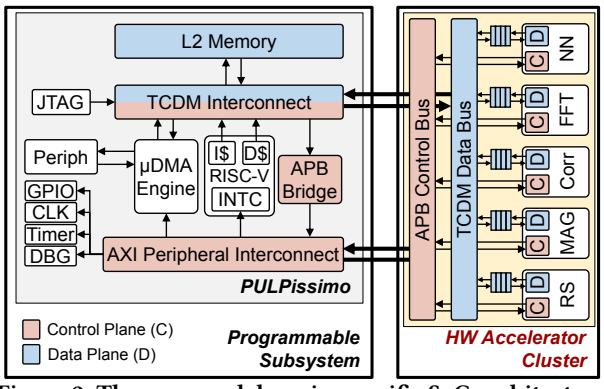

Figure 2: The proposed domain-specific SoC architecture.

accelerator cluster. The TCDM interconnect enables single-cycle transactions between the initiators and responders on-chip. This functionality is supported by providing µDMA units and interrupt controllers to issue handshake signals and move data in a streaming manner over the TCDM interconnect. The RISC-V core does not consume energy for relaying data into the accelerators since the HW accelerators read data directly from L2 memory with the help of the µDMA units. Finally, applications running in the RISC-V core easily utilize our HW accelerators using memory-mapped registers. For instance, a neural network (NN) inference call in the user program issues transactions and initiates the computation in the NN accelerator. Hence, the application developers can conveniently utilize the proposed HW accelerators instead of SW routines.

3.2 Identifying the Tasks to Accelerate

The potential benefits of domain-specific SoCs depend on identifying the tasks to accelerate. This decision is a function of both the prevalence of the task and the energy-performance benefits of accelerating it. We analyze the full text of relevant research papers published between 2015 and 2020 on arXiv. We chose arXiv specifically as it provides open access to the full-text articles. The analysis covers 282 publications found by the following search query:

```
wearable &                                                    (1)
( health || healthcare || medical ) &                         (2)
( device || iot || technology || sensing || edge ||
system || computing || monitoring ) &                         (3)
( signal processing || algorithm || method || technique ||
approach || implementation || application || hardware )       (4)
```

We first build a custom dictionary composed of 164 technical keywords, summarizing the commonly used tasks in wearable applications, such as filtering and statistical features. Then, we find the frequency of these keywords in the selected publications. This analysis summarized in Figure 3 provides the following insights:

(a) Signal statistics (e.g., min, max, mean, variance) are the most common features;
(b) NN, support-vector-machines (SVM), and decision trees (DT) are commonly used for classification;
(c) Windowing (e.g., Hamming window), standardization (e.g., mean removal and normalization), filtering, segmentation, and resampling are commonly used for signal-preprocessing;
(d) Temporal (e.g., peak count), correlation (e.g., covariance), spectral analysis (e.g., fast Fourier transform (FFT)), and vector magnitude computation are frequently used for feature generation.

Using an HW accelerator for a given task is profitable only if the execution time and energy consumption savings surpass the offload overhead. Therefore, we implemented these most frequently used tasks on the proposed SoC using only the RISC-V core to profile their execution time and power consumption. As a result of this analysis (detailed in Section 5), we designed HW accelerators for NN, correlator, resampling, magnitude computation, and used an open-source FFT core [11], as discussed next.

3.3 Design of the Proposed HW Accelerators

NN accelerator: Our design implements the inference of multilayer perceptron-based neural networks. It has two input channels (one for the weights and another for input features) and a single output channel that emits the output probabilities. It incorporates a parameterized *neuron* submodule and multiply-accumulate engine. Hence, it can be configured to implement NNs with an arbitrary number of layers and neurons. For example, a 3-layer network with 4-8-8 neurons in its layers takes 163 cycles to execute and is pipelined to provide a throughput of 1 output every 127 cycles.

Correlation (Corr) accelerator: The correlation engine calculates the cross-correlation between any two equal-length vectors. For example, we use it to calculate the autocorrelation of a signal with time-lagged versions of itself. For efficient vector multiplication, we leverage the Multiply-Accumulate engine released by the developers of the PULPissimo system [15] in the correlation accelerator. It can output one sample per cycle when fully pipelined.

