An Ultra-Low Energy Human Activity Recognition Accelerator for Wearable Health Applications

GANAPATI BHAT, Arizona State University
YIGIT TUNCEL, Arizona State University
SIZHE AN, Arizona State University
HYUNG GYU LEE, Daegu University
UMIT Y. OGRAS, Arizona State University

Human activity recognition (HAR) has recently received significant attention due to its wide range of applications in health and activity monitoring. The nature of these applications requires mobile or wearable devices with limited battery capacity. User surveys show that charging requirement is one of the leading reasons for abandoning these devices. Hence, practical solutions must offer ultra-low power capabilities that enable operation on harvested energy. To address this need, we present *the first fully integrated custom hardware accelerator* (HAR engine) that consumes 22.4 µJ per operation using a commercial 65 nm technology. We present a complete solution that integrates *all steps of HAR*, i.e., reading the raw sensor data, generating features, and activity classification using a deep neural network (DNN). It achieves 95% accuracy in recognizing 8 common human activities while providing three orders of magnitude higher energy efficiency compared to existing solutions.

CCS Concepts: • **Computer systems organization → Embedded hardware**; • **Hardware → Application specific processors**; **Sensor applications and deployments**; **Chip-level power issues**;

Additional Key Words and Phrases: Human activity recognition, wearable electronics, flexible hybrid electronics, hardware accelerator, low-power design, health monitoring.

ACM Reference format:
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 Trans. Embedd. Comput. Syst.* 9, 4, Article 1 (October 2019), 22 pages.
DOI: 0000001.0000001

1 INTRODUCTION

Human activity recognition enables a broad range of health and activity monitoring applications. For example, it can help users suffering from movement disorders and obesity accurately track a

This article was presented in the International Conference on Compilers, Architecture, and Synthesis for Embedded Systems (CASES) 2019 and appears as part of the ESWEEK-TECS special issue.
This work was supported in part by NSF CAREER award CNS-1651624, DARPA Young Faculty Award (YFA) Grant D14AP00068, and National Research Foundation of Korea Grant NRF-2017R1D1A1B03032382.
Author's addresses: G. Bhat, Y. Tuncel, S. An, and U. Y. Ogras, School of Electrical, Computer and Energy Engineering, Arizona State University, Tempe, AZ, 85287; emails: {gmbhat, ytuncel, sizhean, umit}@asu.edu; H.G. Lee, School of Computer and Communication Engineering, Daegu University, South Korea 38453; email: hglee@daegu.ac.kr.

multitude of daily activities, not just simple step counts. This information is valuable to health professionals because it is much more reliable than self-reporting [7, 36]. Furthermore, accurate activity recognition also provides context for more detailed analysis, such as stride length measurement for gait analysis and tremor monitoring for Parkinson's Disease patients [20, 31].

The potential of HAR has led to both academic work [41] and commercial devices [17]. The nature of target applications requires the user to either carry or wear the device that performs HAR. Thus, a significant number of studies during the last decade used smartphones [41], while wearable solutions based on inertial measurement units (IMU) gained momentum with the advances in wearable electronics [26]. A recent user survey concludes that 27% of users give up using the device since charging is inconvenient, especially for those coping with a health problem [13]. Similarly, a significant number of users do not want to transmit their personal data to a different device due to privacy concerns [36]. Thus, practical solutions must perform HAR locally under an ultra-low energy budget that practically eliminates charging requirements.

Existing solutions on smartphones are inconvenient because the users need to carry a device. Furthermore, smartphones consume in the order of few Watts [32] and fail to provide real-time guarantees since activity monitoring is not their primary goal. Low-power wearable devices can address these problems and bring the power consumption down to 10–30 mW [6]. However, this power consumption range is still significantly larger than the capacity of ambient energy harvesting sources, such as photovoltaic cells (indoor: 0.1 mW/cm^2) [37, 44] and human motion (0.73 mW/cm^3) [15]. Moreover, this power envelope still leads to merely 40 hours of operation using a wearable form-factor 1 g battery with 130 mAh capacity [14].

This paper presents the first fully integrated ultra-low power hardware accelerator that provides an end-to-end solution, from reading the sensors all the way to classifying the activity. The proposed HAR engine first reads in the raw sensor readings through a parameterized interface that can adapt to different sensor inputs. In this work, we employ a 3-axis accelerometer sensor and a capacitive stretch sensor sampled at 250 Hz and 25 Hz, respectively. Then, it preprocesses the raw sensor data using a moving average filter. After this step, it generates the features used for classification. Our flexible feature generation blocks allow generation of both simple statistical features and complex features, such as 64-point fast Fourier transform (FFT) and discrete wavelet transform (DWT) coefficients. Finally, the features are used by a deep neural network (DNN) to classify the following locomotion activities: {*jump, lie down, sit, stand, stairs down, stairs up, walk*}. To assist people with movement disorders, we focus on identifying the locomotion activity in which they are engaged.

The novel contributions of this work are threefold. *First*, we present a baseline HAR engine with a single-level classifier to quantify the impact of custom design over programmable solutions. Post-layout evaluations using a 65 nm TSMC technology show that our baseline design has 1.353 mm^2 area. It achieves 95% accuracy while consuming 51 µW active power and 14 µW idle power. This leads to 22.4 µJ per activity, which is about two orders of magnitude lower than the best embedded solution reported in the literature [5, 6]. Detailed power consumption breakdown of our baseline design also reveals that the FFT feature generation and 3-layer DNN are the dominant components. Our *second* contribution is a novel activity-aware low-power HAR engine that uses the insights provided by our baseline design and the nature of human activities. This design classifies the activities first as *static* and *dynamic* using a simple support vector machine classifier and simple statistical features. If the output of the first level is *static*, then the actual activity is found using the same features with a decision tree. This design eliminates both complex features and the DNN classifier to reduce the dynamic power consumption to 5.56 µW. If the output of the first-level classifier is *dynamic*, then we use a smaller single hidden layer DNN to identify the activity. Our novel activity-aware HAR engine design achieves higher (97%) accuracy with a negligible penalty in the area. More importantly, the total power consumption drops from 51 μW to 20 μW and 45 μW for static and dynamic activities, respectively. *Finally*, the proposed designs are evaluated extensively with both in-house data from 22 user studies and a publicly available dataset [40].

The major contributions of this paper are as follows:

- The first fully integrated custom HAR engine that integrates all steps from reading raw sensor data to activity classification,
- A novel activity-aware HAR engine that consumes the lowest energy (22.4 μJ per activity) reported in the literature,
- An extensive experimental study with HAR data from 22 users and post-layout evaluations using 65 nm TSMC technology.

The rest of this paper is organized as follows: Section 2 reviews the related work and highlights our unique contributions. Section 3 gives an overview of the proposed baseline HAR engine, while Section 4 presents our activity-aware 2-level HAR engine. Section 5 describes the power consumption optimization techniques used in the paper. Finally, Section 6 presents the experimental evaluation, and Section 7 summarizes our major contributions.

2 RELATED WORK

Human activity recognition has received increased attention due to its health monitoring and fitness tracking applications [4, 9, 33]. HAR enables these applications by continuously monitoring sensor data and inferring user activity. Most of the existing HAR approaches employ fixed activity windows and statistical features, such as mean, median, minimum, and maximum [2, 25, 38]. Several recent studies use FFT and DWT, similar to our work [11]. Existing studies also use mainly inertial measurement units, which include accelerometer, gyroscope, and magnetometer sensors. They exploit a variety of classifiers, including k-NN, support vector machines, decision trees, and random forest, which are trained offline. A comprehensive survey of these techniques can be found in [26]. Recently, deep convolutional neural networks (CNN) and recurrent neural networks (RNN) have been employed for human activity recognition [19, 35]. While CNN and RNN provide higher accuracy, they impose significant memory and computational overheads.

A significant number of prior HAR techniques are implemented on smartphones [16, 22, 39]. However, smartphones are not suitable for continuous activity monitoring because they are not designed for real-time applications and their power consumption is in the order of Watts. Therefore, recent approaches implement HAR on small form-factor wearable IoT devices to reduce the power consumption [3, 26]. These devices need to operate under tight energy budget and power constraints due to their small size. However, the lowest reported power consumption for human activity recognition is still in the order of 10 mW [5, 6]. Therefore, there is a strong need for hardware accelerators that can significantly lower the power consumption.

