ABSTRACT
Wireless sensor networks have become shared resources providing sensing services to monitor ambient environment. The tasks performed by the sensor nodes and the network structure are becoming more and more complex so that they cannot be handled efficiently by traditional sensor nodes any more. The traditional sensor node architecture, which has software implementation running on a fixed hardware design, is no longer fit to the changing requirements when new applications with complex computation are added to this shared infrastructure due to several reasons. First, the operation behavior changes because of the application requirements and the environmental conditions which makes a fixed architecture not efficient all the time. Second, to collaborate with other already deployed sensor networks and to maintain an efficient network structure, the sensor nodes require flexible communication capabilities. Furthermore, the information required to determine an efficient hardware/software co-design under the system constraints cannot be known a priori. Therefore a platform which can adapt to run-time situations will play an important role in wireless sensor networks. In this paper, we present a hardware/software co-design framework for a wireless sensor platform, which can adaptively change its hardware/software configuration to accelerate complex operations and provides a flexible communication mechanism to deal with complex network structures. We perform real-world measurements on our prototype to analyze its capabilities. In addition, our case studies with prototype implementation and network simulations show the energy savings of the sensor network application by using the proposed design with run-time adaptivity.

Keywords
reconfiguration, sensor networks, hardware accelerator, FPGA, low power, multi-radio

1. INTRODUCTION
A Wireless Sensor Network (WSN) basically consists of a group of energy-limited, multi-functional sensors that are distributed in a geographical area to collect information. A WSN can be used in a large number of various applications, such as agricultural, structural monitoring, and different industrial and domestic applications [23]. Developing and deploying such kind of large scale systems using resource constrained sensor nodes involves considerable efforts. Therefore, to reduce the development and deployment cost, it is a common strategy to share the deployed WSNs as an infrastructure to support multiple applications [3, 8, 14] which can be added into the system at run-time and be assigned to certain nodes. While WSNs are interconnected and supporting an increasing number of applications, they are performing more and more complex tasks. For example, some advanced computations for image/audio processing applications including 2-D convolution, Finite Impulse Response (FIR) filter, Fast Fourier Transform (FFT), etc. are widely used in wireless multimedia sensor networks (WMSNs) [4]. Therefore, to meet the requirements introduced by these new applications, a new sensor platform architecture, which is able to perform complex tasks efficiently and adapt to the change of the applications as well as the environment, is necessary. However, designing a sensor node which is able to adapt to the run-time conditions and provides high performance and energy efficient operations accordingly is a very challenging task. Existing work failed to solve this issue due to the following reasons:

- **Performance constraints:** The traditional sensor nodes, which have fixed hardware configuration [1, 12], suffer from performance issues because their software-implemented features often take very long time to execute the tasks and cannot exploit efficient duty cycling.

- **Energy consumption:** The existing SRAM-based reconfigurable WSN architectures [20, 21] consume considerable energy in sleep mode and during the transition between active and sleep mode which is not acceptable in WSNs.

- **Hardware flexibility:** The architecture with separated microcontroller and ultra-low power FPGA [19] incurs inflexibility and communication bottleneck because the fixed inter-chip communication is not adaptable and has very limited speed.

- **Legacy issues:** Shared WSNs are becoming an essential part of the Internet of Things (IOT) where all the embedded devices can be accessed from the Internet and provide sensing services [17]. The existing WSNs, however, may not be able to collaborate even when they are geographically close to each other if they use
different wireless communication schemes, such as radio frequencies and modulations. Since these networks should operate for years and provide services anywhere pervasively with the help of energy harvesting devices [6], an inefficient design will lead to significant system lifetime reduction.

In this paper, we propose a run-time adaptive WSN platform to address the above issues. Under the resource constraints of the adaptive sensor node, the node can adjust its hardware/software configuration and communication scheme to enhance the performance, improve the energy efficiency, and increase the communication reliability and throughput according to the run-time situation. The key contributions of this work include:

1. We propose HARP, a Highly Adaptive Reconfigurable Platform, with a hardware/software co-design framework to address the key requirements of a run-time adaptive system for WSNs, such as hardware acceleration, low power consumption, run-time reconfiguration and multiple communication schemes.

2. We observe that the run-time adaptivity can provide performance enhancement as well as energy efficiency in WSNs. In fact, depending on the changing requirements of newly added applications, the frequency of event occurrence, or the wireless channel condition, the efficiency of a sensor node can vary significantly.

We employ a run-time adaptive scheme on each node which monitors the run-time situation and changes hardware/software configuration dynamically to adapt to the changing situation.