A ReSampling (RS) accelerator: Resampling changes the sampling rate of a discrete signal either by dropping samples or by interpolating new samples. In wearable edge AI applications, reducing the number of samples is helpful to conserve memory and decrease the processing effort. Thus, we implemented a programmable, broadly usable accelerator that can be configured to downsample from an arbitrary N number of samples to M samples. It has one input channel and one output channel. The operating principle of the RS accelerator is as follows: It retains the first and last samples in an array. Then, if N/M is an integer number, it retains every N/M^{th} sample and discards the remaining samples. If N/M is not an integer number, it retains samples such that the average interval between the samples is N/M. Thus, it outputs one sample every N/M cycles on average and has a latency of $N + M + 2$ clock cycles.

A MAGnitude (MAG) accelerator: The MAG accelerator calculates the magnitudes of a vector of 2–3 tuples. The mode of operation is programmable, such that developers can use it for different purposes. For example, it can be used in the 3-tuple mode to compute the body-acceleration b_{acc} from accelerometer samples coming from a three-axis IMU (a_x, a_y, a_z), where $b_{acc} = \sqrt{a_x^2 + a_y^2 + a_z^2}$. Similarly, it can be used in the 2-tuple mode to compute the magnitudes of a vector of complex numbers, such as the output of an FFT operation. It can output one sample per cycle when fully pipelined and has a 90 clock cycle end-to-end latency.

A Fast-Fourier-Transform (FFT) accelerator: We generate an optimized 64-point FFT core with a 233 cycle latency using the Spiral IP generator [11] to accelerate the FFT computation. It has two input channels for two complex samples (16 bits each for real and imaginary), and similarly two output channels. The FFT accelerator supports a maximum throughput of 2 complex words per cycle.

3.4 Accelerator Integration to the SoC

The RISC-V core controls the operation of HW accelerators through the APB bus. The control interface of each HW accelerator exposes memory-mapped registers in the L2 memory to the APB bus. These registers are responsible for the control operations, such as triggering the operation of an accelerator, and other runtime configuration parameters, such as the input size. We extend the PULP-SDK for each accelerator and provide APIs specific to each accelerator. At runtime, the APIs in the SDK transfer the configuration parameters from the SW code to the specific HW accelerators. For example, we configure the RS accelerator to downsample the accelerometer and stretch sensor 64 and 32 data points, respectively. Providing such

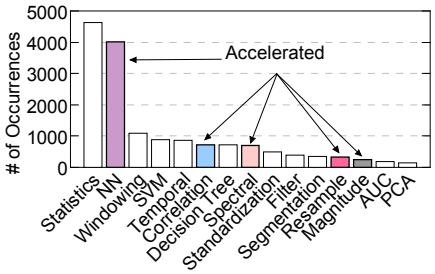

Figure 3: Frequency of keywords in 282 papers from relevant literature.

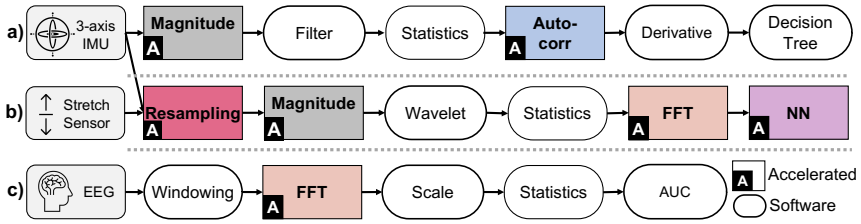

Figure 4: Applications used for the evaluation of the proposed SoC – a) Step counter, b) HAR, c) EEG analytics.

handles to the developers allows for the easy runtime programmability of the SoC. Offloading the computation from SW to an HW accelerator comprises the following steps:

(1) transfer the input data to a chosen buffer space in L2 memory,
(2) program the accelerator's parameters and address space,
(3) initiate accelerator computation,
(4) wait for accelerator completion, and
(5) read the output data from L2 memory buffer space.

Table 1 presents the speedup of tasks in HW accelerators compared to computation in the RISC-V core. We emphasize that the presented speedups include all the additional programming and data transfer overheads, and hence are realistic speedups that users will observe when the respective tasks are accelerated in HW.

Table 1: Sample exec. times of accelerated tasks (at 10 MHz).