Several recent studies have proposed special purpose hardware for human activity recognition [23, 29, 45]. Klinefelter et al. present a system on chip (SoC) that integrates an analog front-end (AFE), a microcontroller, energy harvesting capability, and a wireless modem for biomedical applications [23]. Similarly, the work in [45] proposes a signal-acquisition SoC for personal health applications. The main focus of these studies is to develop an AFE for acquiring sensor data for health applications. The AFE is integrated to an Arm Cortex-M0 processor and multiply-accumulate units to provide digital processing capabilities. While this approach provides a low power consumption for the AFE, it has to use the Arm core for operations such as preprocessing and feature generation, leading to a higher power consumption. The work in [29] develops a hardware accelerator for monitoring

Table 1. Comparison of the proposed HAR engine to relevant custom designs reported in the literature

Ref.	[46]	[30]	[45]	[28]	[10]	**Proposed**
Target App.	Vital signal monitoring	Vital signal monitoring	Signal acquisition	Signal acquisition	Sensor AFE for physical act.	HAR
Technology	130 nm	130 nm	180 nm	180 nm	500 nm	65 nm
Frequency	32 kHz	1-20 MHz	1 MHz	Up to 2 kHz	120 Hz	100 kHz
Voltage	1.0 V	0.9 V	1.2 V	1.1 V	2.7 V-3.3 V	1.0 V
Power	530 μW	93–322 μW	120 μW	88.6 μW	108–132 μW	45–51 μW
Area	16 mm^2	6.25 mm^2	49 mm^2	5.45 mm^2	196 mm^2	1.35 mm^2

the change in activity of the user. Specifically, the authors implement a dynamic time warping-based decision-making module to detect movements of interest. Whenever the module detects a movement of interest, a sophisticated processing unit, such as a microcontroller, is activated to extract more information about the movement. While this design can reduce the idle power consumption, it still requires a higher-power microcontroller to classify activities. In contrast, our HAR engine provides a *fully integrated low-power solution* for all aspects of HAR from data preprocessing to classification.

Custom-designed solutions attracted significant attention in health and activity monitoring applications due to their power consumption advantages. For example, Wong et al. [46] present an SoC for vital sign monitoring implemented in the 130 nm technology. The proposed SoC integrates a full-custom hardware MAC, digital microprocessor core and I/O peripherals, analog to digital converter, wireless transceiver and custom sensor interfaces. Its power consumption is 530 μW at 1 V supply voltage, as shown in Table 1. A more recent vital signal monitoring SoC, also implemented in 130 nm technology, achieves 93–322 μW power consumption at 0.9 V supply voltage [30]. Similarly, a signal-acquisition SoC for personal health applications is proposed in [45]. This design achieves 120 μW power consumption at 1.2 V supply voltage. A more specialized ASIC design for ECG signal acquisition achieves 88.6 μW power consumption by operating at lower than 2 kHz frequency at 1.1 V supply voltage [28]. Finally, a generic sensor front-end architecture for physical activity monitoring systems is presented in [10]. This design provides a flexible way to build a complete sensor interface chip which consumes 120 μW in ON-state.

In this work, we propose an ultra-low energy end-to-end *hardware accelerator* for HAR. We use a commercial AFE to read sensor data and feed it to the accelerator. We demonstrate the proposed HAR engine on the 65 nm LP commercial technology using activity data from 22 users. To the best of our knowledge, this is the first hardware implementation that accelerates all steps of digital processing. It leads to three orders of magnitude higher energy-efficiency compared to existing *software implementations* on low-power microcontrollers [6]. Furthermore, it has competitive power consumption and area compared to relevant ASIC implementations reported in the literature as summarized in Table 1.

3 THE PROPOSED BASELINE HAR ENGINE

3.1 Input Data

The input to the proposed HAR engine is the raw sensor data. The choice of sensors affects both the classification accuracy and power consumption. In this work, we employ a 3-axis accelerometer, one of the most commonly used sensors, and a stretch sensor. In our user studies, we also collected 3-axis gyroscope data. However, experiments show that adding them does not increase the accuracy significantly, even though it incurs a 1-10 mW power consumption overhead.

3-Axis Accelerometer: The proposed HAR accelerator receives the streams from a 3-axis accelerometer. The sampling rate of this device is set to 250 samples/s and each output sample to the accelerator consists of three 16-bit words for x, y, z axes of the accelerometer, respectively.

Stretch Sensor: The textile-based stretch sensor [34] measures the degree of bending at joints of our body. In our design, the stretch sensor is attached to one knee of the user. The output of the stretch sensor is a 16-bit capacitive value, which is normalized to have a range similar to the accelerometer. We use 25 Hz sampling rate and stream the data from the sensor to the proposed HAR engine.

3.2 Preprocessing the Raw Sensor Data

Raw sensor data is commonly preprocessed to filter out noise and prepare for feature generation. The proposed design employs a moving average filter with a width of 8 samples to smooth the input data. All four streams, i.e., 3-axis accelerometer and stretch data, flow through the filter. Since the body acceleration provides useful information about the user movement, we also compute it using the filtered outputs as follows:

$$b_{acc} = \sqrt{a_x^2 + a_y^2 + a_z^2} \tag{1}$$

The final preprocessing step is segmenting the data into activity windows. This is necessary because more than one-second period of data is required to identify the underlying activity. Each new sample is first stored in a FIFO buffer to segment the data efficiently in real-time. Meanwhile, the segmentation block computes the five-point derivative of the stretch data to identify trends in the activity, such as flat, increasing and decreasing regions. It marks the boundary of a window when a new trend is detected or the maximum window duration (3 seconds in our design) is reached, as shown in Figure 1.

The segmentation block also plays a key role in minimizing the energy consumption of the proposed HAR engine, which is one of our major goals. We note that the feature generation and activity classifier can stay mostly in a low-power mode, i.e., clock or power gated, until the data of a whole window is populated. We utilize the output of the segmentation block to enable the feature generation and activity classifier blocks. After completing the classification of the current activity in those blocks, they immediately go back to the low-power mode until the enable signal from the segmentation block is detected, as described in Section 5.

3.3 Feature Generation

Downsampling and Smoothing: Once the segmentation block marks the completion of an activity window, the feature generation block in Figure 1 starts reading the data from the FIFO.

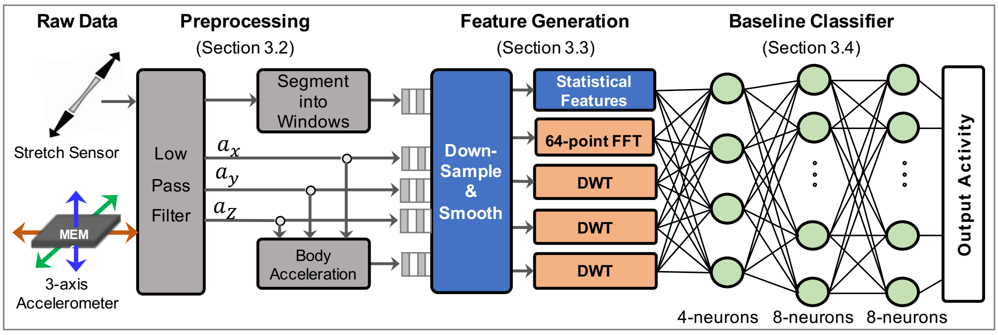

Fig. 1. Architecture of the baseline HAR Engine

The DNN classifier requires a fixed-length of features at its inputs, while the duration of activity window in the segmentation block is variable. For example, the duration of dynamic activities, such as walk and run, is smaller than those of static activities. Moreover, the activity duration may show large variations across different dynamic activities and across users, even for the same activity.

The feature generation step starts with a block to produce a fixed-length feature data set regardless of the activity duration. For this purpose, the segmentation block generates an 8-bit output that specifies the length of the activity window. This length is then used to determine the downsampling rate required to reduce the input data size to the feature data size. In this work, the accelerometer is downsampled to 64-sample windows and stretch data is downsampled to 32-sample windows since larger values do not increase the accuracy. At the same time, our parameterized design allows choosing smaller values to reduce the area and power consumption. Once the rate is determined, the downsampling block (DS) reads input sample data from the FIFO and selectively stores it into its output buffer, as shown in Figure 1.

Fast Fourier Transform Features: The stretch sensor data generally shows a periodic pattern for dynamic activities, such as walking, stairs down and stairs up. This means that FFT coefficients can be utilized to effectively capture this periodicity. To this end, we append two consecutive activity windows, the minimum length required to compose a periodic signal. Then, we take the 64-point FFT of this signal to characterize the frequency spectrum. Among the 64 FFT coefficients, we include the first 16 coefficients in the feature vector since this is sufficient to capture frequencies up to 8.25 Hz, which is higher than the frequency observed in human motion.

Discrete Wavelet Transform Features: The accelerometer data is typically noisy. Therefore, we use the approximation coefficients of the DWT to obtain robust features from the accelerometer data. Analysis on a high-level reference implementation in Python with Keras APIs and Tensorflow-backend shows that approximation coefficients of a single-level DWT are sufficient to capture the acceleration of human activity at 0 to 32 Hz. The proposed HAR engine produces 32 DWT coefficients for a_x, a_z and b_{acc}. Thus, the subsequent DNN classifier uses a total of 96 DWT features each represented as a 16-bit number.