3. We implement the complete hardware and software stack of HARP and analyze the features of HARP with real-world measurement that provides the guidelines for the further investigation of adaptive sensor nodes.

To evaluate HARP in a real scenario, we choose two application scenarios based on the distributed acoustic analysis [9, 22] as case studies. We evaluate an acoustic event recognition system in our office on our HARP prototype to demonstrate that the run-time configuration management reduces the average current consumption by 31.27% on average. Then we extend the experiment to an acoustic object tracking application with network simulations to demonstrate that the adaptivity on configuration and communication increases the network lifetime and throughput up to 101% and 104%, respectively.

The rest of the paper is structured as follows. In Section 2 an overview of the related work is presented. The problem statement is given in Section 3. In Section 4, an overview of proposed design is described before showing the detail implementation in Section 5. The performance analysis of the platform characteristics and two case studies are presented in Section 6 and 7, respectively. Finally, the paper is concluded in Section 8.

2. RELATED WORK

Traditional wireless sensor nodes, a.k.a. “motes”, are developed to execute simple monitoring tasks. MICAz [1], for instance, is equipped with a 8-bit Atmel ATmega128L microcontroller to monitor the temperature, humidity, etc. These low-end microcontrollers have been proven to be inefficient on the computation intensive tasks such as 2-D convolution and consume 2-6 times higher energy than 32-bit microcontrollers [7]. More advanced platforms are introduced recently to cope with more complex operations using ARM Cortex-M3 microcontroller [12]. However, those motes are not efficient or even cannot meet the timing constraints on some critical operations such as acoustic and image processing. This motivates us to investigate a more efficient platform in this work.

A modular FPGA-based sensor platform, Cookie, has been presented in [20] which includes four layers (processing, communication, power supply and sensing). The processing layer includes a microcontroller and a Spartan-3 FPGA. An enhanced version with Spartan-6, HiReCookie, was proposed in [21] to meet the requirements of complex processing. The SRAM-based FPGAs they used on the platform have high power consumption and standby/wake-up transition energy. Another reconfigurable sensor platform, called HaLOEWEEn, has been introduced in [19]. The platform consists of a System-on-Chip (SoC) and an ultra-low-power IGLOO FPGA. Since the communication interfaces between the microcontroller, the FPGA, and the radio module are not reconfigurable in these platforms, the hardware design needs to be modified to support different applications. These studies focus on performance enhancement on single application and do not discuss the run-time adaptation and its challenges in real world WSNs.

In [13], a multi-radio WSN platform has been proposed. It employs four parallel radio transceivers to increase the data transfer rate and decrease communication latency by sending/receiving data simultaneously. Having multiple independent radios, the node can act as a gateway in order to interconnect different WSNs. Although the sensor node is based on an FPGA, they do not support run-time reconfiguration. Since our proposed sensor platform makes use of microcontroller along with reconfigurable FPGA, it supports run-time reconfiguration. The variety of the communicating mechanism (frequency bands, modulation schemes, etc.) makes our platform flexible and reliable for different network structures and channel conditions.

In addition, none of the aforementioned studies considers the run-time adaptation on both computation and communication mechanisms as a means of achieving high performance and energy efficiency. In particular, there is no existing work which has an overall consideration of an efficient run-time adaptive sensor node based on a practical platform and applications. As we foresee that an adaptive platform can improve the efficiency for the WSNs which will last for decades, we propose a highly adaptive system and investigate its impact.

3. PROBLEM STATEMENT AND DESIGN PRINCIPLES

3.1 Challenges on Efficient Sensor Nodes

In WSNs, the applications are deployed to perform advanced sensing (e.g. image/video, audio, etc.) and information processing tasks (en-/decoding, recognition, etc.) to accomplish the mission. To prolong the network lifetime, the nodes apply a low duty-cycle policy to stay in low power mode when they are idle. To reduce the duty-cycle, hardware accelerators can be exploited to perform certain operations more quickly and efficiently than general purpose processors/microcontrollers. To demonstrate this effect, we perform an experiment with a real hardware implementation.
To deal with these challenges, an adaptive platform should be able to monitor the factors which affect the efficiency of the node at run-time and provide the capabilities to decide a hardware/software combination in order to improve the efficiency.