	NN (4-8-8)	Corr (80×80)	RS (205→64)	MAG (64 3-tuples)	FFT (64-point)
SW–RISC-V (ms)	10.80	15.50	5.10	1.80	29.82
Accelerated (ms)	0.63	2.11	0.31	0.06	1.53
Speedup (×)	17.25	7.36	16.71	29.51	19.55

4 DRIVER APPLICATION USE CASES

We implemented three popular applications to demonstrate the proposed SoC: (1) Step counter, (2) HAR, and (3) EEG analytics. They are representative examples since 11 out of 13 tasks in these applications are among the most commonly used kernels listed in Figure 3. Furthermore, we analyze the concurrent execution of these three applications as additional real-life use cases.

Step counter: This application is implemented in most wearable devices due to the strong correlation between daily walking activity and cardiovascular health [6]. A recent study claims that step counter-based interventions that aim to increase daily walking activity lead to significantly fewer new cardiovascular events in the long term [6]. The authors further state that step counter is an essential tool to set realistic daily goals for each individual. In our study, we use an auto-correlation-based step counting algorithm presented in [12], which is outlined in Figure 4a). We adapt an open-sourced implementation of the algorithm[1].

Human Activity Recognition: HAR is a critical enabler for a broad range of healthcare and fitness applications [2]. It can help Parkinson's disease patients by enabling auditory cues when symptoms like tremor or freezing of gait occur. Due to its importance, there are numerous HAR implementations and datasets in the literature. We use the w-HAR dataset that contains data collected from an inertial measurement unit at the ankle and a stretch sensor on the knee for 22 users and seven activities [2]. The authors provide the algorithm summarized in Figure 4b).

EEG analytics: This application computes the power spectral density of input EEG signals and divides them into distinct frequency bands, such as the delta (0.5–4 Hz), theta (4–8 Hz), alpha (8–12 Hz), beta (12–30 Hz), and gamma (30–100 Hz) bands. Sleep and attention studies usually calculate the average power in these bands relative to each other. Increased activity in the delta band is associated with deep sleep, whereas an attenuated delta band activity and increased high-frequency activity is related to wakefulness [17]. In this work, we use Welch's method to calculate the power spectral density (PSD) estimate of the EEG recording and calculate the area under the PSD for the frequency band of interest. Figure 4c) shows the steps of the algorithm. We use the publicly available EEG data provided by the Center for Human Sleep Science at UC Berkeley [2].

5 EXPERIMENTAL EVALUATION

This section evaluates an FPGA implementation of the proposed SoC in terms of execution time and energy efficiency benefits. To this end, we implement the driver applications on the proposed SoC with and without HW acceleration. Before moving to costly tapeout, an FPGA prototype helps us validate the functionality and obtain estimates of execution time gains. As an extra step, we also measure the power consumption on the FPGA and analyze the energy savings enabled by the proposed design. Although the absolute numbers are not representative of a taped-out design, the obtained energy savings compared to the SW implementation on the FPGA gives a conservative estimate of the actual benefits. We expect significantly higher energy efficiency in the taped-out design since the FPGA hardware building blocks (lookup tables and routing matrices) consume higher power than taped-out implementations (logic gates and wires). In the following, we first introduce the experimental setup and procedure. Then, we measure the execution times and power consumption of the applications. Finally, we present the measurement results and our analysis.

5.1 Experimental Setup and Procedure

Setup: The proposed SoC is implemented and evaluated on the Xilinx ZCU102 FPGA board at four different frequencies: 10 MHz, 16 MHz, 25 MHz, and 40 MHz. The entire design is mapped to the programmable logic (PL) on the ZCU102. Since the platform does not allow configuring the clock dividers at runtime, we synthesize

[1]Neraj Bobra. https://github.com/nerajbobra/embedded_pedometer

[2]Raphael Vallat. https://raphaelvallat.com/images/tutorials/bandpower/data.txt

the design using Vivado v2019.2 for each frequency and verify that FPGA utilization is similar in all synthesis results. We emphasize that *all four synthesized designs include all five accelerators*, irrespective of whether they are utilized or not, in order to make fair comparisons within each application and between applications.

Application use cases: The three applications presented in Section 4 are used for evaluation. In addition to running them individually, we also study three additional concurrent real-world use cases by combining (1) Step counter and HAR, (2) Step counter and EEG analytics, and (3) all three applications. We collect measurements for the pure SW implementations and the corresponding accelerated versions to evaluate all use cases at four frequencies.