Statistical Features: Statistical features of the sensor data also provide useful information for static activities, such as sitting and standing, without requiring complex processing. We utilize the minimum and maximum of values observed in an activity window for all four streams, i.e., 3-axis accelerometer and stretch data. In addition, we compute the mean of a_z and the variances of a_x, a_y, a_z and b_{acc} using the accelerometer sensor data. These specific features are selected based on our extensive analysis on the Python reference implementation. The details of this step are omitted since feature selection is not the focus of this paper.

3.4 Single-Level Baseline DNN Classifier

The last step of the proposed HAR engine is the DNN classifier, as depicted in Figure 1. To determine the size of the neural network, we performed a design space exploration with one and two hidden layers. Within each of these structures, we varied the number of neurons in the hidden layers. Then, we trained each neural network and evaluated the accuracy. We observe that using two hidden layers with ReLU activation function provides the best accuracy for the baseline classifier. The first hidden layer has 4 neurons while the second hidden layer has 8 neurons. The output layer of the DNN classifier has 8 neurons, one for each activity. We use a linear activation in the output layer and then choose the activity with the maximum value as the output activity. The choice of max function in the output layers allows us to avoid the use of exponents in the softmax function, which is more commonly employed to obtain the classification output. Next, we describe the operation of the DNN and proposed optimizations performed to improve the implementation.

Architecture and Operation: Our DNN architecture consists of two finite state machines. The first state machine governs the state of the overall network (*State-1*), which also corresponds to the state of the first layer (L1). It has four states: *Init*, *WeightLoad*, *Idle* and *Busy*. The second state machine (*State-2*) governs the remaining, i.e., second and output layers, as well as the max function at the output. The DNN operates as follows using these states:

- *Init*: On power-up, *State-1* enters the *init* state, where all registers are reset to their default values.
- *Init* → *WeightLoad*: *State-1* moves to the *WeightLoad* state when it receives an enable signal that indicates new DNN weights are available for loading. Then, the input weights are loaded from an off-chip ROM to corresponding memory for each neuron. In our implementation, there are a total of 596 weights each represented with 16 bits.
- *WeightLoad* → *Idle*: After the weights are loaded, *State-1* enters the *Idle* state. In this state, the DNN classifier waits for an input valid signal that indicates the presence of new input features.
- *Idle* → *Busy*: When a new set of input features is available at the inputs of the DNN classifier, *State-1* moves to the *Busy* state. In this state, the classifier first registers the 120 input features in parallel. Then, the neurons in the first layer process the registered inputs and activate the second layer. Subsequently, the second and output layer are activated one by one to produce the classifier output. Finally, the output flag is raised at completion of the max block.

Optimizations: The DNN classifier in the proposed HAR engine is parameterized to facilitate configurability and scalability. The basic building block is a parameterized neuron module. It takes the number of inputs to the neuron, weights, and features as input parameters. Therefore, this module can be instantiated in all three layers of the DNN with the appropriate parameters. We have further parameterized the multiply-accumulate (MAC) block, the weight memories, ReLU and Max functions to facilitate design space exploration. Since the parameterized neurons are used to construct the hidden layers and the output layer, the architecture of the DNN can be changed easily. For example, only an hour was required to do the necessary changes and verification in going from the two hidden layer architecture used in the baseline HAR engine (Figure 1) to the single hidden layer architecture used in the activity-aware HAR engine presented in Section 4.

4 ACTIVITY-AWARE 2-LEVEL HAR ENGINE

This section presents a novel hierarchical HAR engine using the insights gained from the implementation of the baseline design. Analyzing the layout of the baseline classifier reveals that the DNN classifier and FFT blocks are the two major contributors to the power consumption and area, as detailed in Section 6. These blocks can be avoided for static activities whose complexity is significantly lower than that of dynamic activities. Moreover, it is relatively easy to distinguish static and dynamic activities using a simple two-class classifier. Therefore, we first employ a support vector machine (SVM) classifier to determine if the activity is static (*lie down, sit, stand*) or dynamic (*jump, stair down, stair up, walk*), as shown in Figure 2. If the outcome is static, then we invoke a relatively simpler decision tree to further classify the activity. Otherwise, we still employ a DNN classifier, albeit a smaller one compared to the baseline design, to maintain high accuracy and facilitate future online learning. We note that its energy consumption overhead will be small since it will be powered down when it is not active. Our modular and parameterized baseline design enables us to reuse the preprocessing and most of the feature generation blocks, as shown in Figure 2. Therefore, we focus on the new blocks and the overall operation in this section.

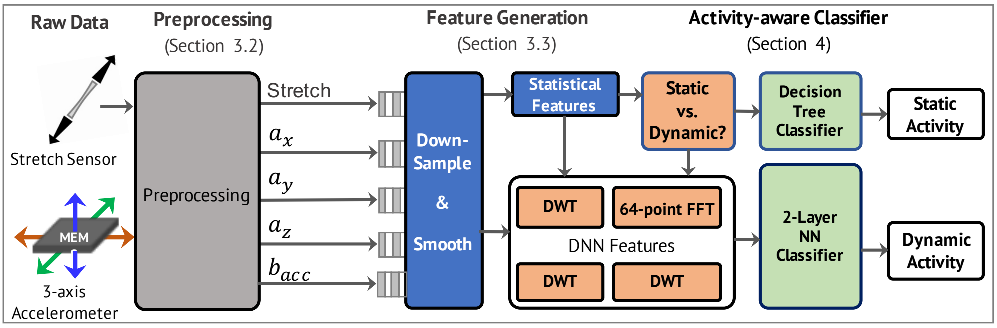

Fig. 2. Architecture of the activity-aware HAR engine

4.1 Two-Class SVM Classifier

The first step of the 2-level activity-aware HAR engine is to differentiate between static and dynamic activities. We use an SVM to classify between the two types of activities since it is easier to implement using fewer resources than a DNN classifier. Moreover, the SVM classifier uses exclusively statistical features to avoid FFT and DWT computations. These features include the minimum, maximum, mean, and variance of the stretch sensor, 3-axis acceleration (a_x, a_y, a_z), as well as the body acceleration (b_{acc}). In addition, we also include the length of the window as the final feature. Using these features, we train the SVM with our user data. Once we obtain the weights of the SVM, the activity type is evaluated at runtime using the following equation:

$$y = b + \sum_{i=1}^{N} \beta_i x_i \tag{2}$$

where b is the value of the bias, x_i are the features, and β_i is the weight for the i^{th} feature. Using this evaluation, the activity is classified static if $y < 0$, and dynamic otherwise. Depending on the output of the SVM classifier, we use a decision tree or a DNN to further classify the activity, as described in the following sections.

4.2 Decision Tree (DT) Classifier for Static Activities

If the level-1 SVM classifier marks an activity as static, we identify the activity using a decision tree, which can be easily implemented in hardware using comparators. Besides its simplicity, the DT classifier provides greater than 95% accuracy for static activities, as shown in the experimental results.

The DT classifier uses the same features as the SVM classifier, i.e., the statistical features for the accelerometer and stretch sensor data, as shown in Figure 2. Reusing the features allows us to compute them only once and optimize the power consumption. Furthermore, the neural network, FFT, and DWT blocks remain in low-power states leading to additional power savings. The decision tree is implemented as a series of comparators, where each node of the tree consists of a comparator. Depending on the output of the comparator, we choose the next branch of the tree. This process continues until we reach a leaf node and the activity is identified.

4.3 DNN Classifier for Dynamic Activities

A DNN classifier is used to identify the activity when the output of the level-1 SVM classifier indicates a dynamic activity. The DNN classifier in the activity-aware 2-level HAR engine has to identify only four activities and the transitions between them. Therefore, the DNN used herein is smaller compared to the one used in the baseline HAR engine. Specifically, the DNN classifier has one hidden layer with 4 neurons and an output layer with 5 neurons. Similar to the baseline design, we use ReLU activation in the hidden layer and softmax classification in the output layer. The feature input to the DNN classifier is the same as the input to the baseline DNN classifier (i.e., the same 120 features). To generate these features, we enable the DWT and FFT blocks whenever the SVM classifier outputs a dynamic activity. Once the features are generated, we evaluate the DNN to obtain the final activity classification.

5 LOW-POWER OPTIMIZATIONS

One of the most important goals of the proposed HAR engines is to operate within the harvested energy budget of wearable devices [8]. This section presents the low-power optimization techniques used in our design that will help in enabling operation under harvested energy budget.