3.2 Motivational Scenario

Figure 2 depicts a situation of unbalanced event occurrence for random deployed (e.g. airborne) nodes. In this scenario, an acoustic object tracking application [22] is deployed to the existing network which uses acoustic sensors to detect the object types. It is not always possible to load all the accelerators required by the application as it has to compete with other applications. However, due to the spatial correlation of the object occurrence rate and the location, the tasks that should be performed by the nodes are different according to their locations. For instance, the nodes deployed near the road (node S) should perform FFT to determine the object types and send related information to the node (node C) which is away from the object and has capacity to perform the beamforming technique for location extraction. An adaptive sensor node can monitor the utilization and the efficiency of the tasks running on it and determine which accelerator should be loaded on the FPGA and which channel should be used to communicate.

3.3 Design Principles

Motivated by the challenges described in the previous section, we can summarize the design principles of an efficient adaptive hardware/software co-design for WSNs.

- **Hardware accelerations** which can handle complex application within reasonable time and enter sleep mode to preserve energy.
- **Low power consumption** in active and sleep mode is essential to prolong the lifetime using low-duty-cycle policy.
- **Run-time reconfiguration** is required to adapt to the changing requirements subject to the newly deployed applications, role exchanging in network structure, unbalanced event occurrence, unbalanced energy harvesting rate, or other environmental factors.
- **Multiple communication schemes** should be provided to communicate with existing sensor networks operating in different frequency band and to improve the communication efficiency.

To achieve these goals, we propose a run-time adaptive sensor node, HARP, which will be described in details in the following sections.

4. SYSTEM ARCHITECTURE OF HARP

In this section we give an overview of proposed HARP and describe how the design goals can be achieved. Figure 3 shows the HARP architecture which consists of hardware and software subsystems. Application software utilizes the platform through system service and drivers. We begin with the software components followed by the hardware architecture underneath.

4.1 Software Subsystem

The main components of the software subsystem have been...
depicted in Figure 3. They provide the interfaces between the application software and the hardware. The accelerators which can not be loaded on FPGA should perform their software counterpart on the driver (Acc. Driver). In addition, they have to monitor the environment and the node itself to adjust the configuration and operating modes for better efficiency.

### 4.1.1 Hardware Adaptation Layer

Hardware adaptation layer provides applications the accessibility to the hardware including peripherals and accelerators. The accelerator drivers implement the corresponding functions with the software while integrating the hardware accelerators with a wrapper. A set of unified programming interfaces is provided to the application software, so that the operations will be performed by the hardware accelerator automatically when it is loaded on the FPGA. In addition, the drivers have the access to the in-situ power meter to profile the energy usage of the functions. This information will be used by the Configuration Manager to determine an efficient configuration to be loaded on the FPGA.

### 4.1.2 Communication Manager

The condition of the wireless channels changes all the time. Therefore the platform should provide necessary information to the applications in order to select a suitable communication channel. The Communication Manager has the access to the radio drivers to obtain the quality of each link and keeps track of the error rate of the communication. The applications or the network protocols can determine which radio link can be used in a certain condition to improve the reliability and the throughput.

### 4.1.3 Power Manager

The Power Manager is integrated with a scheduler whose mission is to keep track of the tasks to be executed in the queue. When the queue is empty, the Power Manager can switch the node into corresponding low power state. In addition, if an accelerator is not required by the current operation, the Power Manager can apply clock gating mechanism to that accelerator to reduce the power.

### 4.1.4 Configuration Manager

The core component to provide run-time adaptivity is the Configuration Manager. In this section, we analyze the energy consumed on a sensor node under typical workloads of WSNs and explain the design goal of a run-time adaptive system. Typical sense-store-send-sleep applications periodically acquire the samples from the on-board sensors, process them and store the required features, then deliver the results to the gateway for further processing. On the other hand, the event-triggered applications are activated when certain events trigger the on-board sensors or the requests coming from other nodes asking for collaboration. Therefore the number of executions for each function is actually the sum of both types of applications. We formulate the energy consumption as follows:

- Let $T$ denote the time period in which the sensor node provides the service. $T$ consists of two parts, $T_{\text{active}}$ and $T_{\text{sleep}}$, in which the sensor node is in active and sleep state, respectively. In active state the node performs the tasks using the functions implemented on either software or hardware components and consumes various power ($P_{\text{active}}$) accordingly while in sleep state the node turns off all unnecessary components and draws constant power ($P_{\text{sleep}}$).
- Let $F = \{f_1, f_2, \ldots, f_n\}$ denote a set of functions implemented on the node. Each function requires time $t$ and consumes power $p$ to finish it task. Some of the functions, $f'$, can be accelerated by the hardware which requires time $t'$, consumes power $p'$ and occupies area $s'$ on the FPGA under the constraint of $\sum s' \leq S$, where $S$ is the total size of the FPGA.
- Assuming a sequential operation on the microcontroller, the energy consumed on the sensor node in time period $T$ can be denoted by