Procedure: We collect (1) execution time and (2) power consumption measurements from the FPGA implementation: (1) Execution time is measured using timer functions provided in the PULP-SDK that has 30µs resolution. These measurements are used to demonstrate the speedup enabled by the HW accelerators integrated into the proposed SoC. (2) For power consumption measurements, we measure the load current on the primary rails that power the programmable logic on the ZCU102 board using a National Instruments DAQ at 250 kS/s. The primary rails are the internal supply voltage (V_{ccint}) and the BRAM voltage (V_{ccbram}) rails.

Each application use case runs 1000 times to allow for averaging of the power measurements. After averaging 1000 measurements, we apply a moving average filter with a window of 50 samples to reduce the high-frequency noise. A 50-sample window is chosen to have a negligible time-shift in order not to distort the signal. The obtained waveform allows us to clearly observe the power consumption, as depicted in Figure 5. Then, we use the power measurements to find the energy consumption of the SoC.

5.2 Execution Time Analysis

This section analyzes the individual applications in terms of their execution time for SW and accelerated implementations.

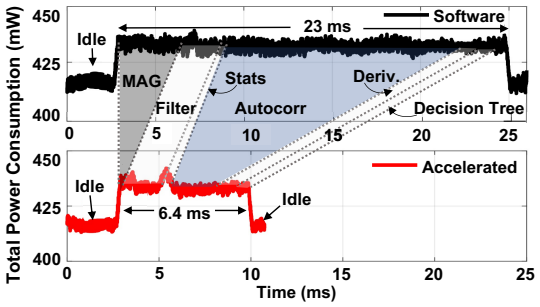

Figure 5: Power consumption of the SW and accelerated implementations of the step counter application at 10 MHz.

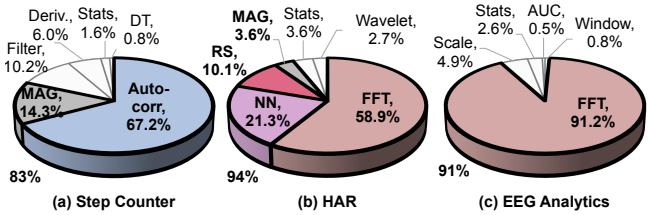

Figure 6: Breakdown of execution times of the 3 applications.

Profiling SW implementations: Figure 6 presents the execution time breakdown of the applications when implemented in SW on the RISC-V core. The execution time analysis shows that:
(a) the magnitude and auto-correlation computations constitute 83% of the total execution time of the step counter application;
(b) resampling, magnitude, FFT computation, and neural network inference constitute 94% of the HAR application execution time;
(c) FFT computation constitutes 91% of the EEG analytics application execution time.
These results justify our choice of accelerated tasks in the SoC.

Execution time: Figure 7-Top illustrates the execution time of the SW and accelerated implementations of the three applications. Utilizing the HW accelerators yields 3.6× speedup in execution time for the step counter application at the same frequency. Specifically, at 10 MHz, the SW implementation takes 23.0 ms to run, whereas the accelerated implementation takes 6.4 ms. Similarly, the accelerated HAR and EEG analytics applications are executed up to 9.1× and 8.8× faster than their SW counterparts, respectively. In addition, execution times scale down linearly with increasing frequency. Furthermore, the execution time speedups remain consistent across different frequencies.

5.3 Power and Energy Analysis

This section presents the power and energy consumption measurements for the SW and accelerated implementations of the six application use cases.

Power measurements: We collect current measurements from the FPGA as outlined in Section 5.1 to analyze the power and energy consumption of the design. The power consumption of the V_{ccint} and V_{ccbram} power rails are proportional to their current consumption since both have a fixed voltage level of 0.85 V.

The core is set to idle before and after program execution while taking the measurements. Figure 5 illustrates the power consumption measurement for the step counter application at 10 MHz. The lower power regions in this figure correspond to the idle state, and the higher power regions correspond to the active state. According to the measurements, the utilization of HW accelerators does not cause a significant change in active state power consumption. In addition, we note that the power consumption measured from the V_{ccbram} rail is a constant 75 mW for all cases. For the illustrated case (at 10 MHz), the average idle and active power consumptions are 416 mW and 440 mW, respectively, as shown in Table 2. Similarly, for HAR and EEG analytics applications, the idle and active power consumptions are {417, 441} mW and {419, 442} mW, respectively, which suggest that power consumption *on the FPGA* is not a strong function of the executed application. According to the power measurements at different frequencies in Table 2, the average idle and active state power consumptions increase with frequency.

Energy consumption and savings: Energy consumption is computed by the area under the total power consumption curve when a

Table 2: Average power consumption of the 3 applications.