5.1 Clock and Data Gating

Human activities typically occur in the order of a few Hertz [27]. We leverage this information to deactivate the part of blocks which are not used all the time. For example, feature generation starts after segmentation and downsampling are completed. Similarly, there is a data dependency from feature generation to classification, as shown in Figure 3. In total, there are three major dependencies:

- Stretch and accelerometer downsampling (i.e., Stretch DS and Accel. DS in Figure 3) depend on the completion of segmentation
- Feature generation dependency is twofold. FFT computation and stretch statistics depend on stretch sensor downsampling. DWT computation and accelerometer statistics depend on accelerometer downsampling.
- Classifier depends on feature generation

We leverage these dependencies to design a custom clock gating solution for HAR applications. The clocks of the stretch and accelerometer downsampling blocks become active only after a valid segment is identified. Since there are fewer stretch sensor data samples than the accelerometer, stretch downsampling finishes earlier and enables the FFT feature generation block (orange line). After downsampling the accelerometer samples, the rest of the feature generation blocks (e.g., DWT) are enabled. Once all features are generated, the clock of the classifier blocks is enabled (green line).

Figure 4 shows the clock gating logic for the segmentation and downsampling blocks. The control signals, called ("output_valid"), are asserted to notify the downstream block when the preceding operation is complete. This signal is connected to the "set" input of an SR-Latch. When it is asserted, the output Q of the latch becomes high. Consequently, the gated clock "gclk" enables the operation

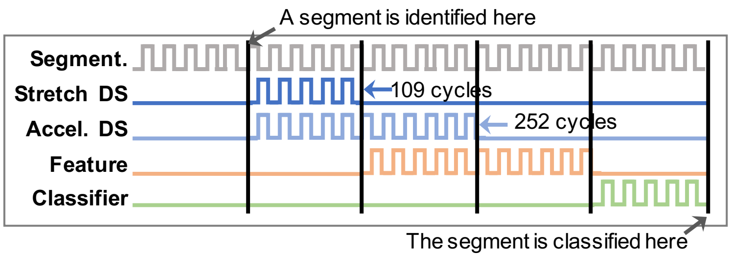

Fig. 3. A representative timing diagram for the activation of the blocks in the proposed HAR engine. The active times of Feature and Classifier clocks are detailed in Table 9.

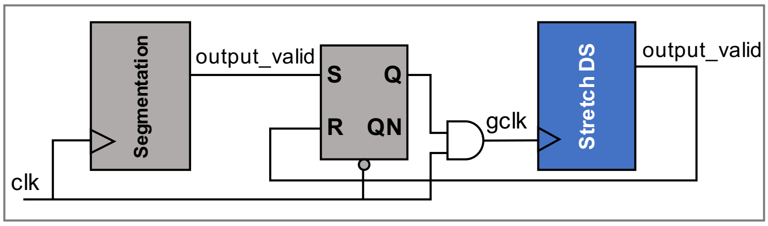

Fig. 4. Clock gating control logic

of the downsampling block. The "output_valid" of the downsampling block is connected to the "reset" input of the same latch. In this way, the gated clock stops when "output_valid" becomes high, i.e., when downsampling is completed. Note that we use a negative latch to prevent any glitches and additional delay in the gated clock.

Clock Gating Implementation Overhead: The clock gating logic uses only an SR-Latch and an AND gate, which introduces negligible overhead. The latch introduces a half-cycle delay before the operation of the next logic begins. To compensate for this and to not lose any data, the outputs of the left-hand logic are registered before they are passed on to the next logic.

The proposed clock gating logic is implemented for all dependencies in the design presented earlier. As a result, we end up with one global clock and four gated clocks for both the baseline and activity-aware HAR engines. Figure 5 illustrates which blocks run on different clocks with different colors. Note that these clocks are gated versions of the same global clock, so their frequencies are the same.

5.2 Power Gating

Clock gating technique is useful for reducing the dynamic power consumption, while power gating can reduce both leakage and dynamic power consumption. Since the feature generation and classification blocks stay mostly idle, power gating is a promising power optimization technique. To this end, we divide the proposed HAR engine into two power domains, as illustrated in Figure 5. The first power domain (PD1) contains the preprocessing and segmentation blocks, which are always on. The second power domain (PD2) integrates the feature generation and classification blocks, which become active intermittently. In this way, these blocks can turn on only after a segment is identified and turned off once the activity is classified.

The power gating functionality is implemented using a power control unit (PCU). Since the PCU must be always on, it operates on PD1 and global clock. The PCU works on the same basic

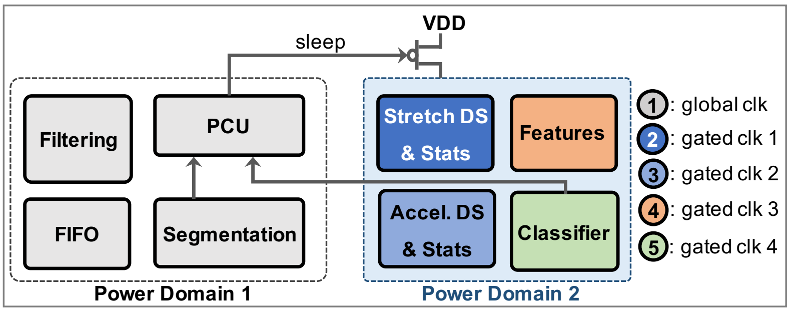

Fig. 5. Power and clock domains in the proposed design

principles as the clock gating logic shown in Figure 4. By default, the second power domain is turned off during preprocessing and segmentation. The "output_valid" signal of the segmentation block is the first input to the PCU. It is asserted when a valid segment is ready. When this input is received, the SR-latch inside the PCU turns on PD2 by driving the sleep transistor, as shown in Figure 5. Then, PD2 stays active until the classifier block output becomes valid.

Power Gating Implementation Overhead: Multiple power domains introduce three distinct overheads. First, the sleep transistor introduces additional leakage current, which can offset the potential power savings, if it is significant. To minimize the leakage of the sleep transistor, we employ high-threshold-voltage low-leakage PMOS header transistors. The high threshold voltage transistors typically have leakage power in the order of nanowatts[1]. Since the leakage power of the proposed HAR accelerator is in the order of microwatts, power gating can achieve significant power savings by lowering the leakage current. The second overhead is the wake-up time, which is typically less than one microsecond. The proposed HAR accelerator receives new sensor samples every few milliseconds when operating at 100–250 Hz sampling frequencies. Therefore, the wake-up time is negligible for the proposed HAR engine. Finally, the PCU and the sleep transistor increase the total area of the design. In our implementation, the logic required for controlling the sleep transistor is a single SR-Latch, which represents a negligible area overhead. Furthermore, the sleep transistor also incurs less than 5% area overhead, which is small when compared to the potential savings in the leakage power. In summary, using two power domains enables significant power savings with a small overhead.

5.3 Use of Low-Power Classifiers

In the course of daily life, static activities occur more frequently than dynamic activities. To estimate the average distribution of activities, we performed an analysis on human activities using the American Time Use Survey (ATUS) [43], which is conducted by the United States Census Bureau to record how people spend their time during a day. According to this survey, people spend almost 84% of the time in static activities, such as lie down, sit, and stand. Dynamic activities, such as walking, running, and stairs up/down, constitute only about 16% of the total time.

Our activity-aware 2-level classifier exploits the distribution of activities to enable additional power and energy savings. First, the DNN, DWT, and FFT blocks are activated only for dynamic activities. For example, if the SVM block classifies an activity as static, these blocks remain clock gated. Consequently, they are activated less frequently compared to the baseline classifier. Furthermore, the DNN block uses fewer resources as detailed in Section 6.2 because it does not need to classify static activities. The benefits of using the activity-aware classifier in terms of power and energy consumption are evaluated in Section 6.

6 EXPERIMENTAL EVALUATION

6.1 Experimental Setup

Design Tools and Hardware Technology: We first design the proposed HAR engine using Verilog Hardware Description Language (HDL). Then, we synthesize it with TSMC 65 nm low power (LP) technology using Synopsys Design Compiler (DC). To obtain the layout and the area of the design, we perform automatic placement and routing using Cadence Innovus. Finally, we use Synopsys PrimeTime to obtain the timing and power consumption of each block and the entire design. In addition to the low-power optimizations presented in Section 5, we also utilize the power

[1]https://blogs.synopsys.com/magicbluesmoke/files/2007/10/sleep_transistor_sizing.pdf

optimization options available in the Synopsys Design Compiler to further optimize the power consumption of the proposed HAR engine. The same optimization options are used for all designs for a fair comparison.

