XPro: A Cross-End Processing Architecture for Data Analytics in Wearables

Aosen Wang[1] Lizhong Chen[2] Wenyao Xu[1]
[1] State University of New York at Buffalo, NY, 14260, USA
[2] Oregon State University, OR, 97331, USA
{aosenwan,wenyaoxu}@buffalo.edu,{chenliz}@oregonstate.edu

ABSTRACT

Wearable computing systems have spurred many opportunities to continuously monitor human bodies with sensors worn on or implanted in the body. These emerging platforms have started to revolutionize many fields, including healthcare and wellness applications, particularly when integrated with intelligent analytic capabilities. However, a significant challenge that computer architects are facing is how to embed sophisticated analytic capabilities in wearable computers in an energy-efficient way while not compromising system performance. In this paper, we present XPro, a novel cross-end analytic engine architecture for wearable computing systems. The proposed cross-end architecture is able to realize a generic classification design across wearable sensors and a data aggregator with high energy-efficiency. To facilitate the practical use of XPro, we also develop an Automatic XPro Generator that formally generates XPro instances according to specific design constraints. As a proof of concept, we study the design and implementation of XPro with six different health applications. Evaluation results show that, compared with state-of-the-art methods, XPro can increase the battery life of the sensor node by 1.6-2.4X while at the same time reducing system delay by 15.6-60.8% for wearable computing systems.

CCS CONCEPTS

• **Computer systems organization** → **Architectures**; *Other architectures*; Heterogeneous (hybrid) systems;

KEYWORDS

Cross-End Architecture, Low Power, Data Analytics, Wearable Devices.

ACM Reference format:
Aosen Wang[1] Lizhong Chen[2] Wenyao Xu[1] [1] State University of New York at Buffalo, NY, 14260, USA [2] Oregon State University, OR, 97331, USA . 2017. XPro: A Cross-End Processing Architecture for Data Analytics in Wearables. In *Proceedings of ISCA '17, Toronto, ON, Canada, June 24-28, 2017,* 12 pages.
https://doi.org/10.1145/3079856.3080219

ISCA '17, June 24-28, 2017, Toronto, ON, Canada

ACM ISBN 978-1-4503-4892-8/17/06…$15.00
https://doi.org/10.1145/3079856.3080219

1 INTRODUCTION

Advancement in technology has led to continued miniaturization of sensing, computing and communication devices. This enables wearable systems [31] for a broad range of new applications in a number of important domains, such as wellness and healthcare [4, 50]. Due to their increasing impact on various aspects of our lives, wearable computing systems have become a growing area in the global electronic markets, hitting a market value of $14 billion in 2016 and likely reaching $34 billion by 2020 [22].

Early applications of wearable computing systems mostly focus on health monitoring [39, 49]. As shown in Fig. 1, a body sensor network (BSN)-based wearable computing system is usually comprised of a body sensor (e.g., an ECG pulse wristband) and a data aggregator (e.g., smartphone). The body sensor acquires body signals and wirelessly streams the data to the local aggregator. The data aggregator then uploads the personal body events to cloud servers for health information analytics. Therefore, the early wearable computer systems only focus on the acquisition of health data [56].

Recently, there is an increasing demand to push analytic capabilities to wearable computing systems locally [52]. Performing data analysis locally in the aggregator can enable real-time feedback and timely intervention to chronic longitudinal care. For example, 326,200 people experienced cardiac arrests in the U.S. in 2015 and only 10.6% survived [1]. A wearable heart monitor with an abnormality analytic engine [33], rather than in the cloud and relying on Internet access, can detect cardiac arrests in real-time and significantly increase the chance of rescuing the victims. More broadly, there is a pressing need to support real-time analysis of a variety of vital body signals, such as heart signals, brain signals and muscle signals, right in the BSN system and other wearable computing systems alike.

Conventionally, the widely used approach to achieve real-time analysis of various body signals is the generic classification algorithm, which is originally designed for the mainstream non-wearable biomedical applications (e.g., [6]). Unfortunately, the typical generic classification designs usually employ a computing-intensive structure consisting of a set of feature extractors and classifiers [38]. Early work [36] has indicated that a generic classification implementation can drain a 40mAh battery (which is standard in wearable sensor nodes such as the ECG pulse wristband) in less than 6 hours, which greatly handicaps the longitudinal healthcare services of wearable computing.

Research Problem: In this work, the key problem that we aim to solve is *how to efficiently embed a generic classification scheme into wearable computing systems without compromising sensor battery*

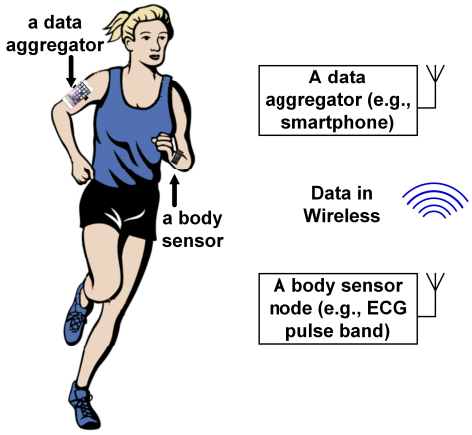


Figure 1: The architecture of a wearable computing system that consists of a body sensor and a data aggregator.

lifetime and system performance. Our goal of optimizing energy-efficiency with time constraint raises a significant challenge for existing efforts on analytic engine designs. To date, analytic engine designs mainly fall into two categories, i.e., in-sensor approach and in-aggregator approach. In-sensor approach implements the analytic module as a whole into the front-end sensor [32]. By performing data analysis in the sensor, this approach reduces the data volume that needs to be transmitted on wireless channels, thus reducing the energy consumption in wireless communication. However, the limited computing capacity and energy budget in the sensor node only empower simple analysis algorithms (e.g., supporting vector machine (SVM) with linear kernel [46]) to be executed in the analytic engine, and running a full-fledged generic classification would drain the sensor battery extremely fast. On the other hand, in-aggregator approach embeds the entire analytic module in the back-end data aggregator [27], such as a smartphone, to leverage the increased computational resource and looser energy constraint in the aggregator. However, this approach requires the sensor node to transmit a large amount of raw data wirelessly to the aggregator, thus still putting a great challenge on the battery life of the sensor node as well.

Cross-end Architecture: As neither the in-sensor front-end approach nor the in-aggregator backend approach can efficiently realize the generic classification, we propose to explore the viability of a *cross-end* architecture that is potentially very promising in addressing the time-constrained energy issue. The concept of cross-end designs have been broadly applied in the system levels (e.g., mobile-cloud cross-end), but the opportunities of applying this concept in the hardware/architecture level, i.e., embedding the generic classification to wearable computing, has largely been unexplored. Specifically, in the cross-end architecture, an analytic engine is divided into fine-grained computing primitives, each of which is mapped to either the front-end or the back-end. By performing an appropriate level of processing in the front-end to reduce the data volume to be sent over wireless channels, the overall energy consumption can be greatly reduced.

Nevertheless, two key challenges must be addressed to enable cross-end architecture. One major challenge is how to design energy-efficient finer-grained computing primitives, as there are substantial function variations among primitives and enormous circuit-level design parameters. The other major challenge is the rapid exploration of architectural design space, i.e., different partitioning and distributing schemes for the computing primitives. First, the partitioning quality must be guaranteed as it decides both the energy consumption and delay in wearable computing systems. Second, the design space of analytic engine partitioning is extremely large, which requires highly efficient exploration methods.

Our Work: In this paper, we present a novel configurable cross-end architecture named XPro, to efficiently implement generic classification in wearable computing systems. The proposed XPro architecture can *formally* partition the generic classification into two parts: one on the wearable sensor node and the other on the aggregator. The two parts collectively implement the full functionality of the generic classification, while minimizing the energy consumption of sensor nodes. Specifically, we devise the generic framework with a set of configurable fine-grained functional cells. We investigate three design rules to improve the energy efficiency of the functional cell implementation in XPro. To facilitate the practical use of XPro, we also design an Automatic XPro Generator. It formulates the problem of partitioning functional cells across two ends with system delay constraint, into a closed-analytic form, and automatically generates the XPro implementation by solving a graph theory problem. In the experiments, we perform an in-depth evaluation on the XPro performance in 6 health monitoring applications, and the results demonstrate the advantage of XPro in improving the energy efficiency of sensor nodes as well as the overall system delay, compared with the state-of-the-art single-end approaches. Furthermore, it can be theoretically proven that XPro has the advantage over single-end approaches, as the in-sensor and in-aggregator implementations are essentially two extreme design cases (i.e., pushing all functional cells into one end in XPro) in the XPro design and implementation space.

Our contribution in this paper is three-fold:

- We propose a configurable cross-end architecture, XPro, that addresses the challenging issue of integrating generic classification engine into wearable systems without compromising system delay.
- We propose three heuristic design rules to optimize energy efficiency from the perspective of the single functional cell. We consider a functional cell as an asynchronous computing unit and optimize cell ALU mode and resource reuse strategy.
- We develop an Automatic XPro Generator for the proposed XPro, which can find the optimal partitioning for the cross-end architecture in polynomial time. We achieve this by formulating the design space searching problem into a standard graph theory problem.