Power in mW	Step counter		HAR		EEG Analytics	
	P_{idle}	P_{act}	P_{idle}	P_{act}	P_{idle}	P_{act}
10 MHz	416	440	417	441	419	442
16 MHz	438	475	438	473	438	473
25 MHz	465	510	460	510	460	510
40 MHz	509	577	503	579	503	579

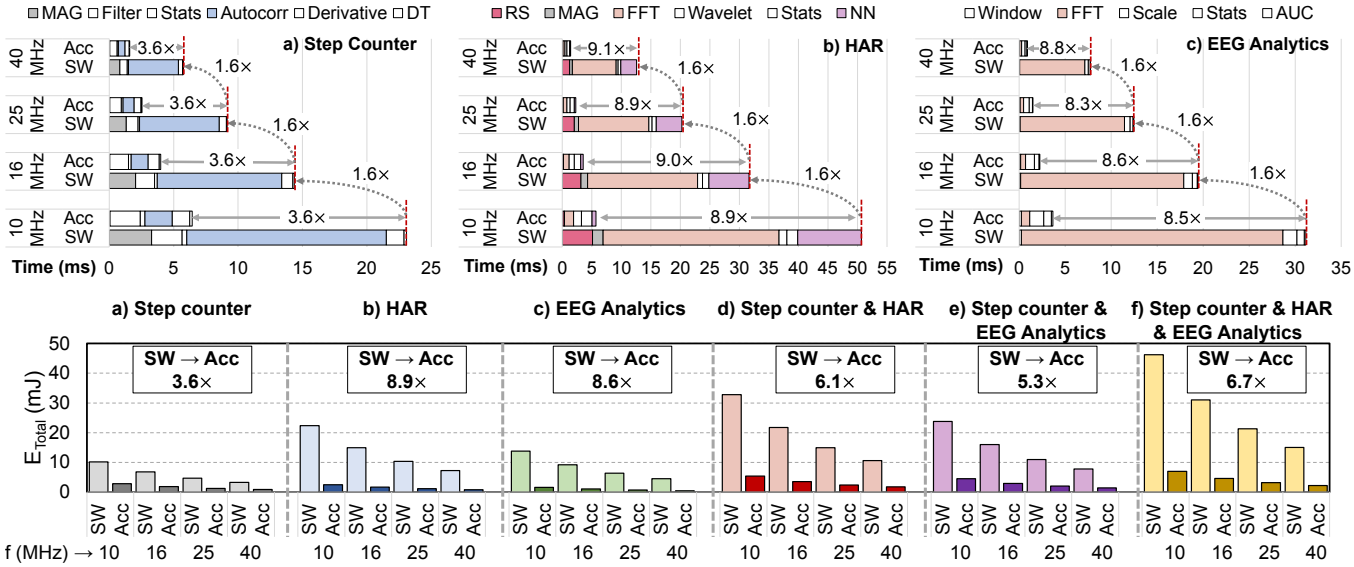

Figure 7: Top) Execution times of the three applications. Bottom) Energy consumption of the six application use cases.

program is executed (i.e., active region in Figure 5). The energy consumption scales with a similar factor as the execution time since the SW and accelerated implementations have similar power consumption, as summarized in Figure 7-Bottom. According to our results, the proposed SoC achieves a 3.6× reduction in energy consumption for the step counter application at all frequencies. For example, at 10 MHz, the energy consumption for the step counter application reduces from 10.14 mJ to 2.83 mJ. Similarly, 8.9× and 8.6× gains are achieved for the HAR and EEG analytics applications, respectively. Moreover, the proposed SoC achieves between 5.3× to 6.7× reduction in energy consumption for the concurrent application use cases. The absolute power values collected from the FPGA are not indicative of a chip design implemented with a low-power process technology. However, power and energy consumption analysis on the FPGA is helpful to obtain early conservative estimates and a lower bound of the gains that the proposed SoC can achieve.

6 CONCLUSION AND FUTURE WORK

This paper presented a domain-specific SoC design that integrates a general-purpose RISC-V core with five HW accelerators, which are selected by analyzing the frequently used tasks in the domain of wearable health monitoring applications. Our evaluations on an FPGA prototype with six real-world application use cases show that the proposed domain-specific SoC enables up to 9.1× faster execution time and up to 8.9× higher energy efficiency than SW implementations. As part of our future work, we aim to improve the energy-efficiency of the proposed SoC by: i) studying the optimal number of instances for each type of HW accelerator, and ii) further optimizing the HW accelerators. Finally, we also plan to implement and fabricate the design in a recent process technology and extend our evaluations beyond the FPGA prototype.