User Studies: To evaluate the accuracy of the designed HAR engines, we collected user data from 22 users using the accelerometer in a TI Sensortag IoT [42] device and a stretch sensor [34]. The IoT device is connected to a host smartphone application through a Bluetooth interface and the data is recorded on the smartphone. In each experiment, the users perform the target activities. Then, the segmentation algorithm divides the activities into non-uniform length activity windows and labels them. We use this labeled data to extract the features and to train the DNN classifiers. Moreover, raw sensor data from the user studies are fed to the HAR engine to segment the windows and perform the activity classification.

Training, Cross-validation, and Test data: Data samples from 4 randomly selected users are reserved exclusively for testing. Then, samples from the remaining 18 users are divided into 60% training, 20% cross-validation, and 20% additional test data. Overall, 37% of the test data comes from 4 unknown users.

6.2 Design Area

6.2.1 Baseline HAR Engine. Figure 6 shows the layout of the baseline HAR engine, which has a total area of 1.353 mm^2. While optimizing the floorplan, the order of the blocks is matched to the logic flow described in Figure 1. The area breakdown in Figure 8 shows that the FFT block has the largest area. This is expected as the FFT block is the most compute-intensive block in the design. Among other blocks, most of the area is occupied by blocks that have registers to store their input or output data. For example, the FIFO block occupies 21% of the total area. This is also expected since it has to store the filtered data until a new activity window is detected by the segmentation block. Similarly, the DNN block incurs the overhead of storing the weights and performing MAC operations. In contrast, the low pass filter and segmentation blocks take up a smaller area since they do not have to store a significant amount of data for their computations.

6.2.2 Activity-Aware 2-Level HAR Engine. Figure 7 shows the layout of the 2-level HAR engine, which shows a total cell area of 1.357 mm^2. We observe that the layout closely resembles the baseline classifier except for the feature generation and neural network blocks. This is because both classifiers share the blocks for filtering, segmentation, and downsampling. The activity-aware 2-level design has additional blocks for the support vector machine classifier and decision tree for static activities. However, the total area of the 2-level classifier is only 0.3% more than the baseline classifier because it uses a smaller DNN with fewer neurons for dynamic activities. The 2-level classifier with a slightly larger area allows us to significantly improve the accuracy and lower the power consumption, as we show in the following sections.

Figure 9 shows the area breakdown of the activity-aware 2-level HAR engine. Similar to the baseline design, the FFT block consumes the largest area while the neural network block has a smaller area in the 2-level classifier due to its smaller size. Furthermore, the SVM and DT blocks consume a negligible portion of the total area despite their significant power and accuracy benefits.

6.3 Accuracy Evaluation

6.3.1 Baseline HAR Engine. We start the accuracy analysis of the baseline HAR engine by generating the feature vectors for each activity window in our dataset. To maintain compatibility with the hardware implementation, the features are stored in a 16-bit integer format. This feature data is then supplied to a neural network training algorithm implemented in Python with Keras APIs [12]

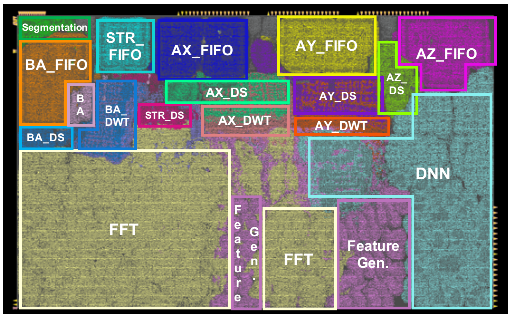

Fig. 6. Floorplan of the baseline HAR engine

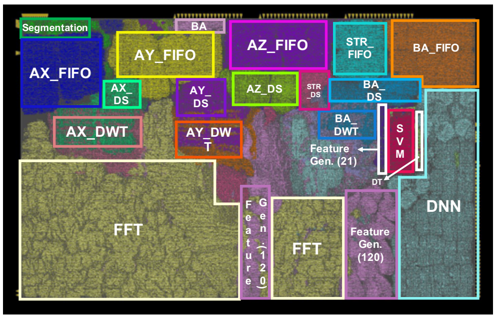

Fig. 7. Floorplan of the activity-aware 2-level HAR engine

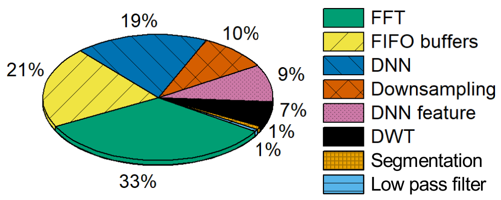

Fig. 8. Area of the major blocks used in the baseline HAR engine

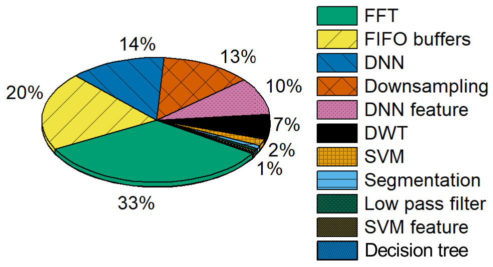

Fig. 9. Area of the major blocks used in the hierarchical 2-level HAR engine

and Tensorflow-backend [1] using the parameters summarized in Table 2. These parameters are determined by sweeping them and evaluating the accuracy. For example, the batch size is incremented from 25 to 100 in steps of 25. Of these, a batch size of 50 gives the best accuracy for our dataset. Similarly, we sweep the number of training epochs from 100 to 500. 200 training epochs are used in the experiments since this is sufficient to achieve good accuracy while avoiding overfitting.

At the end of the training, we obtain the weights of the network in the floating point format. Since our hardware implementation uses 16-bit integer precision, we uniformly quantize the weights and activations to 16 bits using the approach in [24]. For each layer of the neural network, we first find the weight with the maximum magnitude $W_{\max}$. Then, the quantization factor Δ_q is obtained as:

$$\Delta_q = \frac{2W_{\max}}{2^{16}} \tag{3}$$

The uniform quantization ensures that zero in floating-point precision is mapped to zero in integer precision as well. We use the quantization factor of each layer to obtain the quantized weights of

Table 2. Parameters used for deep neural network training

Optimizer	Adam [21]
Loss function	Categorical cross-entropy
Initial learning rate	0.001
Epochs	200
Batch size	50

the neural network. After quantizing the weights, we use the 16-bit integer features to quantize the activations in the neural network. Finally, we evaluate the accuracy of the quantized neural network for all the activity windows in our dataset.

Table 3 shows the confusion matrix for the baseline HAR engine. It contains one row and column for each activity classified in this work. The rows represent the true activity whereas the columns represent the activity classified by the proposed baseline HAR engine. The diagonal entries in each the confusion matrix shows the number of instances that are classified correctly. The baseline DNN classifier is able to recognize all the activities with few or no misclassifications. For example, the walk activity is classified correctly 1913 times out of 2007 activity windows, leading to 95% classification accuracy. We also observe that the walk activity is misclassified as either jump, stairs down or transition. This happens since the jump and stairs down pattern of some users are similar to that of walk. All the instances of the lie down activity are classified correctly, leading to a 100% recognition accuracy. The lowest accuracy of classification is observed for transitions between the various activities. This is expected since the transitions typically contain features of two different activities, such as in stand to sit transition, which makes it harder for the classifier to identify them with a high accuracy. In summary, the baseline DNN classifier achieves an accuracy greater than or equal to 93% for all the activities used in this work. Note that additional activities could degrade the accuracy, especially if they are close to the existing activities. More complex classifiers and feature may be needed with increasing number and complexity of target activities.

Table 3. Confusion matrix for the baseline classifier

	Jump	**L**ie Down	**S**it	**St**and	**W**alk	**S**tairs **U**p	**S**tairs **D**own	**T**ran-sition
J	**442**	0	0	0	5	0	5	6
L	0	**474**	0	0	0	0	0	0
S	0	0	**665**	26	0	0	0	5
St	0	0	16	**576**	1	0	0	27
W	31	0	1	10	**1913**	0	10	42
SU	0	0	0	0	1	**101**	6	1
SD	0	0	0	0	1	1	**97**	1
T	7	2	7	14	14	4	0	**229**

Evaluation with the Opportunity Dataset [40]: In addition to our in-house data, the accuracy of the proposed HAR engine is also evaluated using the publicly available Opportunity dataset [40], which includes data from 4 users for a range of daily activities. Since our work focuses on ambulatory activities, we extract the corresponding data from the Opportunity dataset. Furthermore, we use only the accelerometers mounted on the right knee and the shoe since these setups are closest to the sensors used in our experiments. The data is divided into windows of approximately two seconds to compute the DWT and statistical features for each window. Finally, the DNN is trained with the same structure as our baseline neural network classifier using the extracted features. The proposed DNN classifier achieves an overall accuracy of 95% for the locomotion activities in the Opportunity dataset. This result shows that the proposed features and DNN classifier can classify activities from multiple datasets accurately.