2 BACKGROUND AND XPRO OVERVIEW

XPro is a cross-end architecture that implements generic classification in wearable computing systems. The generic classification is fully realized between a wearable sensor and a data aggregator. In

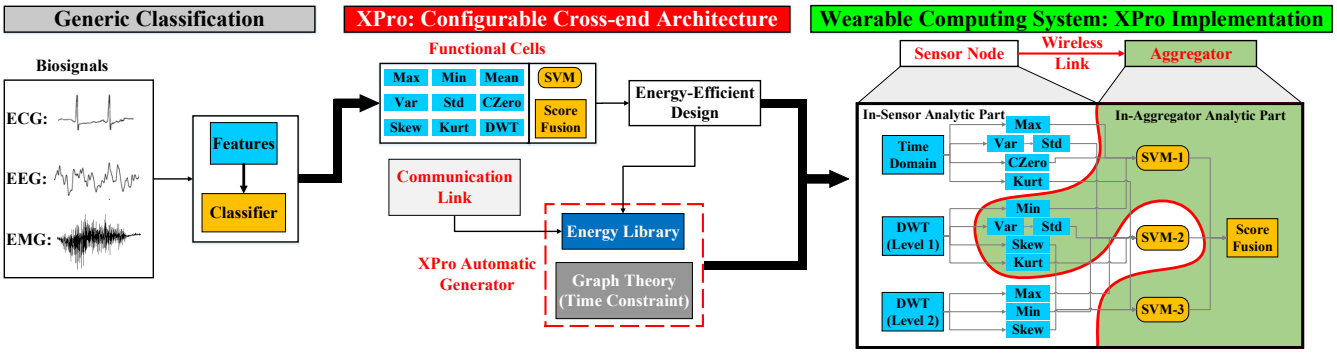


Figure 2: An example of the proposed XPro architecture to embed a generic biosignal classification into a wearable computing system. XPro includes three key components: functional cells, communication link and an automatic generator. In the wearable computing system, the red graph cut line divides the generic framework into the in-sensor and in-aggregator analytic parts.

this section, we first introduce two basic modules in the generic classification, i.e., feature extractors and classifiers (Section 2.1). Then we describe the XPro architecture overview, including the design goal and opportunities (Section 2.2).

2.1 Generic Classification Framework

The generic classification aims to cover diverse types of mainstream vital biosignals, such as Electrocardiography (ECG), Electroencephalography (EEG) and Electromyography (EMG) signals. Different biosignals often have their own descriptive attributes. For example, ECG has salient features in the time-domain [12], EEG is with a good data representation under discrete wavelet transform (DWT) [35], and EMG is more sensitive to the classifier [18]. In this work, we target a generic classification framework with a lightweight statistical *feature set* on the time domain and DWT domain, and a widely used random subspace [21] method as the *classifier*. The chosen feature set is the union of all the preferred features of different biosignals, and the selected classifier can automatically find the favorable features for specific biosignal type in the training phase. Below is more information on this specific feature set and classifier, but the proposed XPro is applicable to other generic classification frameworks as well.

Feature Set: We include 8 hardware-friendly statistical features, including maximal value (*Max*), minimal value (*Min*), mean value (*Mean*), variance (*Var*), standard deviation (*Std*), zeros crossing value (*Czero*), skewness (*Skew*) and kurtosis (*Kurt*). These features can provide a meaningful representation of biosignals and be computed with low hardware cost. Besides time domain, we also extract these statistical features on the multiple levels of discrete wavelet transform (DWT) domain. DWT can provide multi-scale observations for signal analysis and has shown good performance in biosignal applications [20].

Random Subspace: The classifier is based on the random subspace method. It consists of an ensemble of base classifiers. We use supporting vector machine (SVM) as the base classifier for its robust performance. In a generic classification framework, different types of biosignals usually have different preferred feature subsets. Random subspace method can identify their preferences by training base classifiers on random subsets from the complete statistical set. This makes the random subspace method more suitable for generic classification frameworks than other popular ensemble methods, such as bagging [14] and Adaboost [44].

2.2 XPro Architecture Overview

Without loss of generality, Figure 2 depicts an example of a typical XPro architecture to embed the generic classification into a wearable computing system. XPro consists of three major components: functional cells, communication links and an automatic generator. Functional cells are the basic computing units in XPro, which include all the modules that form the generic classification framework. The communication links present the connection among functional cells. The automatic generator is to allocate functional cells and their data links across the sensor end and the aggregator end (Section 3.2). The subset of functional cells that are assigned in the sensor end is referred to as "in-sensor analytic part", whereas the complementary cell subset in the aggregator end is called "in-aggregator analytic part". These three components are explained a bit more below, while the detailed designs are illustrated in the next section.

Functional Cell: To provide flexibility in adjusting workloads and energy consumption between the sensor node and the aggregator, the generic classification framework is divided into fine-grained functional cells. All the functional cells are organized by their execution order in the generic classification (i.e., data-driven execution [34]). Each functional cell can be on either the sensor node or the aggregator. Note that the two special cases of pushing all the cells to either end form the in-sensor approach and the in-aggregator approach mentioned previously. Thus, our proposed cross-end architecture explores a large design space that contains the two existing approaches to improve energy efficiency. Another emphasis on the functional cells is that not all the statistical features are necessarily used in the time domain or different DWT scales. This is because the number of functional cells is decided by the feature set and random subspace training.

Communication Link: The communication link has two types, i.e., intra-end and inter-end. The functional cells in the same end use intra-end communication. In the in-sensor analytic part, intra-end

communication always uses hardware connection, while in the in-aggregator analytic part, it is through memory accessing. The energy consumption of intra-end communication on either the sensor node or the aggregator is usually very small compared with other components under consideration [15]. In the BSN applications, the inter-end communication indicates data exchanging between the sensor node and the aggregator, which is done via near-field communication protocol [5, 29, 30]. The energy consumption of inter-end communication is large and can even be dominant.

Automatic XPro Generator: The proposed automatic generator plays a significant role in the energy efficiency of the sensor node as it determines which functional cells are placed into the in-sensor analytic part. The energy consumption of the sensor node consists of two aspects. One is the computing energy in executing the required computation. The other is the communication energy in transmitting data from a predecessor to a successor along the generic classification topology. Due to the complexity of the partitioning, it is compute-intensive to explore the optimal distribution, particularly under delay constraint. Thus, a more efficient way is much needed.

Opportunities: The ultimate goal of the proposed cross-end architecture is to improve the energy efficiency of the sensor node while not compromising delay and system performance. To achieve that, we maximize the energy efficiency at both the functional cell level and architecture level. On the functional cell level, we optimize the single functional cell implementation based on low-energy design techniques. On the architecture level, we improve energy efficiency by exploring functional cell distribution schemes using the proposed automatic generator. The next section elaborates our designs at these two levels.

3 XPRO DESIGN DETAILS

In this section, we present the design and implementation of XPro. Specifically, we investigate the design of functional cells considering the data flow model, pipeline/parallel units and module reuse (Section 3.1). We also devise an automatic generator for XPro to optimally and efficiently configure XPro according to design constraints (e.g., energy model and delay constraint). The XPro optimization on cross-ends is formulated into a closed form and resolved by a graph-theory problem (Section 3.2).

3.1 Functional Cells in XPro

In this subsection, we discuss the design rules to improve the energy-efficiency of single functional cell implementation. The entire analytic engine in XPro is divided into an in-sensor analytic part and an in-aggregator analytic part. The in-aggregator analytic part is implemented in software, as the aggregator (e.g., a smartphone) typically contains a general-purpose CPU. The in-sensor analytic part is usually based on specialized hardware, such as FPGA or ASIC, to reduce the hardware redundancy (and the associated energy overhead) of general computing platforms. The implementation of the in-sensor analytic part has a large design space. In what follows, we discuss the design space exploration in terms of the primitive computing unit, ALU mode and resource reuse, and present three heuristic design rules.

3.1.1 Asynchronous Computing Unit. Our first design rule is to implement each functional cell as an independent and asynchronous micro-computing unit. The functional cell is the smallest unit prompted by data-driven processing in our XPro architecture. It needs to finish the task independently once its input data is available. Therefore, every functional cell is a micro system with private specialized ALU (S-ALU), cache/buffer and clock. The circuit-level details of our implementation for the functional cell are shown in Figure 3.

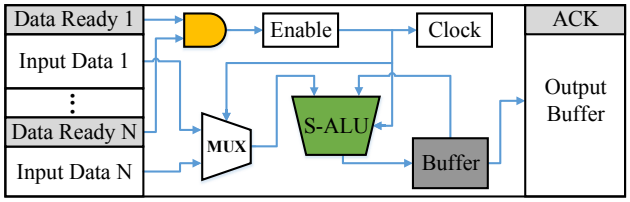


Figure 3: The circuit-level functional cell implementation.