\[
E_T = \sum_{i=1}^{n} k_i \cdot T_i \cdot P_i + (T - \sum_{i=1}^{n} k_i \cdot T_i) \cdot P_{\text{sleep}} \tag{1}
\]

where $k_i$ is the number of executions of $f_i$ and

\[
T_i \cdot P_i = \begin{cases} f_i \cdot p_i & \text{if } f' \text{ is loaded on FPGA} \\ t_i \cdot p_i & \text{otherwise} \end{cases}
\]

- On a node which can adapt to the environment with run-time reconfiguration, for a time period $T+1$ when a new configuration ($F_{T+1}$) takes place, the energy consumption of the reconfiguration should be amortized by the energy saving of the new set of accelerated functions in order to gain the profit, which means

\[
E_{T+1}(F_{T+1}) + E_{\text{reconf}} + E_0 \leq E_T(F_T) \tag{2}
\]

where $E_{\text{reconf}}$ is a constant energy consumption of the reconfiguration and $E_0$ is a threshold of the minimal energy saving above which we can trade higher efficiency of the next time window with the overhead introduced by the reconfiguration.

As shown in Equation 2, the reconfiguration energy should be amortized by the energy saving of the new configuration to yield a profitable reconfiguration. Configuration Manager should collect the necessary information to infer a profitable configuration in the future. We should note that the event related parameter $k$ in Equation 1 cannot be known a priori. Nonetheless, the event occurrence tends to have spatial and temporal correlations which might be able to be estimated through historical data [5]. We develop a heuristic strategy to determine a set of accelerators to be loaded on the FPGA based on the information collected by the drivers. This standalone strategy does not consider the workload which can
Algorithm 1: Configuration Management

Initialize: Initialize vector Accel\[i\] with $K, q, s$ collected by the drivers and calculate $\varepsilon$ according to equation 3

Initialize: Initialize vector NewConfig\[i\] with 0

Initialize: UsedArea $\leftarrow 0$

Sort Accel\[i\] based on its $\varepsilon$

for $i = 1$ to sizeof(Accel\[i\]) do

if UsedArea + Accel\[i\]s $< \text{MAX-FPGA-SIZE}$ then

NewConfig\[i\] $\leftarrow 1$

UsedArea $\leftarrow$ UsedArea + Accel\[i\]s;

else

break;

end

OldGain $\leftarrow$ calculateEnergyGain(OldConfig\[i\], Accel\[i\])

NewGain $\leftarrow$ calculateEnergyGain(NewConfig\[i\], Accel\[i\])

if NewGain $> \text{OldGain} + \text{CONF-ENERGY} + \text{MIN-GAIN}$ then

OldConfig\[i\] $\leftarrow$ NewConfig\[i\]

perform reconfiguration;

end

be mapped and scheduled among different nodes. The application assignment and configuration management involving multiple nodes is more complex and deserves further investigations [11].

The service time of a sensor node is divided into equally spanned time windows, $T_w$. At the beginning of each $T_w$, the Configuration Manager determines whether to perform the reconfiguration with a new set of accelerators. As shown in Equation 1, the energy gain, $g_i$, of an accelerator $i$ for each operation is $t_i \cdot p_i - t_i' \cdot p_i'$. Therefore the total energy gain for this time window, $G_w$, can be calculated by $k_i \cdot g_i$. To smooth the historical event occurrence out, we use exponential moving average (EMA)$^1$ to calculate $K_{Tw} = \alpha \times K_{Tw-1} + (1 - \alpha) \times K_{T_0}$, $\alpha \in \mathbb{R}[0 \leq \alpha \leq 1]$ where $\alpha > 0.5$ is chosen in favor of the recent observation. The size of the FPGA constrains the number of the accelerators to be included in the configuration, therefore we define the efficiency function

$$\varepsilon_i = \frac{K_i \times g_i}{s_i}$$

(3)

where $s_i$ is the size of the accelerator $i$. The goal of the Configuration Manager is to maximize the energy gain under the size constraint. If the Configuration Manager finds a new configuration which is more efficient than the current one and can amortize the reconfiguration energy, the reconfiguration will take place in the next time window. We highlight the strategy in Algorithm 1. Due to the sorting operation, the time complexity of this approximation is $O(\alpha \log n)$ with a worst-case performance ratio equal to $\frac{1}{2}$, which is suitable for the sensor nodes with limited resources like the energy and the memory.