6.3.2 *Activity-Aware 2-Level HAR Engine.*

First-Level SVM Classifier: The activity-aware 2-level HAR engine recognizes the activities in two phases, as described in Section 4. The SVM classifier employs only the statistical features to determine if an activity is static or dynamic to minimize power consumption. The SVM parameters are determined by using the Statistical and Machine Learning Toolbox in MATLAB. We perform

10-fold cross-validation during training to ensure that the classifier is robust. After finding the weights of the SVM classifier, they are converted to 16-bit integer precision using Equation 3. Then, we obtain the confusion matrix for the SVM classifier using the integer weights, as shown in Table 4.

Our SVM implementation classifies only 38 windows out of 4740 activity windows incorrectly. More specifically, it misclassifies only 9 static and 29 dynamic windows, as shown in Table 4. Hence, it achieves more than 98% overall accuracy.

Table 4. Confusion matrix for the level 1 SVM classifier. The static and dynamic activities are classified with over 99% accuracy.

	Static	Dynamic
Static	**1781**	9
Dynamic	29	**2921**

Second-Level Decision Tree for Static Activities: Static activities are classified using a decision tree, as described in Section 4.2. We train the DT classifier in Weka [18] using 16-bit integer features for all the activity windows in the static class. Table 5 shows the confusion matrix for the three static activities in our data set. The DT classifier achieves high accuracy for all the three activities. Overall, it classifies only 13 out of 1790 activities incorrectly. Furthermore, it has better accuracy than the DNN classifier used in the baseline HAR engine. For instance, the number of misclassifications for the stand activity reduces from 44 to 8, which translates to an accuracy improvement from 93% to 99%. As a result, the two-level design has both accuracy and power consumption benefits.

Table 5. Confusion matrix for the activity-aware HAR engine for static activities

	Lie Down	Sit	Stand
Lie Down	**473**	0	1
Sit	0	**691**	4
Stand	8	0	**612**

Second-Level Neural Network for Dynamic Activities: The activity-aware HAR engine uses a neural network with a single hidden layer to classify individual dynamic activities, as explained in Section 4.3. The neural network uses the same features and training settings as the baseline DNN classifier. Similar to the baseline DNN, we quantize the weights and the activations using Equation 3.

The confusion matrix for the dynamic activities is shown in Table 6. We observe that the number of misclassifications for all the activities is similar to or fewer than the baseline DNN classifier. The largest improvement is seen for transitions where the number of mistakes decreases from 48 to 25,

Table 6. Confusion matrix for the activity-aware HAR engine for dynamic activities

	Jump	**W**alk	**S**tairs **U**p	**S**tairs **D**own	**T**ransition
J	**445**	8	0	1	4
W	19	**1937**	2	9	40
SU	1	1	**105**	1	1
SD	0	0	0	**99**	0
T	5	14	1	5	**252**

Table 7. Power consumption summary for the baseline HAR engine at f=100kHz, V=1.0V

Operation Mode	Block	Dynamic Power (µW)	Leakage Power (µW)	Total Power (µW)
Data Collection & Preprocessing (always ON)	Filtering	0.49	0.05	0.54
	Segmentation	0.43	0.05	0.48
	FIFO	12.10	1.16	13.26
Classification (once per activity)	Downsample	4.39	0.75	5.14
	DWT	2.49	0.42	2.91
	FFT	11.90	2.61	14.51
	DNN Feature Merge	1.77	0.75	2.52
	DNN	10.14	1.16	11.30
Total (Data Collection + Classification)		43.71 (13.02+30.69)	6.95 (1.26+5.69)	50.66 (14.28+36.38)

which is a drop of about 48%. The overall accuracy of the proposed second-level neural network classifier is 96%. In contrast, a decision tree classifier for dynamic activities could achieve only 90% accuracy. In summary, the activity-aware 2-level HAR engine is able to improve the classification accuracy while also reducing the average power consumption.

6.4 Power Consumption Evaluation

Power and energy consumption are among the most important evaluation metrics for the HAR engine. Therefore, we perform a detailed analysis of the power consumption of each major component in the proposed design. Our analysis assumes 100 kHz operating frequency, which provides sufficient performance (4.19 ms operation per activity in the baseline HAR engine), as described in Section 6.5. To analyze the power consumption, we first generate the switching activity data of the design using sensor data from our user studies. The switching activity file is then used to calculate the power consumption of the major blocks in the design. The power consumption values reported in this section include savings achieved by the clock gating described in Section 5. We estimate up to 30% additional savings by power gating the feature generation and DNN blocks using two power domains until a new activity window is detected.

6.4.1 Baseline HAR Engine. Table 7 summarizes the average power consumption of the major blocks of the baseline HAR engine. Power consumption is divided into two distinct parts based on the operating mode of each block. The first row shows the power consumption of the always-on preprocessing and FIFO blocks. The majority of the power in the preprocessing block is attributed to active power. Since this is mainly due to the FIFO memories in the preprocessing block, this power consumption can be further reduced by optimizing the memory design. Rest of the rows of the table show the power consumption of blocks that are executed only once per activity. Of these blocks, the FFT block has the highest power consumption as expected. This aligns with our earlier observation that the FFT block has the highest area among all the blocks. The DNN block has a higher active power consumption due to the multiplications involved in each neuron. The total power consumption for classifying the activity after the prepossessing blocks complete is about 36.4 µW. This represents about 325× reduction in processing power when compared to implementations on a programmable microcontroller.

6.4.2 Activity-Aware 2-Level HAR Engine. This section analyzes the power consumption of the activity-aware 2-level HAR engine. The power values for the always-on blocks remain unchanged since they are reused. The power consumption breakdown of the remaining blocks is summarized in Table 8.

Table 8. Power consumption summary for the activity-aware HAR engine at f=100kHz, V=1.0V

Operation Mode		Block	Dynamic Power (µW)	Leakage Power (µW)	Total Power (µW)
Classification (once per activity)	Common	Downsample	3.33	0.84	4.17
		SVM Feature Input	0.47	0.04	0.51
		SVM	0.47	0.15	0.62
	Static	Decision Tree	0.24	0.02	0.26
	Dynamic	DWT	1.96	0.43	2.39
		FFT	9.64	2.66	12.3
		DNN Feature Input	1.42	0.75	2.17
		DNN	7.60	0.85	8.45
Total	Static (Data Collection + Classification)		17.24 (12.73+4.51)	2.29 (1.24+1.05)	19.53 (13.97+5.56)
	Dynamic (Data Collection + Classification)		37.62 (12.73+24.89)	6.96 (1.24+5.72)	44.58 (13.97+30.61)

As described in Section 4, the power consumption of this design depends on the activity, i.e., whether the activity is static or dynamic. The first part of the table shows the power consumption of the blocks that are common to both static and dynamic activities. These blocks include Downsample and the SVM classifier. We observe that the SVM classifier has a negligible power consumption of 0.62 µW, while the Downsample block has similar power consumption as the baseline HAR engine. The "Static" part of the table shows the blocks that are used when a static activity is detected by the SVM classifier. Since the decision tree for static activities employs the same features as the SVM classifier, it is the only component contributing to the power consumption, i.e., there is no need to generate other features. As we can see in Table 8, the DT consumes only 0.26 µW of power. Finally, the "Dynamic" part of the table shows the power consumption of the engine when a dynamic activity is identified by the SVM classifier. This includes computation of the detailed features required for the neural network and the neural network execution itself. As expected, the total power for dynamic activities is higher than the power consumption for static activities.

Several blocks, such as Downsample, have lower power consumption when used in the activity-aware 2-level HAR engine compared to the baseline design. This difference stems from two reasons. First, these two designs are synthesized separately as a whole, as opposed to integrating blocks as black boxes. Hence, the synthesized block, their output ports, and lengths of the buses they drive have differences. For example, the output of the Downsample block connects to the SVM block in the 2-level HAR engine. In contrast, it connects to all the feature generation blocks, including FFT and DWT, in the baseline design. Second, the power consumption is found using the VCD files in Primetime tool. Since most of the blocks in the 2-level activity-aware design are activated less frequently, they end up having a smaller switching factor, hence, lower power consumption.

In summary, the activity-aware classifier consumes 5.56 µW for static activities and 30.61 µW for dynamic activities once a segment is identified. Including the power consumption of the sensor and data communication, this leads to 1.3 mW, which is 10 times lower than the embedded system solutions.