The functional cell has two states, idle and working, which are controlled by the "Enable" module in the figure. The S-ALU module is the core of computation, working with the Buffer module. When the needed input data are not available, the cell is in the idle state. In this state, its input channel passively waits for the data arriving, and all other processing modules are powered off via power gating. When data are ready, the state transitions from idle to working, and all the modules are woken up, including private clock, multiplexer, S-ALU and buffer. Functional cells use asynchronized clocks. This can reduce the static energy consumption of clock tree. The S-ALU comprises both basic arithmetic operations and super computation, which support exponent, square root and reciprocal needed by generic classification algorithms.

3.1.2 ALU Mode in S-ALU Implementation. Our second design rule for the functional cell is to use energy-efficient monotonic working mode for the specialized ALU module. A component in XPro (e.g., *Max*) may consist of multiple functional cells, each of which has an S-ALU module. An S-ALU can typically work in three modes, i.e., serial, parallel and pipeline. Serial processing reduces power consumption, but may compromise timing. The parallel mode incurs more power consumption but increases throughput. The pipeline mode is similar to the parallel mode, but it cannot achieve a scale as large as the parallel. As a result, there is a problem of selecting the mode for each S-ALU in every component of XPro. Because XPro is designed for low-energy wearable computing systems, which monitor and analyze the sparse biosignal events at low sampling rates with typical values of several thousand of hertz, the timing bound is relatively loose. However, it is still complicated to search for the best energy efficiency design if mix-mode is allowed. Considering the time efficiency of exploration, we propose to use a monotonic ALU mode for all the functional cells within one component, but different components may use different ALU mode.

We have performed a preliminary study to investigate the best energy-efficient ALU mode for each component in XPro. Our evaluation criterion for the ALU modes is based on energy/event, where the event indicates a signal segment analysis. Note that the energy

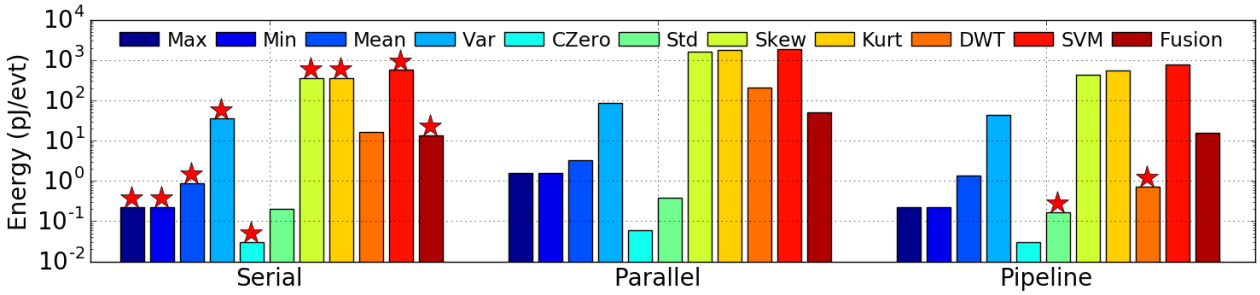


Figure 4: Energy characterization of three ALU modes of the functional cells for each module, with the energy unit as pJ/event. The red star indicates the optimal ALU mode of functional cells for a module.

is the product of power and time, so an ALU-mode with the lowest power does not necessarily lead to the lowest energy. We simulate the function cells under 90nm TSMC library with random input data generation. As plotted in the logarithmic scaled Figure 4, most components are most energy-efficient in serial working mode, as indicated by the red star markers (with some simple operations, such as *Max*, *Min* and *Czero*, being similar to the pipeline mode). However, *std* and *DWT* have the best energy efficiency under the pipeline mode. This is because *std* has only a square root operation and the *DWT* is a matrix multiplication, and in both cases the serial mode has an extremely large delay. We can also observe that the parallel mode of *DWT* has tremendous energy overhead, about two orders of magnitudes larger than the serial mode. This is because the monotonic parallel mode needs a large number of multipliers to compute simultaneously.

3.1.3 Functional Cell Level Reuse. Resource reuse strategy is usually a thorny problem to be addressed in hardware design. Larger-extent resource reuse can reduce implementation duplication and save energy. However, reuse in a larger scale always needs more complex control logic to schedule the resource using. In our individual functional cell design, we apply our third design rule that adopts resource reuse only at the functional cell level, i.e., the reused part must be a single functional cell. A typical example is that the standard deviation feature can reuse the entire variance feature, which is shown in Figure 5. Thus, only a square root operation is needed additionally in *std*. The functional cell level reuse can improve the energy efficiency without complex extra control logic and without imposing additional complexity on generating XPro instances.

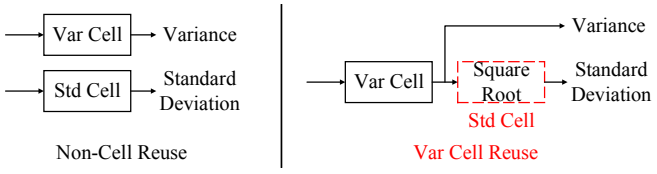


Figure 5: An example to reuse *Var* cell in *Std* Cell.

3.2 Automatic XPro Generator

In this section, we formulate the problem of functional cell distribution scheme between the sensor node and aggregator. Our goal is to minimize the energy consumption of sensor node while not compromising processing delay. Specifically, we first introduce our energy models for design space exploration. We then formulate the cell distribution scheme into a standard graph cut problem without considering the delay. Finally, when delay constraint is added, we transform the distribution problem into a standard max-flow min-cut problem.

3.2.1 Energy Model. The ultimate goal of our cross-end architecture, XPro, is to minimize the energy consumption of the sensor node. The energy consumption in sensor node comprises the energy of analytic computing part, E_a, the energy on wireless communication, E_w, and the energy of sensing, E_s, as the followings:

$$E = E_p + E_w + E_s. \quad (1)$$

However, the energy from biosignal sensing, E_s, can be reduced to an extremely small level [7] compared to the other two components, so our energy model mainly focuses on analytic computing and wireless communication.

We first identify the energy model for the computation of the analytic engine. It can be represented as the multiplication between power consumption and run time, as the following formula shows:

$$E_p = \Sigma_{i=1}^{n} P_i \times t_i, \quad (2)$$

where P_i is the power of the i-th functional cell, t_i is the time delay of the i-th functional cell and n indicates the number of functional cells of the in-sensor analytic part. With finer-grained functional cells and implementation details, we can simulate each functional cell independently with the help of ASIC/FPGA development tools, which can provide us the power consumption information and critical path. By the critical path, we can calculate how many clocks the delay is.

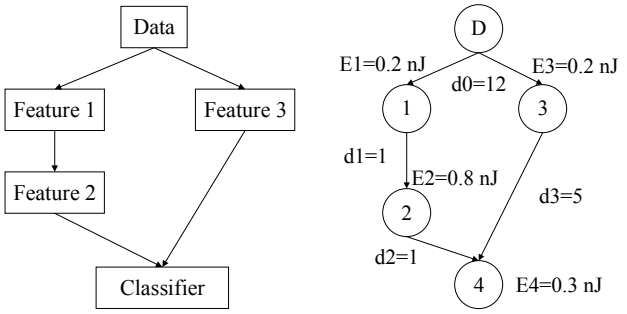
For a wireless transceiver, transmission and reception are two different modes and their energy consumption usually differs. We model this difference into our wireless communication energy consumption as follows:

$$E_w = N_t \times B \times C_t + N_r \times B \times C_r, \quad (3)$$

where N_t is the sample number in transmission task, B is the bit width of each sample and C_t is the average energy model for 1-bit data transmission, which is decided by the specific transceiver design and technology [53]. The N_r and C_r are the corresponding parameters for the data receiving process.

3.2.2 Problem Formulation without Delay Constraint. Our aim in this part is to find a functional distribution scheme between the sensor node and the aggregator, making the energy consumption

of the sensor node minimal. For the simplicity of presentation, we take a basic example of feature-based classification framework to illustrate our formulation. The example is shown in Figure 6.



(a) Computing topology of signal classification. **(b) Functional cell topology graph.**

Figure 6: An example of a generic signal classification and its functional cell topology graph.

Modeling: Figure 6a shows the computing topology with three features and one classifier and Figure 6b is the functional cell topology graph. In this example, we assume that the energy consumption of functional cells corresponding to feature 1, feature 2, feature 3 and classifier are E1=0.2 nJ, E2=0.8 nJ, E3=0.2 nJ and E4=0.3 nJ, respectively. The output dimension of feature 1, feature 2 and feature 3 are d1=1, d2=1 and d3=5 samples. The source data have 12 samples. All samples are 1 bit. The transmission model in the wireless transceiver is C_t=0.1 nJ/bit and reception model is C_r=0.11 nJ/bit.