4.2 Hardware Subsystem

To provide efficient run-time adaptation, the hardware subsystem is built around a coprocessor-based computation unit. Figure 4 shows the hardware architecture of HARP, which is described in the following sections.

1 http://www.itl.nist.gov/div898/handbook/pmc/section3/pmc324.htm

Figure 4: Hardware architecture of HARP sensor platform

4.2.1 Reconfigurable Computation Unit

The computation unit consists of a microcontroller and an FPGA. The microcontroller accommodates the software subsystem and provides access to other hardware components. Hardware accelerators are implemented on the FPGA, each of which has an input buffer and an output buffer. The buffers are accessible from microcontroller via a high-speed bus. The software code on microcontroller feeds the input data to the input buffer of hardware accelerator and receives the output results from the output buffer. Hardware accelerators can run parallel to each other. New hardware accelerators can be added to the FPGA after deployment by means of run-time reconfiguration.

The FPGA is also in charge of the radio interface management. Radio modules are connected to the SPI cores provided by the FPGA. The controller sends/receives the data to/from the SPIs through the bus. Implementing the controller on the FPGA provides this possibility to exploit both the radio modules at the same time. It is also possible to preprocess the data received from the radio modules on the FPGA simultaneously, so that the microcontroller can be relieved from handling masses of data for some applications. The interconnections between major components such as the FPGA, the microcontroller, the interfaces and the peripherals are reconfigurable. Online reconfiguration provides this possibility to change the architecture based on new conditions, if needed.

The system controller provides the interface for the Configuration Manager and the Power Manager to perform run-time reconfiguration and control the low-power mode of HARP, so that the adaptation can be achieved with energy efficiency.
4.2.2 Radio Control

Due to the flexible interconnection provided by the computation unit, it is possible to support multiple radio transceivers. The transceivers can be controlled either by the microcontroller or a customized controller implemented on the FPGA. This also opens the possibility to process the received data on-the-fly in parallel to the microcontroller.

4.2.3 Power Profiler

Power profiling is a key-component for the run-time adaptation. Since the Configuration Manager requires the information about the efficiency of the operations in terms of the energy consumption, the in-situ power meter samples the current consumption of the node and provides the necessary information.

4.2.4 Sensing Unit

Sensors are connected to the ADCs which is controlled by the computation unit. Depending on the application requirements, the samples can be acquired on demand, stored on the memory and analyzed either by the microcontroller or the FPGA.

5. IMPLEMENTATION

In this section, we introduce the hardware and software implementation of HARP in details. Figure 5 shows the complete HARP platform.

5.1 Reconfigurable System-on-Chip (SoC)

Microsemi offers SmartFusion2 which is a low-power, flash-based customizable SoC [18]. This SoC solution integrates an FPGA, a full-feature microcontroller subsystem (MSS) and several analog functions on the same chip. SmartFusion2 has very low static power by providing low power modes for the FPGA, the microcontroller and the peripherals which make it suitable for low duty cycle applications. SmartFusion2 devices use Flash*Freeze (FF) technology which provides an ultra-low power (~1mW) static mode for FPGA fabric. In FF mode, all the SRAM and register information are retained. The FPGA fabric can be dynamically reprogrammed using In-Application Programming (IAP). The embedded Cortex-M3 processor obtains new programming bitstream file and passes it to the system controller. Then, the system controller runs the IAP service to reconfigure the FPGA.

The MSS contains a hard 32-Bit Cortex-M3 processor, running at up to 166 MHz. The Cortex-M3 is in charge of controlling peripherals, sensor modules, hardware accelerators and performing dynamic reconfiguration.

5.2 Radio Modules

The wireless communication unit is composed of two low power radio modules, one for short distance transmission and the other for communicating over longer ranges. Two radio transceivers transmit data on two different, noninterfering channels simultaneously. We choose CC2420 and CC1100 radio modules which are widely used in WSN platforms such as Telos, MICAz, MICA, etc. They operate on different frequency bands (2.4 GHz and Sub-1GHz). Hence, they have different transmission range and fading characteristics.

The radio transceivers are connected to a SPI interface to communicate and exchange data with the FPGA. A hardware module is implemented on FPGA to get the packets from radio modules and send the data to the microcontroller, and vice versa.