Voltage Scaling: Once an activity window is detected, the proposed HAR engine takes 421 cycles to generate the features and classify the activity. Hence, operating at as low as 42 kHz provides around 10 ms processing window, which is comparable to the sampling time of the sensors. This slack enables us to lower the supply voltage without a noticeable difference in performance. We

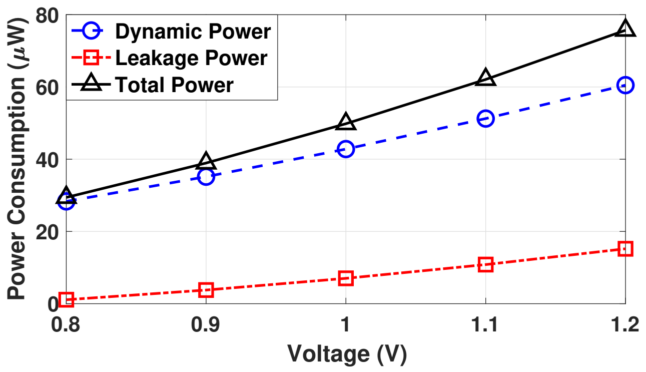

Fig. 10. Power consumption as a function of voltage

exploited this observation by sweeping the supply voltage and measuring the resulting power consumption. The power consumption at 1.2 V is 75 μW, as shown in Figure 10. Operating at 1.0 V and 0.8 V decrease the power consumption to 45 μW and 30 μW, respectively.

Peak Power Consumption Evaluation: Our goal is to operate the HAR engine using ambient energy harvesting. Therefore, it is also important to evaluate the peak power consumption of the design to ensure that the power consumption is within the capabilities of energy-harvesting technologies. Figure 11 compares the total power consumption of the two classifiers for the two types of activity windows, static and dynamic. Specifically, the figure shows a static activity at t = 376.9 s and a dynamic activity at t = 379.8 s. The black line shows the power consumption of the baseline single level HAR engine. It has the same instantaneous power consumption for both activity types. Furthermore, it has a peak power consumption of almost 51 μW. This can be limiting for energy-harvesting devices since they have to support a peak power of 51 μW for each activity window. In contrast to this, the activity-aware classifier, shown with a red line, has a peak power consumption of only 19.5 μW for static activities, as seen in the zoomed in portion inside the Figure 11. Therefore, the peak power that energy-harvesting devices need to provide is reduced significantly, especially for static activities.

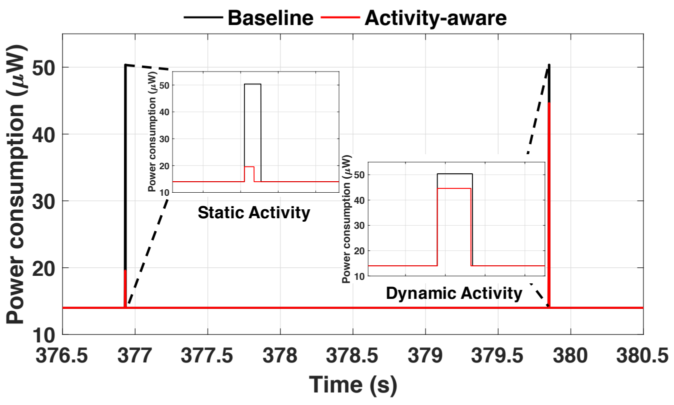

Fig. 11. Comparison of power consumptions of the Baseline and Activity-aware HAR Engines

6.5 Performance Evaluations

This section presents the execution time of each block in the proposed HAR engines. The time scales of human activities and sampling data rates are much smaller than the maximum operating frequency we can achieve with the commercial TSMC 65 nm technology. In contrast, the power consumption is of extreme importance. Both the baseline and activity-aware 2-level HAR engines can operate comfortably at 1 MHz, which is much higher than required. Therefore, we summarize the number of cycles taken by each major block in the rest of this section.

The always-on preprocessing and segmentation blocks take one cycle for processing each incoming sample. Once the segmentation block marks a segment, the corresponding blocks for each classifier are activated. The latencies for these blocks are summarized in Table 9. We divide the table into three parts. The first part, denoted by "Common" includes the blocks that are common to both baseline and activity-aware engines. The second part, denoted by "Baseline Classifier" includes the blocks used only in the baseline classifier. Finally, the third part of the table lists the latencies for blocks that are exclusive to the 2-level classifier.

The Downsample block for the accelerometer data has the highest latency while the DWT block has the lowest latency. The Downsample block incurs a higher latency because it has to go through all the samples in a segment serially before generating its output. Moreover, the calculation of statistical features is integrated into the Downsample block, which adds a few cycles of additional execution. We also observe that the Downsample block for the stretch sensor takes fewer cycles to execute. This can be attributed to the fact that the sampling rate of the stretch sensor is lower than that of the accelerometer. Once the Downsample blocks complete their execution, the DWT and FFT blocks execute in parallel leading to their low execution latency of 4 and 21 cycles, respectively. The FFT and DWT blocks are executed for the baseline HAR engine independent of the class of activity, whereas for the activity-aware engine they are executed for dynamic activities only.

The last step in the baseline HAR engine is to evaluate the activity classification using the three-layer DNN, which takes 163 cycles. For the activity-aware engine, we first evaluate the SVM classifier to distinguish between static and dynamic activities. If the activity is static, we execute the decision tree with a latency of 2 cycles. Otherwise, we execute the FFT, DWT, and the 2 layer DNN, which take 166 cycles in total.

The total latency is not simply summing every block up since there are some overlaps. In summary, including data collection blocks, the total latency for the baseline HAR engine is 419 cycles, which results in 4.19 ms for the classifying the activity at 100 kHz operating frequency. Total latencies for dynamic activity and static activity in the activity-aware HAR engine are 421 cycles (4.21 ms @100 kHz) and 257 cycles (2.57 ms @100 kHz), respectively. Consequently, the time

Table 9. Summary of latency of the blocks in the baseline HAR engine and the activity-aware HAR engine

Usage	Block	Latency (cycles)
Common	Acc. Downsample	252
	Stretch Downsample	109
	FFT	21
	DWT	4
Baseline HAR engine	3 Layer DNN	163
Activity-aware HAR engine	SVM	3
	Decision Tree	2
	2 Layer DNN	145

taken for activity recognition is negligible compared to the time required for data collection (in the order of a few seconds).

7 CONCLUSIONS AND FUTURE WORK

This paper presented the first hardware accelerator that integrates all aspects of human activity recognition. We first designed a baseline HAR engine following the most commonly used single classifier for all the activities. Then, we exploited the nature of human activities to design an activity-aware 2-level HAR engine that identifies activities in two steps. This approach improves the classification accuracy while reducing the power consumption of the HAR engine. We implemented both HAR engines using a commercial 65 nm process technology. Our extensive post-layout simulations show that the proposed HAR engines are able to reduce the power consumption by about two orders of magnitude compared with conventional programmable solutions. Further optimization in power consumption and area can be obtained by using an 8-bit quantization for the DNN. Our preliminary results show that it can achieve up to 7% savings in power consumption and a 6% reduction in area. However, 8-bit quantization also reduces the robustness of classification accuracy. Therefore, our future work will consider quantization-aware training approaches to achieve lower power and smaller area.

REFERENCES

[1] Martín Abadi et al. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. [Online] http://tensorflow.org/, accessed 31 July 2019.

[2] Muhammad Arif, Mohsin Bilal, Ahmed Kattan, and S Iqbal Ahamed. 2014. Better Physical Activity Classification Using Smartphone Acceleration Sensor. *J. of Med. Syst.* 38, 9 (2014), 95.

[3] Ferhat Attal et al. 2015. Physical Human Activity Recognition using Wearable Sensors. *Sensors* 15, 12 (2015), 31314–31338.

[4] Akin Avci, Stephan Bosch, Mihai Marin-Perianu, Raluca Marin-Perianu, and Paul Havinga. 2010. Activity Recognition using Inertial Sensing for Healthcare, Wellbeing and Sports Applications: A survey. In *Proc. Int. Conf. on Arch. of Comput. Syst.* 1–10.

[5] Ganapati Bhat, Kunal Bagewadi, Hyung Gyu Lee, and Umit Y. Ogras. 2019. REAP: Runtime Energy-Accuracy Optimization for Energy Harvesting IoT Devices. In *Proc. of Annual Design Autom. Conf.* 171:1–171:6.

[6] Ganapati Bhat, Ranadeep Deb, Vatika Vardhan Chaurasia, Holly Shill, and Umit Y. Ogras. 2018. Online Human Activity Recognition using Low-Power Wearable Devices. In *Proc. Int. Conf. on Comput.-Aided Design.* 72:1–72:8.

[7] Ganapati Bhat, Ranadeep Deb, and Umit Y Ogras. 2019. OpenHealth: Open Source Platform for Wearable Health Monitoring. *IEEE Design & Test* (2019). To be published, doi: 10.1109/MDAT.2019.2906110.

[8] Ganapati Bhat, Jaehyun Park, and Umit Y Ogras. 2017. Near-Optimal Energy Allocation for Self-Powered Wearable Systems. In *Proc. Int. Conf. on Comput.-Aided Design.* 368–375.