It can be observed from the topology graph in Figure 6b that feature 1 and feature 3 are both calculated from the original data segment. We refer to them as "grouped". It can be shown that grouped functional cells must be placed in the same end in an energy-minimal distribution. This is because, if they are in different ends (e.g., 1 in the sensor node and 3 in the aggregator), it means that the original data segment has already been sent to the aggregator (e.g., for the processing of 3). In that case, if we re-arrange and move the remaining functional cells in that group (e.g., 1) from the sensor node to the aggregator, it would not increase the wireless transmission energy but would reduce the computing energy in the sensor node, thus resulting in less total energy consumption of the sensor node. Therefore, "grouped" functional cells must stay in the same end.

S-T Graph: We continue to build an s-t graph as shown in Figure 7 based on this functional cell topology graph. We use an "F" node to indicate the front-end sensor node and a "B" node for the back-end aggregator.

In Figure 7, functional cells are all between the "F" node and "B" node. All the functional cells connect to the aggregator with their energy overhead of computation as the weight. Only the functional cells who connect to the original data can build edges to the sensor node. However, a direct connection will not guarantee their "grouped" property, so we propose to add a dummy node "D" to indicate the original source data. The sensor node connects to the "D" node with a weight of the total energy of all samples transmitted to the aggregator. The "D" node connects to the "grouped" functional cell nodes with weights of infinite energy consumption. Since the min-cut will not be on the infinite edges, this technique can properly

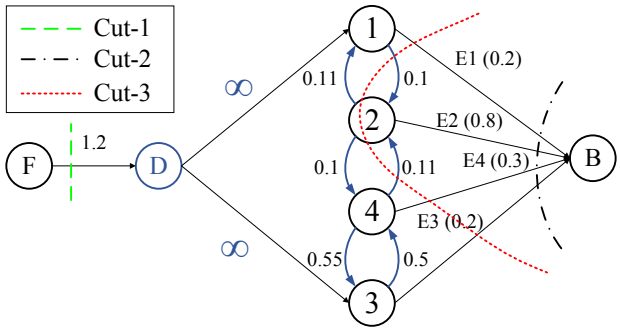


Figure 7: The s-t graph of the generic classification in XPro. Cut-1 is equal to an in-aggregator design; Cut-2 is equal to an in-sensor design; Cut-3 is a general cross-end design.

capture the "grouped" functional cells. According to the topology graph, some functional cell pairs have data dependency relationship. We build two directed edges between such two functional cells to model the energy consumption on wireless communication if they are placed on different ends. The weight of a directed edge from a predecessor to a successor in the topology graph is the transmission energy consumption and the weight of the reverse edge indicates the energy consumption of the reception. In this way, we accomplish the s-t graph building.

Min-Cut Solution: The interesting property of this s-t graph is that if we take a graph cut between the sensor node, "F", and the aggregator node, "B", the cut capacity is equal to the total energy consumption of sensor node under this cut. Therefore, the XPro design without delay constraint can be solved by a standard min-cut solution. After cutting, the functional cells which can reach the "F" node are placed in the in-sensor analytic part and other functional cells connecting to the aggregator are placed in the in-aggregator analytic part.

Examples: Figure 7 depicts three cut examples. Cut-1 disconnects "F" with the source data node, so all the sensing data needs to be transmitted to the aggregator and all the functional cells are in the aggregator. This cut essentially represents the in-aggregator approach. Cut-2 is along the edges between each functional cell and "B". The total energy consumption is the sum of the computation energy from all the functional cells, which essentially forms the in-sensor approach. Cut-1 and Cut-2 are not min-cut. Finally, Cut 3 finds a subset of edges that involve computation energy and wireless communication energy. This is the minimum cut in this example, and thus represents the minimal energy for the sensor node. We can see that the functional cells are distributed across the front-end and back-end in Cut-3. The automatically generated XPro guarantees "not worse" solution than traditional approaches.

3.2.3 Minimizing Energy with Delay Constraint. Delay constraint is another important design consideration in XPro to keep the real-time processing of analytic engines. To take the delay constraint into the formulation, our solution is to add another attribute, delay, for each edge in the previous s-t graph. To this end, we have two attributes in total for each edge, i.e., energy and delay. If we take energy (objective) as the cost and delay (constraint) as the flow in the s-t graph, we can transform the optimal cut problem with

delay constraint into a standard max-flow min-cut problem. Note that, in this case, energy is no longer taken as the flow. It is now the cost of the flow, and the delay acts as the flow. Essentially, the min-cut part minimizes the energy consumption, and the max-flow part guarantees the delay to be under a maximum limit. Also, the updating rule of our flow is to use the maximizing operation, not the accumulating operation anymore.

We use the end-to-end processing time of a signal segment in the wearable computing system from data sensing, to data transmission, to data analysis as the delay. Due to the real-time processing consideration of XPro, we set the delay limit in the max-flow min-cut problem as follows:

$$T_{XPro} = \min\{T_F, T_B\}, \tag{4}$$

where T_F and T_B are the delays for the in-sensor approach and in-aggregator approach, respectively. In this way, the solution found by the XPro generator would have a delay that is at least as good as the better one between the in-sensor and in-aggregator approach. Similarly, we can always guarantee the existence of a solution, i.e., the in-sensor or the in-aggregator in the worse case, although for all the evaluated cases in this work, the Automatic XPro Generator is able to find better functional cell distributions than the two.

4 EVALUATION METHODOLOGY

4.1 Biosignal Dataset

We evaluate the proposed cross-end XPro architecture in the general problem of binary biosignal classification. Binary classification of biosignals is a critical procedure in physiological activity analysis [51]. It not only benefits the automatic abnormal event detection, but also provides basic statistics for high-level biosignals understanding. We select the following testing benchmarks of biosignals. Electrocardiography (ECG), Electroencephalography (EEG) and Electromyography (EMG) are three widely-used representative biosignals [18]. ECG can provide information of the heart state of human beings. It may also provide some underlying information about people's emotions. EEG signal is the activity recordings of our brains and EMG can monitor the muscle behaviors. For a comprehensive demonstration, we choose five datasets of biosignals from UCR Time Series [9], neural spike data [40] and UCI machine learning Repository [43]. The detailed information of test cases from the datasets is listed in Table 1.

Table 1: Attributes of 6 test cases from 5 Biosignal Datasets.

Dataset	Symbol	Segment Length	Segment Number
ECGTwoLead [9]	C1	82	1162
ECGFivedays [9]	C2	136	884
EEGDifficult01 [40]	E1	128	1000
EEGDifficult02 [40]	E2	128	1000
EMGHandLat [43]	M1	132	1200
EMGHandTip [43]	M2	132	1200

Based on these datasets, we extract six testing cases in total, i.e., TwoLeadECG (C1), ECGFivedays (C2), EEGDifficult01 (E1), EEGDifficult02 (E2), EMGHandLat (M1) and EMGHandTip (M2). All the cases are directly from their corresponding datasets. For EEGDifficult, due to different data characteristics, we use the first two categories of data as EEGDifficult01 and the last two categories as EEGDifficult02. The EMGHandLat case tends to use the movement data of lateral and spherical, while EMGHandTip is to identify the movements between tip and hook.

4.2 Wireless Transceiver Models

Energy model in wireless communication is an important factor impacting the implementation of XPro. In this study, we develop a wireless transceiver simulator by adopting the energy model statistics from three ultra-low power wireless transceivers [5, 29, 30], which are specifically designed for medically implanted applications. The three devices are all validated in practical biosignal monitoring applications. The first one [5] is a 350μW FSK/MSK direct modulation transmitter and a 400μW OOK super-regenerative receiver design for medical implants. It can achieve 2.9nJ/bit on transmission and 3.3nJ/bit on reception. The second model [29] incorporates a new current-reuse structure and an inductor-sharing technique to drastically reduce energy consumption. Its energy efficiency is 1.71nJ/bit on the receiver and 1.53nJ/bit on the transmitter at 2Mbps data rate. The last model [30] is an optimized 2Mbps implantable OOK transceiver with 0.42nJ/bit on transmission and 0.295nJ/bit of receiving energy consumption. The simulator employs a common communication protocol and considers an 8-bit header in each payload. Note that we did not include the Bluetooth low energy (BLE) model. While BLE is a popular low-energy design, prior research [47] has shown that it is still orders of magnitude higher than the required μW level sensor hardware design, thus not being suitable for our targeted applications.

4.3 Sensor Node Hardware

We use Synopsys Design Suite [2] to accomplish functional cell designs and exploration. We take TSMC 45nm, 90nm and 130nm standard cell libraries [42] and implement the functional cells in Verilog with Verilog Compile Simulator (VCS). We also carry out the logic simulation of XPro and record the logic toggles in the SAIF file during the simulation of VCS to achieve an accurate power estimation. Power-gating overhead is appropriately accounted for, although we have a similar observation as prior research [19] that the energy and delay overhead from power gating is very limited and does not affect the design and conclusion of the proposed cross-end architecture. Design Compiler (DC) is adopted to synthesize the Verilog design and Power Compiler is used to report the power consumption. Given the typically low duty cycle of biosignals [23], the XPro designs are simulated at a 16MHz clock frequency in this study.