5.3 Power Meter

The power meter is composed of an adjustable series of shunt resistors, a differential amplifier and a low power 12-bit ADC, Texas Instruments (TI) ADC121C021. Since the current consumption of sensor platform varies significantly over time and states (sleep mode, active mode, idle mode), the power meter should be able to capture this wide dynamic range. The microcontroller can control this range dynamically by adjusting the resistance of the circuit.

5.4 Sensor Modules

To connect to the most of the commercial sensor modules, we adopt the widely used 51-pin sensor board connector. Since the interconnection is reconfigurable, it provides more flexibility than traditional motes. In this work, we integrate a sensor board with an acoustic sensor, a motion sensor and a magnetic sensor connecting to a low power 4-channel 16-bit ADC, TI AD9834J1, for high resolution sensing requirements.

5.5 Software System

We integrate TinyOS [2] to provide the system services of HARP. One advantage of TinyOS is that there are numerous applications and network protocols developed under it. Our software architecture allows the application developers integrating they TinyOS applications without knowing the hardware details. The run-time adaptation occurs spontaneously under the control of Configuration Manager. We implement the hardware drivers and IAP to provide the adaptivity based on the decision of Configuration Manager. Cortex-M3 processor requests and receives the configuration through the wireless transceivers and stores it on an external flash memory for run-time reconfiguration.

6. EVALUATION

To evaluate the performance and the efficiency of HARP, we conducted a series of experiments by profiling the operations performed by HARP with different configurations. We use Agilent N6705B DC Power Analyzer to supply the power to the prototype and record the current consumption of each operation. To measure the execution time of each op-
Table 1: Power consumption of SAM3U and Smartfusion2

<table>
<thead>
<tr>
<th>Frequency</th>
<th><a href="mailto:SAM3U@1.8V">SAM3U@1.8V</a> (mW)</th>
<th><a href="mailto:Smartfusion2@1.2V">Smartfusion2@1.2V</a> (mW)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4MHz</td>
<td>8.964</td>
<td>1.782</td>
</tr>
<tr>
<td>48MHz</td>
<td>54.342</td>
<td>21.096</td>
</tr>
<tr>
<td>96MHz</td>
<td>86.4</td>
<td>39.592</td>
</tr>
<tr>
<td>144MHz</td>
<td>-</td>
<td>78.3</td>
</tr>
</tbody>
</table>

6.1 Accelerations

The applications running on HARP can perform the task using either software or hardware implementation. The software implementation consumes less current than its hardware counterpart but incurs longer execution time. We implement several operations which are widely used in sensor network applications such as FFT for acoustic event recognition, Advanced Encryption Standard (AES) for secure communication [15], Reed-Solomon (RS) coding for reliable communication, and FIR for noise filtering. Table 2 shows the current consumption, the execution time, and the energy consumed. We can see that the energy saving from the hardware accelerations varies from several to $10^3$ folds. In a real deployment, the access rate of these operations depends on the applications running on the node and the event occurrence rate. This emphasizes the importance of the run-time profiling and reconfiguration for an efficient system.

6.2 Low Power Mode

HARP provides several levels of low power mode. The platform can request the microcontroller and the FPGA to enter low power mode individually. Table 3 shows the current consumption of different low power modes. It is worth noting that it takes time to reach the stable current consumption after a certain component enters certain low power mode. Figure 7 shows the current consumption of HARP in different low power modes.

Figure 7: Entering low power mode

6.3 Run-time Reconfiguration

The duration of the reconfiguration depends on the size of the configuration data (bitstream) which is proportional to the size of the FPGA. On our prototype we use M2S010 which has 12084 reconfigurable tiles resulting 611 KB of configuration data. Table 4 shows the duration and energy consumption of the reconfiguration. We can see that the reconfiguration is an expensive operation in terms of energy consumption. This cost should be considered when assigning a new application to a sensor node or determining whether to change the configuration.

Even though the energy needed for the run-time reconfigu-
Current (mA)
Energy (J)
0.051532
0.113689
24.071
21.689
0.002055
0.002939
19.699
0.002114
-22.555
5.149
5.199
12.134
0.003064
41.111
15.556
11.776

should decide an efficient configuration manager because there is no additional overhead to configure the FPGA while switching to active mode like on a SRAM-based device. For instance, a SPARTAN-6 used in [21] consumes 50mA at 1.2V for configuration each time when the FPGA wakes up. Utilizing the partial reconfiguration feature, it takes 0.5 second to load the configuration. It means that for an application which wakes up every 5 seconds, 518.4 Joule will be consumed in one day solely for loading the configuration at wake-up.