[9] Judit Bort-Roig, Nicholas D Gilson, Anna Puig-Ribera, Ruth S Contreras, and Stewart G Trost. 2014. Measuring and Influencing Physical Activity With Smartphone Technology: A Systematic Review. *Sports Medicine* 44, 5 (2014), 671–686.

[10] Wouter Bracke, Patrick Merken, Robert Puers, and Chris Van Hoof. 2006. A 1 cm^3 Modular Autonomous Sensor Node For Physical Activity Monitoring. In *2006 Ph.D. Research in Microelectronics and Electronics.* 429–432.

[11] Yufei Chen and Chao Shen. 2017. Performance Analysis of Smartphone-Sensor Behavior for Human Activity Recognition. *IEEE Access* 5 (2017), 3095–3110.

[12] François Chollet et al. 2015. Keras. [Online] https://keras.io, accessed 31 July 2019.

[13] Ana Lígia Silva de Lima et al. 2017. Feasibility of Large-Scale Deployment of Multiple Wearable Sensors in Parkinson's Disease. *PLOS One* 12, 12 (2017), e0189161.

[14] DMI International Distribution Ltd. 2017. Curved Lithium Thin Cells. [Online] http://www.dmi-international.com/data%20sheets/Curved%20Li%20Polymer.pdf Accessed 31 July 2019.

[15] Matthias Geisler et al. 2017. Human-Motion Energy Harvester for Autonomous Body Area Sensors. *Smart Materials and Structures* 557, 1 (2017), 012024.

[16] Norbert Győrbíró, Ákos Fábián, and Gergely Hományi. 2009. An Activity Recognition System for Mobile Phones. *Mobile Networks and Appl.* 14, 1 (2009), 82–91.

[17] Mostafa Haghi, Kerstin Thurow, and Regina Stoll. 2017. Wearable Devices in Medical Internet of Things: Scientific Research and Commercially Available Devices. *Healthcare Informatics Research* 23, 1 (2017), 4–15.

[18] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. 2009. The WEKA Data Mining Software: An Update. *ACM SIGKDD Explorations Newsletter* 11, 1 (2009), 10–18.

[19] Nils Y. Hammerla, Shane Halloran, and Thomas Plötz. 2016. Deep, Convolutional, and Recurrent Models for Human Activity Recognition Using Wearables. In *Proc. Int. Joint Conf. on Artificial Intell.* 1533–1540.

[20] Dustin A Heldman et al. 2017. Telehealth Management of Parkinson's Disease Using Wearable Sensors: An Exploratory Study. *Digital Biomarkers* 1, 1 (2017), 43–51.

[21] Diederik P Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. In *Proc. Int. Conf. on Learning Representations.* 1–13.

[22] Morwenna Kirwan, Mitch J Duncan, Corneel Vandelanotte, and W Kerry Mummery. 2012. Using Smartphone Technology to Monitor Physical Activity in the 10,000 Steps Program: A Matched Case–Control Trial. *J. of Med. Internet Research* 14, 2 (2012).

[23] Alicia Klinefelter et al. 2015. 21.3 A 6.45 μW Self-Powered IoT SoC with Integrated Energy-Harvesting Power Management and ULP Asymmetric Radios. In *Proc. IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. of Techn. Papers.* 384–385.

[24] Raghuraman Krishnamoorthi. 2018. Quantizing Deep Convolutional Networks for Efficient Inference: A Whitepaper. *arXiv preprint arXiv:1806.08342* (2018).

[25] Jennifer R Kwapisz, Gary M Weiss, and Samuel A Moore. 2011. Activity Recognition Using Cell Phone Accelerometers. *ACM SigKDD Explorations Newsletter* 12, 2 (2011), 74–82.

[26] Oscar D Lara and Miguel A Labrador. 2013. A Survey on Human Activity Recognition using Wearable Sensors. *IEEE Commun. Surveys & Tut.* 15, 3 (2013), 1192–1209.

[27] Baihua Li and Horst Holstein. 2004. Perception of Human Periodic Motion in Moving Light Displays - A Motion-Based Frequency Domain Approach. *Interdisciplinary J. of Artificial Intell. and the Simulation of Behaviour (AISBJ)* 1, 5 (2004), 403–416.

[28] Xin Liu et al. 2012. An Ultra-Low Power ECG Acquisition And Monitoring ASIC System For WBAN Applications. *IEEE J. on Emerg. and Sel. Topics in Circuits Syst.* 2, 1 (2012), 60–70.

[29] Reza Lotfian and Roozbeh Jafari. 2013. An Ultra-Low Power Hardware Accelerator Architecture for Wearable Computers using Dynamic Time Warping. In *Proc. Conf. on Design, Autom. and Test in Europe.* 913–916.

[30] Yuxuan Luo et al. 2017. A 93μW 11Mbps Wireless Vital Signs Monitoring Soc With 3-Lead ECG, Bio-Impedance, And Body Temperature. In *Proc. IEEE Asian Solid-State Circuits Conf.* 29–32.

[31] Walter Maetzler, Jochen Klucken, and Malcolm Horne. 2016. A Clinical View on the Development of Technology-Based Tools in Managing Parkinson's Disease. *Movement Disorders* 31, 9 (2016), 1263–1271.

[32] Luis Morillo, Luis Gonzalez-Abril, Juan Ramirez, and Miguel de La Concepcion. 2015. Low Energy Physical Activity Recognition System on Smartphones. *Sensors* 15, 3 (2015), 5163–5196.

[33] Arsalan Mosenia, Susmita Sur-Kolay, Anand Raghunathan, and Niraj K Jha. 2017. Wearable Medical Sensor-Based System Design: A Survey. *IEEE Trans. Multi-Scale Comput. Syst.* 3, 2 (2017), 124–138.

[34] Ben O'Brien, Todd Gisby, and Iain A Anderson. 2014. Stretch Sensors for Human Body Motion. In *Proc. Electroactive Polymer Actuators and Devices*, Vol. 9056. 905618.

[35] Francisco Ordóñez and Daniel Roggen. 2016. Deep Convolutional and LSTM Recurrent Neural Networks for Multimodal Wearable Activity Recognition. *Sensors* 16, 1 (2016), 115.

[36] Anneli Ozanne et al. 2018. Wearables in Epilepsy and Parkinson's disease-A Focus Group Study. *Acta Neurologica Scandinavica* 137, 2 (2018), 188–194.

[37] Jaehyun Park, Hitesh Joshi, Hyung Gyu Lee, Sayfe Kiaei, and Umit Y. Ogras. 2017. Flexible PV-cell Modeling for Energy Harvesting in Wearable IoT Applications. *ACM Trans. Embedd. Comput. Syst.* 16, 5s (2017), 156:1–156:20.

[38] Susanna Pirttikangas, Kaori Fujinami, and Tatsuo Nakajima. 2006. Feature Selection and Activity Recognition From Wearable Sensors. In *Proc. Int. Symp. on Ubiquitious Comput. Systems.* 516–527.

[39] Stephen J Preece et al. 2009. Activity Identification Using Body-Mounted Sensors–A Review of Classification Techniques. *Physiological Measurement* 30, 4 (2009), R1.

[40] Daniel Roggen et al. 2010. Collecting Complex Activity Datasets in Highly Rich Networked Sensor Environments. In *Proc. Int. Conf. on Networked Sensing Syst. (INSS).* 233–240.

[41] Muhammad Shoaib, Stephan Bosch, Ozlem Durmaz Incel, Hans Scholten, and Paul J.M. Havinga. 2015. A Survey of Online Activity Recognition Using Mobile Phones. *Sensors* 15, 1 (2015), 2059–2085.

[42] Texas Instruments Inc. 2016. CC-2650 Microcontroller. [Online] http://www.ti.com/product/CC2650, accessed 31 July 2019.

[43] US Department of Labor. 2017. American Time Use Survey. [Online] https://www.bls.gov/tus/, accessed 13 July 2019.

[44] Adrian Valenzuela. 2008. Energy Harvesting for No-Power Embedded Systems. [Online] http://focus.ti.com/graphics/mcu/ulp/energy_harvesting_embedded_systems_using_msp430.pdf, accessed 31 July 2019.

[45] Nick Van Helleputte et al. 2014. 18.3 A Multi-Parameter Signal-Acquisition Soc For Connected Personal Health Applications. In *Proc. IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. of Techn. Papers*. 314–315.

[46] Alan CW Wong et al. 2008. A 1 V, Micropower System-On-Chip For Vital-Sign Monitoring In Wireless Body Sensor Networks. In *Proc. IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. of Techn. Papers*. 138–602.

Received April 2019; revised June 2019; accepted July 2019