4.4 Approaches under Comparison

We evaluate and compare three design approaches: in-sensor approach (referred to as sensor node engine), in-aggregator approach (referred to as aggregator engine), and the proposed cross-end engine (XPro) under different implementation technologies in biosignal classification tasks. All the six biosignal testing benchmarks are used, i.e., TwoLeadECG (C1), ECGFivedays (C2), EEGDifficult01 (E1), EEGDifficult02 (E2), EMGHandLat (M1) and EMGHandTip (M2). Due to the size of input biosignals, we choose 5-level DWT transformation, whose lengths on different levels are 64, 32, 16, 8 and 4,

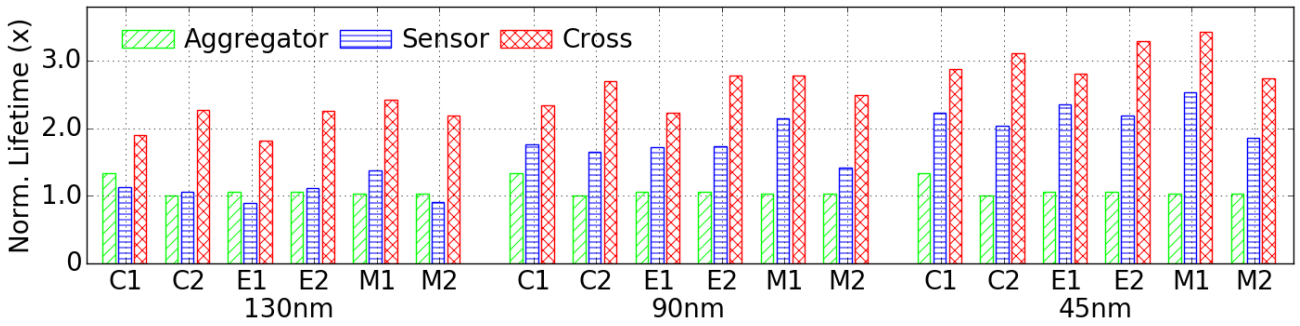


Figure 8: Battery life of the sensor node under 130nm, 90nm and 45nm process technologies with wireless Model 2 for the sensor node engine, aggregator engine and cross-end engine.

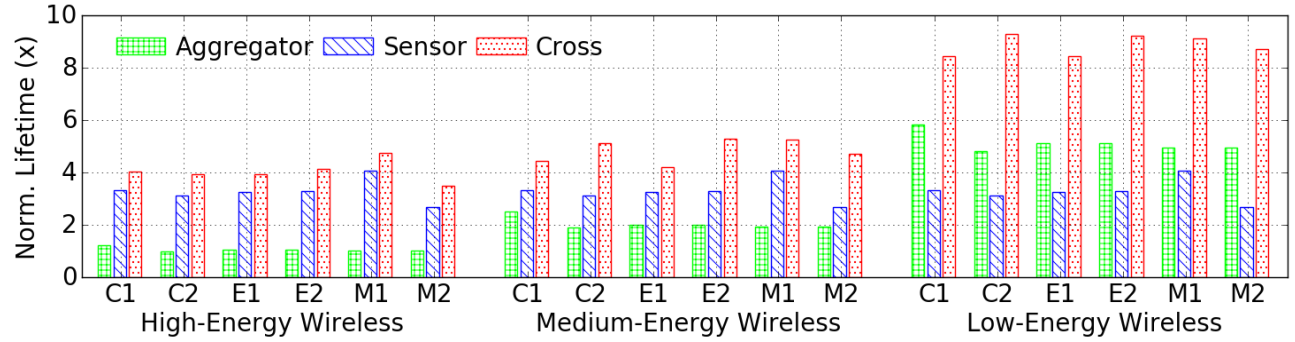


Figure 9: Battery life of the sensor node under three wireless channel models under 90nm process technology for the sensor node engine, aggregator engine and cross-end engine. The three models have the wireless transmission and reception energy of (2.9nJ/bit, 3.3nJ/bit), (1.53nJ/bit, 1.71nJ/bit) and (0.42nJ/bit, 0.295nJ/bit), respectively.

respectively. Note that the 5-th level has two 4-sample segments. All the statistical features are normalized to range [0, 1]. We adopt 32-bit fixed-number with 16-bit integer and 16-bit decimals for functional cells. We choose a binary SVM classifier with radial basis function (RBF) as its kernel. The random subspace takes weighted voting scheme which is trained by the least square method, and 12 features are randomly chosen for each base classifier training. We repeat this procedure 100 times and choose the top 10% on accuracy as the final base classifiers. For each testing case, we randomly choose 75% of the entire data entries as the training set and the other 25% as the testing set. A 10-fold cross-validation technique is applied to train random subspace classifier on each training set. We repeat the training set generation 50 times and choose the classifier with the highest classification accuracy as the final classifier. The delay constraint of the Automatic XPro Generator is set to be the minimal delay value of the sensor node engine and the aggregator engine.

5 EVALUATION RESULTS AND ANALYSIS

5.1 Impact of Process Technology on Battery Lifetime

We first study the impact of process technology on the lifetime of the sensor node in three process setups, including TSMC 130nm, 90nm and 45nm. A typical wireless channel model with 1.53nJ/bit for transmission and 1.71 nJ/bit for reception is used for all the testings here (the impact of wireless models is evaluated in the next subsection). We follow the popular Polymer Li-Ion battery model [8] to estimate the lifetime of sensor node. Figure 8 compares the lifetime results normalized to the aggregator engine approach.

It can be seen that the proposed cross-end architecture not only correctly embeds a complex and energy-hungry analytic engine into the BSN system, but also manages to significantly improve the lifetime of the sensor node by an average of 2.4X and 1.6X compared with the state-of-the-art aggregator engine and the sensor node engine, respectively. As process technology advances, the energy consumption of analytical computation decreases and, consequently, the energy consumption of wireless communication gradually becomes the dominant factor in limiting the lifetime of the sensor node. This trend can be observed in the figure. In the 130nm case, the lifetime of both sensor node engine and aggregator engine is similar, but when it comes to 90nm and 45nm, the lifetime of sensor node engine is much better than the aggregator engine. This is due to the fact that the aggregator engine requires large raw data to be transmitted, thus consuming an increasing percentage of energy in wireless communication. However, in all three process technologies, the proposed XPro has a substantial advantage over the other two approaches, indicating that the cross-end architecture is a promising candidate for years to come.

5.2 Impact of Wireless Models on Battery Lifetime

The wireless model is another major factor that influences the lifetime of the sensor node. To investigate the impact of wireless models, we compare the three wireless designs mentioned in Section 4.2 that

are specifically for medical applications. All the models are low-power but, relatively, Model 1 is "high-energy" with 2.9nJ/bit for transmission and 3.3nJ/bit for the reception. Model 2 is "medium-energy" about twice energy-efficient as Model 1, with 1.53nJ/bit for transmission and 1.71nJ/bit for the reception. Model 3 is ultra-low power and "low-energy", with only 0.42nJ/bit for transmission and 0.295nJ/bit for reception. For the process technology, we select 90nm as the common setup. The results in Figure 9 are normalized to the aggregator engine approach under Model 1.

We can see from the figure that analytic engine implementations are sensitive to wireless models. In wireless Model 1, as the wireless communication takes up the majority of energy consumption, the sensor node engine which performs all the computation in the sensor can greatly reduce the energy on the wireless channel and is much better than the aggregator engine. The proposed cross-end engine can further improve the lifetime by 26.6% on top of the sensor node engine. In wireless Model 2, the energy consumption of computation and wireless communication are evenly matched. The sensor node engine is slightly better than the aggregator engine; whereas the cross-end engine can explore the best configuration to help improve energy efficiency, thereby having the longest battery lifetime. In wireless Model 3, the energy on wireless communication is tremendously reduced due to its ultra-low power wireless model. An intuitively good solution is to transmit all the data back to the aggregator via the cheap wireless channel (i.e., the aggregator engine). This can be confirmed in Figure 9 where the aggregator engine reserves the trend in the previous two wireless models, and now has 74.6% longer lifetime than the sensor node engine. However, by splitting and distributing fine-grained functional cells across the sensor and aggregator, the cross-end architecture is able to beat the intuitively good solution by a large margin, with 73.7% improvement over the aggregator engine and 302% over the sensor node engine, on average. Overall, the proposed cross-end engine successfully embeds the generic classification and achieves the best lifetime across the 3 wireless models and 6 test cases. In the remainder of the evaluation section, unless otherwise stated, we use the "medium-energy" wireless Model 2 and the TSMC 90nm process technology.

5.3 Delay Analysis

The delay in processing an event by the wearable computing system is also a key issue besides energy efficiency. To assess the delay of the aggregator engine (A), sensor node engine (S), and cross-end engine (C), we present the detailed breakdown of the delay. The delay includes all the time it takes to process an event (e.g., a signal segment analysis) which can be spent on the computation in front-end, computation in back-end, and wireless communication. The aggregator engine (A) approach does not have computation in front-end, whereas the sensor node engine (S) approach does not have computation in back-end. The delay breakdown results of the three approaches are plotted in Figure 10.