6.4 Communication
We characterize the radio modules integrated in HARP by measuring the duration and the current consumption at each stage of communication operations. The operation can be mainly divided into two parts, namely microcontroller related and transceiver related. Microcontroller should transfer data between RAM and transceiver’s FIFO and then the transceiver should transfer/receive the data in FIFO to/from the antenna. Table 5 shows the characteristics of both CC2420 and CC1100. The data size is 64 bytes for both transceivers. CC2420 performs standard IEEE802.15.4 operations due to its fixed configuration, while CC1100 is configured to run at 433 MHz with MSK (Minimum-shift keying) modulation. As we can see in the table, both transceivers share similar characteristics in terms of the current consumption and the duration. We observe that the time needed for the communication between the microcontroller and the transceiver dominates the total duration required. Depending on the complexity of the protocol, the microcontroller should spend corresponding time to process the data. This fact justifies the choice of connecting multiple transceivers using dedicated interfaces connected to the FPGA because this enables the possibility of acceleration and parallelism.

### Table 5: Communication Components

<table>
<thead>
<tr>
<th>chip</th>
<th>Operation</th>
<th>Current (mA)</th>
<th>Duration (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CC1100</td>
<td>Idle</td>
<td>4.216</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Receive</td>
<td>22.666</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>TX MSS + Transceiver</td>
<td>5.199</td>
<td>0.041657</td>
</tr>
<tr>
<td></td>
<td>TX Transceiver (0dBm)</td>
<td>12.134</td>
<td>0.03064</td>
</tr>
<tr>
<td></td>
<td>RX MSS + Transceiver</td>
<td>5.149</td>
<td>0.034611</td>
</tr>
<tr>
<td></td>
<td>RX Transceiver</td>
<td>19.699</td>
<td>0.02114</td>
</tr>
<tr>
<td>CC2420</td>
<td>Idle</td>
<td>4.334</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Receive</td>
<td>24.071</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>TX MSS + Transceiver</td>
<td>4.829</td>
<td>0.021432</td>
</tr>
<tr>
<td></td>
<td>TX Transceiver (0dBm)</td>
<td>15.033</td>
<td>0.002939</td>
</tr>
<tr>
<td></td>
<td>RX MSS + Transceiver</td>
<td>22.555</td>
<td>0.136089</td>
</tr>
<tr>
<td></td>
<td>RX Transceiver</td>
<td>21.689</td>
<td>0.002055</td>
</tr>
</tbody>
</table>

6.5 Power Profiling
To evaluate the built-in power meter, we output the current readings to the UART port and record them into a file with markers of specific events. The built-in power meter is calibrated using Agilent N6705B DC Power Analyzer and the timing of each reading is measured using the oscilloscope. Due to the overhead of UART output, the minimum interval of each reading is 1.78 ms which is around 562 samples per second. With the output turned off, the built-in power meter can achieve more than 8 kops. We compare the current consumption measured using our power meter with the measurements on the DC Power Analyzer and plot the results on Figure 8. As we can see, although the built-in power meter has lower resolution than the DC Power Analyzer, it can correctly capture the current consumption of the operations at real-time. By comparing the constant current draw on the DC Power Analyzer, the output from the built-in power meter has less than 1% error. This high accuracy is important for the power profiling to manage the applications and configurations efficiently.

7. CASE STUDY
In this section we use two applications to demonstrate the advantages of HARP based on the prototype implementation and simulation. An experiment based on real world sensing data is performed to investigate the behavior of HARP in a real world scenario.

7.1 Indoor Monitoring System
In this experiment, an indoor monitoring system in an office environment with 7 sensor nodes is deployed as depicted in Figure 9. The sensor nodes detect the moving objects using Passive Infrared (PIR) sensors and record the sound samples to recognize the event. The log data must be transmitted through a secure and reliable channel periodically. Therefore, 4 kinds of accelerators can be used on the sensor nodes which are FFT, FIR, RS and AES for sound classification and filtering, channel coding and encryption, respectively. The **Configuration Manager** should decide an efficient configuration based on the run-time observation. FIR and FFT accelerators are installed by default since they have higher speed-up ratio. We record the event trace in a duration of 4 days and use it as the test data input. Then we record the energy consumption of the node using Agilent N6705B DC Power Analyzer. Figure 10 shows the PIR event distribution at each location during the office hour. As we can see in the figure, the nodes can be divided into three groups. Node 1 and 6 are not often triggered by the PIR events, while node 2, 4 and 5 are PIR event dominated. Node 3 and 7 are somewhere in between. In our implementation, we record 8k audio samples when the PIR sensor is triggered, then use FIR to filter the noise and FFT to classify the sound. On the other hand, the node should transmit data encrypted by AES and encoded with RS to increase the reliability.
The average current consumption of selected nodes is shown in Figure 11. We compare the node with only software implementation, the fixed hardware configuration, and the run-time adaptation. We can see that even though the PIR events occur more frequently on node 4, the default configuration does not yield the best efficiency. The Configuration Manager decides to load the configuration based on FIR and AES during the experiment and save 7.77% and 22.07% of current consumption compared with the default configuration and the one without acceleration, respectively. However, since the PIR event rarely occurs on node 1, the Configuration Manager chooses AES and RS based configuration and has 10.22% and 11.42% savings compared to the default configuration and software implementation, respectively. To see the effect of frequent PIR events, we manually trigger the PIR operations every 10 seconds. The results of this imaginary node (Node-X) show that default configuration is actually the best choice in this case and yields 63.05% of current saving compared to the software implementation.