ACKNOWLEDGMENTS

This work was supported by the NSF CNS-2114499, EECS-2132904, CNS-1845469, CNS-2112562 and DARPA D14AP00068 awards.

REFERENCES

[1] Samet E Arda et al. 2020. DS3: A system-level domain-specific system-on-chip simulation framework. *IEEE Trans. Comput.* 69, 8 (2020), 1248–1262.
[2] Ganapati Bhat et al. 2020. w-HAR: An Activity Recognition Dataset and Framework using Low-power Wearable Devices. *Sensors* 20, 18 (2020), 5356.
[3] Ganapati Bhat, Ranadeep Deb, and Umit Y Ogras. 2019. OpenHealth: Open-source Platform for Wearable Health Monitoring. *IEEE Design & Test* 36, 5 (2019), 27–34.
[4] Ganapati Bhat, Yigit Tuncel, Sizhe An, Hyung Gyu Lee, and Umit Y Ogras. 2019. An Ultra-low Energy Human Activity Recognition Accelerator for Wearable Health Applications. *ACM TECS* 18, 5s (2019), 1–22.
[5] Marco Guermandi et al. 2018. A Wearable Device for Minimally-invasive Behind-the-ear EEG and Evoked Potentials. In *IEEE BioCAS*. 1–4.
[6] Tess Harris et al. 2019. Effect of Pedometer-based Walking Interventions on Long-term Health Outcomes: Prospective 4-year Follow-up of Two Randomised Controlled Trials using Routine Primary Care Data. *PLoS medicine* 16, 6 (2019).
[7] John L Hennessy and David A Patterson. 2019. A New Golden Age for Computer Architecture. *Commun. of the ACM* 62, 2 (2019), 48–60.
[8] Shaoshan Liu et al. 2019. Edge Computing for Autonomous Driving: Opportunities and Challenges. *Proc. IEEE* 107, 8 (2019), 1697–1716.
[9] Xin Liu et al. 2010. A Miniature On-Chip Multi-Functional ECG Signal Processor with 30 μW Ultra-low Power Consumption. In *Annual International Conf. of the IEEE Engineering in Medicine and Biology*. 2577–2580.
[10] Michele Magno and David Boyle. 2017. Wearable Energy Harvesting: From Body to Battery. In *12th IEEE DTIS*. 1–6.
[11] Peter Milder et al. 2012. Computer Generation of Hardware for Linear Digital Signal Processing Transforms. *ACM TODAES* 17, 2 (2012), 1–33.
[12] Dan Morris et al. 2014. RecoFit: Using a Wearable Sensor to Find, Recognize, and Count Repetitive Exercises. In *ACM CHI*. 3225–3234.
[13] Antonio Pullini et al. 2019. Mr. Wolf: An Energy-Precision Scalable Parallel Ultra Low Power SoC for IoT Edge Processing. *IEEE JSSC* 54, 7 (2019), 1970–1981.
[14] Farzad Samie, Lars Bauer, and Jörg Henkel. 2020. Hierarchical Classification for Constrained IoT Devices: A Case Study on Human Activity Recognition. *IEEE Internet of Things Journal* 7, 9 (2020), 8287–8295.
[15] Pasquale Davide Schiavone et al. 2018. Quentin: An Ultra-low-power PULPissimo SoC in 22nm FDX. In *IEEE S3S*. 1–3.
[16] Semiconductor Research Corporation (SRC). 2022. Decadal Plan for Semiconductors - Full Report. https://www.src.org/about/decadal-plan/decadal-plan-full-report.pdf, accessed 1 Mar. 2022.
[17] Kristina Šušmáková. 2004. Human Sleep and Sleep EEG. *Measurement Science Review* 4, 2 (2004), 59–74.
[18] Yigit Tuncel, Toygun Basaklar, and Umit Ogras. 2021. How much energy can we harvest daily for wearable applications?. In *2021 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*. IEEE, 1–6.
[19] Sam Xi et al. 2020. Smaug: End-to-end Full-stack Simulation Infrastructure for Deep Learning Workloads. *ACM TACO* 17, 4 (2020), 1–26.