As can be seen, the delays of all the testing cases of the three engine approaches are less than 4 ms. They can all meet the real-time processing of biosignal streams, considering that the signal sampling rate is typically well below a thousand of hertz. The aggregator engine (A) has the largest delay for all the six test cases, as its delay is dominated by the wireless communication and back-end

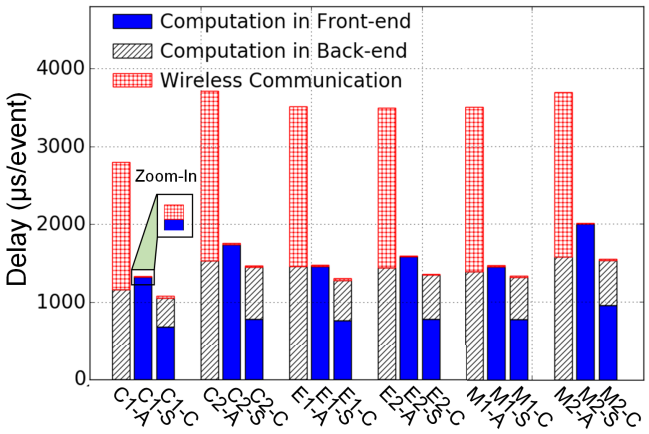


Figure 10: Delay comparison for the aggregator engine (A), sensor node engine (S) and cross-end engine (C).

functional cell processing. It does not assign functional cells in front-end. In contrast, the wireless delay of the sensor node engine (S) is extremely small because it only sends classification result to the back-end. We provide a zoom-in view to highlight this. Finally, the cross-end XPro reduces the delay by 60.8% over the aggregator engine and 15.6% over the sensor node engine on average. The insight behind this improvement is our cross-end approach moves some computations from sensor node to the aggregator. The 15.6% improvement of delay shows that this reduction can well compensate for the increased data transmission delay. This highlights that the proposed XPro is not only able to improve the energy efficiency but at the same time also reduce the delay in processing and analyzing a signal segment.

5.4 Energy Breakdown on Sensor Node

In this subsection, we perform a detailed analysis on the energy consumption of the sensor node and illustrate why XPro saves more energy. Specifically, we examine the contributing factors of the computation (i.e., functional cells) and wireless communication of the sensor node battery lifetime. The energy consumption breakdown is shown in Figure 11, assuming 90nm and wireless Model 2.

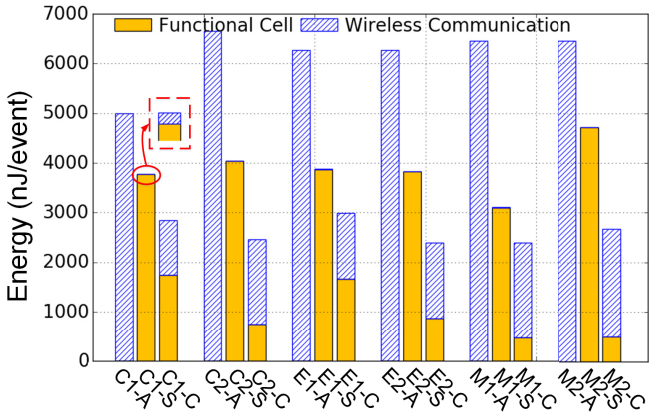


Figure 11: Energy comparison of the aggregator engine (A), sensor node engine (S) and cross-end engine (C).

In this setting, the aggregator engine (A) has the largest energy consumption. It needs to transmit all the sensing data from the sensor node to the aggregator, which is also why its energy consumption on the sensor node is equal to the energy consumption of data transmission. The sensor node engine (S) can reduce an average of 36.6% of the energy consumption compared with the aggregator engine (A). In the breakdown, the energy of the wireless communication of the sensor node engine (S) is hardly visible (shown in the zoom-in), as it only sends classification results back to the aggregator. As for the proposed cross-end engine (C), it achieves the best energy efficiency among the three types of engines in all the six benchmarks. The main reason is that the cross-end engine exploits the large design space of distributing functional cells between the wearable sensor and the aggregator, which the aggregator engine (A) and the sensor node engine (S) cannot take advantage of. Compared with the sensor node engine (S) approach, XPro can reap an additional energy saving of 31.7% (and 56.9% compared with the aggregator engine).

5.5 Effectiveness of Automatic XPro Generator

In this subsection, we take a closer look at the XPro design and investigate the effectiveness of the Automatic XPro Generator in dividing the generic classification and determining the partitioning (i.e., placing a cut). Figure 12 compares the battery lifetime of four cuts. The aggregator engine and the sensor node engine are two extreme cuts where all the functional cells are placed to the back-end and the front-end, respectively. Also shown in the figure is an intuitive way of placing the cut when applying the proposed cross-end approach. This trivial cut is placed between the feature extractors and the classifier, because the features are usually a compact representation of the data and can reduce the energy of wireless communication if put together. The bars labeled as "Cross" correspond to the cut that is obtained with the Automatic XPro Generator.

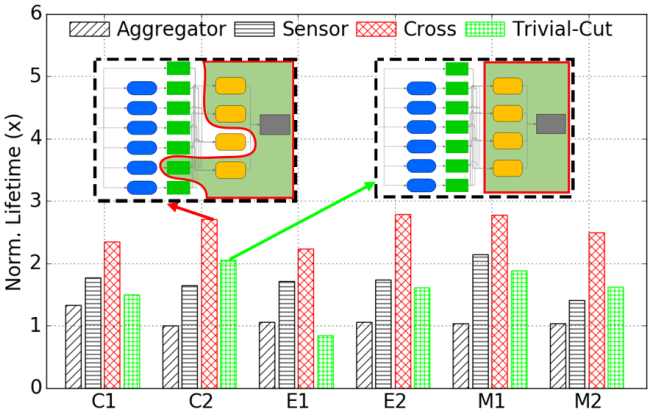


Figure 12: Lifetime comparison for four possible cuts.

First, we can see that by using the cross-end approach directly, even a trivial cut can increase the lifetime in some cases (C2 and M2). However, this improvement is not very consistent, as the lifetime of the trivial cut falls between the aggregator engine and the sensor node engine for C1, E2 and M1, and is worse than both in E1. Second, significant and consistent improvement can be achieved when the Automatic XPro Generator is applied. As shown in the figure, the cut produced by the generator arranges a basic SVM classifier to the sensor node and some light-weight features onto the aggregator to achieve the best energy efficiency. Such cuts are difficult to search through conventional heuristic algorithms, but can be obtained in the proposed generator that cleverly formulates the search into a graph theory problem. Note that some basic SVM classifiers have fewer supporting vectors due to the good data separability of the dataset. When their inputs need some energy-hungry features, such as *Skew* or *Kurt*, placing the classifiers in the sensor node may reduce the energy consumption in computation, with a relatively small energy overhead on the wireless channel. The above results show that the proposed cross-end architecture is a promising approach to improve the lifetime of the sensor node, and the Automatic XPro Generator provides a useful way to obtain a practical XPro design automatically and effectively.

5.6 Energy Overhead of XPro on Aggregator

In wearable computing, the energy impact on the aggregator is much less sensitive compared to the sensor node because 1) the aggregator can be equipped with a large-capacity battery (e.g., 3000mAh in aggregator vs. 40mAh in sensor), and 2) it is very convenient for the aggregator to recharge or replace batteries. To increase the comprehensiveness of our evaluation for XPro, we also quantitatively evaluate the energy overhead on the aggregator. Considering that the sensor node engine has no functional cells in the aggregator, we focus on the comparison between the aggregator engine and cross-end engine here.

We use gem5 [3] to simulate a widely-used mobile CPU, ARM Cortex A8. We also adopt McPAT [25] to collect the power consumption of the functional cells in the aggregator that are implemented with C++ library (implementing the back-end functional cells in software provides the flexibility for the aggregator such as a smartphone to work with different sensor nodes). As shown in Figure 13, the energy overhead of the proposed cross-end architecture is less than half of the aggregator engine. Compared with the aggregator engine, the cross-end engine has potentially fewer functional cells in the aggregator and also allows the aggregator to enter into low-power states when the data are being processed in the sensor node. As an example, if the aggregator has a 2900mAh battery (iPhone 7) with 3.5V voltage, the aggregator can empower XPro for more than 52 hours, and this lifetime can be easily extended beyond 100 hours if operating at a lower sampling and analyzing rate. Given this low impact on the aggregator, the proposed XPro is a highly viable design in practice.

5.7 Discussion

Extension to multi-classification: The binary classification is used as an example throughout the paper. If multi-classification is needed, we can simply add more base classifiers that extend only the topology of generic classification. The rest of the proposed methodology can be applied directly.