7.2 Acoustic Object Tracking Application

To investigate all the features of HARP at network level, we use network simulator to simulate an acoustic object tracking application. In this application, Network clustering technique [10] is implemented, which organizes the network into clusters, each has a cluster head (CH) and cluster members (CM). CMs record audio samples using their microphones and recognize the object type. They forward the result to CHs, which track the object using beamforming algorithms [22] to locate its position. In order to balance the workload, every certain period of time (called "round"), clusters are reformed and different nodes are elected to become CHs. In our case study, we deploy 96 nodes randomly distributed in a 250m × 250m area using NS2 network simulator. Each node has initial energy of 18720 Joules which is equivalent to the energy of two AA batteries. A member node performs the sensing task in every 10-second frame as depicted in Figure 12. The task of a member consists of audio recording, audio clip analyzing (FFT based), encoding the data with RS error correcting code [16], sending the data to the CH via CC2420 transceiver, then entering sleep mode. A CH collects packets from its members, decodes the data, performs the beamforming algorithm (FIR based) to estimate the location, then sends it to the base station via CC1100 transceiver. When clusters are formed, the reconfiguration takes place to accelerate the cluster head. We compare HARP in terms of energy consumption and the network lifetime with three other cases: 1) no acceleration available. 2) fixed hardware accelerators. 3) SRAM-based FPGA with reconfiguration.

Figure 13 shows the residual energy of the whole network (96 × 18720 Joules) during the simulation. We can see that the case without hardware accelerations is very inefficient and drain out the energy very quickly. Comparing to the fixed hardware implementation, the run-time reconfiguration shows the benefit when round time increases because the longer the round time is, the less reconfiguration overhead is introduced. When the round time is set to 1, 2, and 3 hours, HARP obtains 5%, 9% and 12% energy saving, respectively. Comparing with the SRAM-based FPGA, the energy savings are 0.8%, 4.1% and 7.5%, respectively.

In Figure. 14 we compare the network lifetime with different round time. Comparing with the case without hardware accelerations, HARP with 1, 2, and 3 hours round time increases the lifetime by 96%, 99% and 101%, respectively. Comparing with fixed hardware implementation, the lifetime is extended by 5.6%, 7.7% and 9.5%, respectively. Com-
paring with the SRAM-based FPGA the increase is 1.1%, 3.2% and 5.3%, respectively. In addition, we measure the throughput of the system with and without multiple radio transceivers. When the node utilizes both radios for data transmission, it can reach 335.69 kbps while using a single transceiver, the throughput is only 164.53 kbps.

8. CONCLUSION

In this paper we presented HARP – a hardware/software co-design framework for a reconfigurable wireless sensor platform which can adapt to the dynamic application requirements and the run-time situations in wireless sensor networks. HARP provides high performance and efficiency for applications by using hardware accelerators. It employs multiple radio transceivers with dedicated communication interfaces to achieve flexible communication and high throughput. HARP is capable of run-time reconfiguration which can remotely request and change the hardware/software combination to perform new tasks or improve the efficiency of applications. We provided a detailed description and a comprehensive analysis of HARP in terms of energy consumption and execution time with real-world measurements. The case studies show that the current consumption drops 31.27% on average and the network lifetime increases up to 101%. In the future, this framework can be integrated with energy harvesting devices and software-defined radios to further improve the energy efficiency and communication flexibility.

Acknowledgment

This work was supported by German Research Foundation (DFG), Research Training Group GRK 1194 (Self-organizing Sensor-Actuator-Networks).

References