Extension to other wireless models: Three representative wireless models are simulated in this work to demonstrate the energy-efficiency of XPro. While more detailed wireless communication models can be used to increase the evaluation accuracy, this work

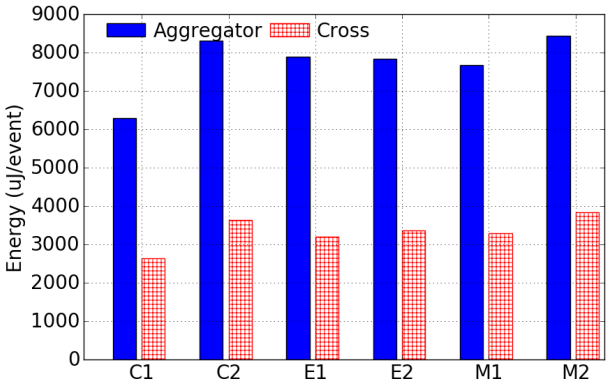


Figure 13: Energy overhead on the aggregator.

presents an important study to assess the general potential of using cross-end implantation for generic classification in BSN systems.

Extension to multiple sensor nodes: A BSN with one sensor node is used in this paper as a case-in-point study to illustrate the essential design principles of XPro. The proposed cross-end approach and the Automatic XPro Generator can also be used with minimal modifications for the case of multiple sensor nodes associated with a data aggregator. MIMO [37] or other specialized wireless protocol can be applied to avoid potential information conflict on the aggregator end.

6 RELATED WORK

Body sensor networks. Miniaturized wearable sensor devices along with a data aggregator (e.g., a smartphone) can form a body sensor network [55]. These devices on different body locations collaboratively provide pervasive health monitoring. Due to the energy constraint on wearable sensors, limited analytic capabilities can be integrated into the sensor node. Therefore, most of the prior work focuses on how to transmit data from the wearable sensors to the data aggregator [10] that has more energy budget and computational resources to implement sophisticated data processing algorithms [41]. However, as discussed in this work, this in-aggregator approach is not sufficient to achieve the needed energy-efficiency of wearable nodes.

Ultra-low power data processing. With the growth of wearable and mobile computers, data processing has been considered to shift from cloud end to user end [45, 54]. Orthogonal to conventional research effort on designing general ultra-low power [16] or even energy-harvesting [24] techniques, a recently popular topic is to devise a specific data processing method, such as a convolutional neural network (CNN), into an energy-efficient implementation [11]. Although significant research thrusts in the computer architecture community have advanced the CNN unit for mobile applications [28], CNN is mainly used for image recognition rather than biosignal processing. There is no in-depth work towards optimizing the power consumption of the generic classification framework for biosignals.

Hardware/Software Partitioning. Hardware/software partitioning is a system approach to implementing an integrated design of hardware and software components in embedded systems [48]. The main foci in software/hardware partitioning is to identify the bottleneck of computing by software profilers [13] and design skilled techniques [17] to optimize the throughput on a hybrid platform (e.g., CPU + FPGA [26]), where the reconfigurable hardware and software computing components are physically connected. However, in wearable computing, wearable sensors and the aggregator are physically located on different body parts and networked through wireless communication. Considering the energy disparity between wearable sensors and the aggregator [7], the optimization criteria and techniques in wearable computing are very different from the integrated hybrid platforms.

7 CONCLUSIONS

Wearable computers, equipped with sensors, are transforming healthcare applications. Integrating analytic engines with wearable computers is in urgent demand. In this work, we investigate a cross-end approach that has largely been unexplored, and present XPro, a cross-end analytic engine architecture for wearable computing. It is able to realize a generic classification and also improve the battery lifetime of wearable sensors without compromising the delay in the wearable computing system. We also developed an Automatic XPro Generator that can efficiently and effectively generate XPro according to the design constraints. Evaluation results demonstrate significant advantages of XPro in both energy-efficiency and processing delay over state-of-the-art approaches. Evaluation results show that XPro can increase the battery life by 2.4X and reduce delay by 60.8% over the in-aggregator approach, and improve the battery life by 1.6X and reduce delay by 15.6% over the in-sensor approach.

REFERENCES

[1] AHA Releases 2015 Heart and Stroke Statistics. http://www.sca-aware.org/sca-news/aha-releases-2015-heart-and-stroke-statistics. (????). Accessed: 2016-08-01.

[2] Himanshu Bhatnagar. 2007. *Advanced ASIC Chip Synthesis: Using Synopsys® Design CompilerTM Physical CompilerTM and PrimeTime®*. Springer Science & Business Media.

[3] Nathan Binkert, Bradford Beckmann, Gabriel Black, Steven K Reinhardt, Ali Saidi, Arkaprava Basu, Joel Hestness, Derek R Hower, Tushar Krishna, Somayeh Sardashti, and others. 2011. The gem5 simulator. *ACM SIGARCH Computer Architecture News* 39, 2 (2011), 1–7.

[4] Jit Biswas, Andrei Tolstikov, Maniyeri Jayachandran, Victor Foo, Aung Aung Phyo Wai, Clifton Phua, Weimin Huang, Louis Shue, Kavitha Gopalakrishnan, Jer-En Lee, and others. 2010. Health and wellness monitoring through wearable and ambient sensors: exemplars from home-based care of elderly with mild dementia. *Annals of telecommunications-annales des télécommunications* 65, 9-10 (2010), 505–521.

[5] Jose L Bohorquez, Anantha P Chandrakasan, and Joel L Dawson. 2009. A 350μW CMOS MSK transmitter and 400μW OOK super-regenerative receiver for medical implant communications. *IEEE Journal of Solid-State Circuits* 44, 4 (2009), 1248–1259.

[6] Sheryl Brahnam. 2010. *Advanced Computational Intelligence Paradigms in Healthcare 5: Intelligent Decision Support Systems*. Vol. 5. Springer Science & Business Media.

[7] Benton H Calhoun, John Lach, John Stankovic, David D Wentzloff, Kamin Whitehouse, Adam T Barth, Jonathan K Brown, Qiang Li, Seunghyun Oh, Nathan E Roberts, and others. 2012. Body sensor networks: A holistic approach from silicon to users. *Proc. IEEE* 100, 1 (2012), 91–106.

[8] Min Chen and Gabriel A Rincon-Mora. 2006. Accurate electrical battery model capable of predicting runtime and IV performance. *IEEE transactions on energy conversion* 21, 2 (2006), 504–511.

[9] Yanping Chen, Eamonn Keogh, Bing Hu, Nurjahan Begum, Anthony Bagnall, Abdullah Mueen, and Gustavo Batista. 2015. The UCR Time Series Classification Archive. (July 2015). www.cs.ucr.edu/~eamonn/time_series_data/.

[10] Yajing Chen, Shengshuo Lu, Hun-Seok Kim, David Blaauw, Ronald G Dreslinski, and Trevor Mudge. 2016. A low power software-defined-radio baseband processor for the Internet of Things. In *2016 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 40–51.

[11] Yunji Chen, Tao Luo, Shaoli Liu, Shijin Zhang, Liqiang He, Jia Wang, Ling Li, Tianshi Chen, Zhiwei Xu, Ninghui Sun, and others. 2014. Dadiannao: A machine-learning supercomputer. In *Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture*. IEEE Computer Society, 609–622.

[12] P De Chazel and RB Reilly. 2000. A comparison of the ECG classification performance of different feature sets. In *Computers in Cardiology 2000*. IEEE, 327–330.

[13] Rolf Ernst, Jörg Henkel, and Thomas Benner. 1993. Hardware-software cosynthesis for microcontrollers. *IEEE Design & Test of computers* 10, 4 (1993), 64–75.

[14] Mikel Galar, Alberto Fernandez, Edurne Barrenechea, Humberto Bustince, and Francisco Herrera. 2012. A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on* 42, 4 (2012), 463–484.

[15] Martijn JR Heck and John E Bowers. 2014. Energy efficient and energy proportional optical interconnects for multi-core processors: Driving the need for on-chip sources. *IEEE Journal of Selected Topics in Quantum Electronics* 20, 4 (2014), 332–343.

[16] Mark Hempstead, Nikhil Tripathi, Patrick Mauro, Gu-Yeon Wei, and David Brooks. 2005. An ultra low power system architecture for sensor network applications. In *ACM SIGARCH Computer Architecture News*, Vol. 33. IEEE Computer Society, 208–219.

[17] Jörg Henkel and Rolf Ernst. 2001. An approach to automated hardware/software partitioning using a flexible granularity that is driven by high-level estimation techniques. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 9, 2 (2001), 273–289.

[18] Ulf Jensen, Matthias Ring, and Bjoern Eskofier. 2012. Generic features for biosignal classification. *Sportinformatik 2012* (2012), 112.

[19] Hailin Jiang, Malgorzata Marek-Sadowska, and Sani R Nassif. 2005. Benefits and costs of power-gating technique. In *2005 International Conference on Computer Design*. IEEE, 559–566.

[20] Parmod Kumar and Devanjali Agnihotri. 2010. Biosignal denoising via wavelet thresholds. *IETE journal of Research* 56, 3 (2010), 132–138.

[21] Ludmila I Kuncheva, Thomas Christy, Iestyn Pierce, and P Mansoor Sa'ad. 2011. Multi-modal biometric emotion recognition using classifier ensembles. In *Modern approaches in applied intelligence*. Springer, 317–326.

[22] P Lamkin. 2016. Wearable Tech Market To Be Worth $34 Billion By 2020. *Forbes* (2016).

[23] Shuenn-Yuh Lee, Chih-Jen Cheng, Cheng-Pin Wang, and Shyh-Chyang Lee. 2009. A 1-V 8-bit 0.95 mW successive approximation ADC for biosignal acquisition systems. In *2009 IEEE International Symposium on Circuits and Systems*. IEEE, 649–652.

[24] Chao Li, Wangyuan Zhang, Chang-Burm Cho, and Tao Li. 2011. SolarCore: Solar energy driven multi-core architecture power management. In *2011 IEEE 17th International Symposium on High Performance Computer Architecture*. IEEE, 205–216.

[25] Sheng Li, Jung Ho Ahn, Richard D Strong, Jay B Brockman, Dean M Tullsen, and Norman P Jouppi. 2009. McPAT: an integrated power, area, and timing modeling framework for multicore and manycore architectures. In *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*. ACM, 469–480.

[26] Yanbing Li, Tim Callahan, Ervan Darnell, Randolph Harr, Uday Kurkure, and Jon Stockwood. 2000. Hardware-software co-design of embedded reconfigurable architectures. In *Proceedings of the 37th Annual Design Automation Conference*. ACM, 507–512.

[27] Wei Liang, Yinlong Zhang, Jindong Tan, and Yang Li. 2014. A novel approach to ECG classification based upon two-layered HMMs in body sensor networks. *Sensors* 14, 4 (2014), 5994–6011.

[28] Robert LiKamWa, Yunhui Hou, Julian Gao, Mia Polansky, and Lin Zhong. 2016. RedEye: Analog ConvNet Image Sensor Architecture for Continuous Mobile Vision. In *ACM/IEEE International Symposium on Computer Architecture (ISCA)*. ACM/IEEE.

[29] Junhua Liu, Chen Li, Long Chen, Yehui Xiao, Jiayi Wang, Huailin Liao, and Ru Huang. 2011. An ultra-low power 400MHz OOK transceiver for medical implanted applications. In *ESSCIRC (ESSCIRC), 2011 Proceedings of the*. IEEE, 175–178.

[30] Li-Chen Liu, Ming-Han Ho, and Chung-Yu Wu. 2011. A medradio-band low-energy-per-bit CMOS OOK transceiver for implantable medical devices. In *Biomedical Circuits and Systems Conference (BioCAS), 2011 IEEE*. IEEE, 153–156.

[31] Paul Lukowicz, Tnde Kirstein, and Gerhard Troster. 2004. Wearable systems for health care applications. *Methods of Information in Medicine-Methodik der Information in der Medizin* 43, 3 (2004), 232–238.

[32] W Marnane, S Faul, C Bleakley, R Conway, E Jones, E Popovici, M de la Guia Solaz, F Morgan, and K Patel. 2010. Energy efficient on-sensor processing in body sensor networks. In *2010 Annual International Conference of the IEEE Engineering in Medicine and Biology*. IEEE, 2025–2029.

[33] Roshan Joy Martis, U Rajendra Acharya, and Lim Choo Min. 2013. ECG beat classification using PCA, LDA, ICA and discrete wavelet transform. *Biomedical Signal Processing and Control* 8, 5 (2013), 437–448.

[34] George Matheou and Paraskevas Evripidou. 2015. Architectural support for data-driven execution. *ACM Transactions on Architecture and Code Optimization (TACO)* 11, 4 (2015), 52.

[35] Hasan Ocak. 2009. Automatic detection of epileptic seizures in EEG using discrete wavelet transform and approximate entropy. *Expert Systems with Applications* 36, 2 (2009), 2027–2036.

[36] Chulsung Park, Pai H Chou, Ying Bai, Robert Matthews, and Andrew Hibbs. 2006. An ultra-wearable, wireless, low power ECG monitoring system. In *2006 IEEE Biomedical Circuits and Systems Conference*. IEEE, 241–244.

[37] Arogyaswami J Paulraj, Dhananjay A Gore, Rohit U Nabar, and Helmut Bolcskei. 2004. An overview of MIMO communications-a key to gigabit wireless. *Proc. IEEE* 92, 2 (2004), 198–218.

[38] Benjamin Pfundt, Marc Reichenbach, Björn Eskofier, and Dietmar Fey. 2013. Smart sensor architectures for embedded biosignal analysis. In *Design and Architectures for Signal and Image Processing (DASIP), 2013 Conference on*. IEEE, 174–181.

[39] Rosalind W Picard and C Du. 2002. Monitoring stress and heart health with a phone and wearable computer. *Motorola Offspring Journal* 1 (2002), 14–22.

[40] R Quian Quiroga, Zoltan Nadasdy, and Yoram Ben-Shaul. 2004. Unsupervised spike detection and sorting with wavelets and superparamagnetic clustering. *Neural computation* 16, 8 (2004), 1661–1687.

[41] Yvan Saeys, Iñaki Inza, and Pedro Larrañaga. 2007. A review of feature selection techniques in bioinformatics. *bioinformatics* 23, 19 (2007), 2507–2517.

[42] Michael Sanie, Michel Côté, Philippe Hurat, and Vinod Malhotra. 2001. Practical application of full-feature alternating phase-shifting technology for a phase-aware standard-cell design flow. In *Design Automation Conference, 2001. Proceedings*. IEEE, 93–96.

[43] Christos Sapsanis, George Georgoulas, and Anthony Tzes. 2013. EMG based classification of basic hand movements based on time-frequency features. In *Control & Automation (MED), 2013 21st Mediterranean Conference on*. IEEE, 716–722.

[44] Robert E Schapire. 2013. Explaining adaboost. In *Empirical inference*. Springer, 37–52.

[45] Ryota Shioya, Masahiro Goshima, and Hideki Ando. 2014. A front-end execution architecture for high energy efficiency. In *Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture*. IEEE Computer Society, 419–431.

[46] Mohammed Shoaib, Kyong Ho Lee, Niraj K Jha, and Naveen Verma. 2014. A 0.6–107 μW Energy-Scalable Processor for Directly Analyzing Compressively-Sensed EEG. *Circuits and Systems I: Regular Papers, IEEE Transactions on* 61, 4 (2014), 1105–1118.

[47] Matti Siekkinen, Markus Hiienkari, Jukka K Nurminen, and Johanna Nieminen. 2012. How low energy is bluetooth low energy? comparative measurements with zigbee/802.15. 4. In *Wireless Communications and Networking Conference Workshops (WCNCW), 2012 IEEE*. IEEE, 232–237.

[48] Greg Stitt, Roman Lysecky, and Frank Vahid. 2003. Dynamic hardware/software partitioning: a first approach. In *Proceedings of the 40th annual Design Automation Conference*. ACM, 250–255.

[49] Mingui Sun, Lora E Burke, Zhi-Hong Mao, Yiran Chen, Hsin-Chen Chen, Yicheng Bai, Yuecheng Li, Chengliu Li, and Wenyan Jia. 2014. eButton: a wearable computer for health monitoring and personal assistance. In *Proceedings of the 51st Annual Design Automation Conference*. ACM, 1–6.

[50] Nagender Kumar Suryadevara and Subhas Chandra Mukhopadhyay. 2012. Wireless sensor network based home monitoring system for wellness determination of elderly. *IEEE Sensors Journal* 12, 6 (2012), 1965–1972.

[51] EM Tamil, HM Radzi, MYI Idris, and AM Tamil. 2008. A Review on Feature Extraction & Classification Techniques for Biosignal Processing (Part II: Electroencephalography). In *4th Kuala Lumpur International Conference on Biomedical Engineering 2008*. Springer, 113–116.

[52] Naveen Verma, Zhuo Wang, and Jintao Zhang. 2015. A look at signal analysis in resource-constrained medical-sensor applications. In *Biomedical Circuits and Systems Conference (BioCAS), 2015 IEEE*. IEEE, 1–4.

[53] Andrew Y Wang and Charles G Sodini. 2004. A simple energy model for wireless microsensor transceivers. In *Global Telecommunications Conference, 2004. GLOBECOM'04. IEEE*, Vol. 5. IEEE, 3205–3209.

[54] Mark Woh, Sangwon Seo, Scott Mahlke, Trevor Mudge, Chaitali Chakrabarti, and Krisztian Flautner. 2009. AnySP: anytime anywhere anyway signal processing. In *ACM SIGARCH Computer Architecture News*, Vol. 37. ACM, 128–139.

[55] Guang-Zhong Yang and Magdi Yacoub. 2006. Body sensor networks. (2006).

[56] Ya-Li Zheng, Xiao-Rong Ding, Carmen Chung Yan Poon, Benny Ping Lai Lo, Heye Zhang, Xiao-Lin Zhou, Guang-Zhong Yang, Ni Zhao, and Yuan-Ting Zhang. 2014. Unobtrusive sensing and wearable devices for health informatics. *IEEE Transactions on Biomedical Engineering* 61, 5 (2014), 1538–1554.