Chimera: A Low-power Reconfigurable Platform for Internet of Things

Emekcan Aras, Stéphane Delbruel, Fan Yang, Wouter Joosen, Danny Hughes

Chimera: A low-power reconfigurable platform for Internet of Things

EMEKCAN ARAS, STÉPHANE DELBRUEL, FAN YANG, WOUTER JOOSEN, and DANNY HUGHES, KU Leuven, Belgium

The Internet of Things (IoT) is being deployed in an ever-growing range of applications, from industrial monitoring, over smart buildings to wearable devices. Each of these applications has specific computational requirements arising from their networking, system security, and edge analytics functionality. This diversity in requirements motivates the need for adaptable end-devices, which can be re-configured and re-used throughout their life-time to handle computation-intensive tasks without sacrificing battery life-time. To tackle this problem, this paper presents Chimera, a low-power platform for research and experimentation with reconfigurable hardware for the IoT end-devices. Chimera achieves flexibility and re-usability through an architecture based on a Flash Field Programmable Gate Array (Flash FPGA) with a reconfigurable software stack that enables over-the-air hardware and software evolution at run-time. This adaptability enables low-cost hardware/software upgrades on the end-devices and an increased ability to handle computationally-intensive tasks. This paper describes the design of the Chimera hardware platform and software stack, evaluates it through three application scenarios, and reviews the factors which have thus far prevented FPGAs from being utilized in IoT end-devices.

CCS Concepts: • **Computer systems organization → Embedded hardware**; *Sensor networks*; *Embedded software*.

Additional Key Words and Phrases: Reconfigurable Sensor Node, Sensor Network, FPGA

ACM Reference Format:
Emekcan Aras, Stéphane Delbruel, Fan Yang, Wouter Joosen, and Danny Hughes. 2020. Chimera: A low-power reconfigurable platform for Internet of Things. 1, 1 (December 2020), 25 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

The Internet of Things (IoT) has become an important element of the information technology landscape over the past decade. Contemporary IoT applications cover a large and growing application space, from smart homes to safety-critical industrial systems. The IoT is becoming a critical part of daily life, including smart electricity grids [57], smart cities [58], and fleets of smart connected vehicles [58] and health-care management [13]. These applications are highly diverse in terms of networking, system security, and edge analytics requirements. This raises new architectural challenges that have encouraged the development of a rich diversity of IoT platforms that enable

Authors' address: Emekcan Aras, emekcan.aras@cs.kuleuven.be; Stéphane Delbruel, stephane.debruel@cs.kuleuven.be; Fan Yang, fan.yang@cs.kuleuven.be; Wouter Joosen, wouter.joosen@cs.kuleuven.be; Danny Hughes, danny.hughes@cs.kuleuven.be, KU Leuven , Leuven , Belgium.

XXXX-XXXX/2020/12-ART $15.00
https://doi.org/10.1145/1122445.1122456

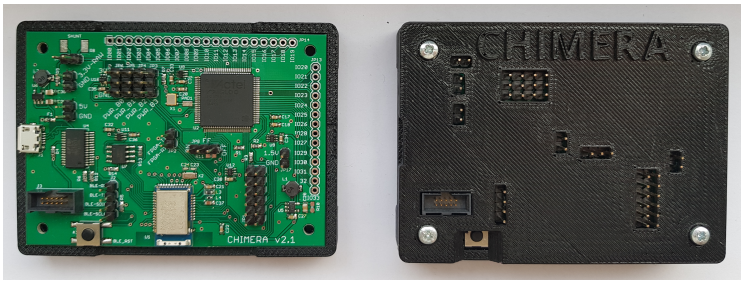


Fig. 1. Chimera Platform.

the rapid roll-out of IoT applications. Examples include; Particle.io[1], Electric Imp[2], Libellium[3] and VersaSense[4]. The ability to deploy IoT devices for extended periods of battery-powered operation reduces costs and management complexity. However, this lifetime can be limited when hardware updates become necessary.

The need to perform edge analytics (e.g., on vibration, audio, or image data) and support complex cryptographic primitives (e.g.SHA, AES, RSA) demands high-performance computation at the edge of the network. However, the Micro Controller Units (MCUs) used in IoT platforms are often too limited to support complex in-situ processing, which hinders the deployment of edge analytics and the roll-out for new security features. Furthermore, researchers and attackers continue to uncover new security bugs and threats, some of which cannot be efficiently fixed in software, thus requiring a more radical change in their hardware architecture [27]. In short, these platforms are tailored to do a specific set of tasks efficiently and cannot cope with the major updates/changes in application or security protocols.

Single-board Linux computers and FPGA development boards offer higher computational performance. While generic platforms such as Raspberry Pi and BeagleBone are capable of supporting a wide range of applications, their power consumption precludes their application in battery-powered scenarios. On the other hand, FPGA based platforms are mostly used in powered scenarios such as data centers [16, 45] to increase performance and provide flexibility [5, 6, 24]. They are not utilized as an end-device in IoT or wireless sensor networks due to their cost and power consumption.

Extensive prior work on FPGAs in embedded applications has shown their potential as: multimedia co-processors [46], cryptographic accelerators [36] and flexible network accelerators [49]. However, as of today, this work has yet to break through into main-stream IoT platforms. We identify two key reasons for this delay in adoption: *(i.)* the high power consumption of most FPGA-based platforms and *(ii.)* the lack of systematic support for FPGA use in contemporary IoT tool-chains. Chimera addresses these problems by providing an open-source FPGA-based platform that allows over-the-air reconfiguration of both software and hardware resources with a tool-chain that can optimize and reconfigure the FPGA cores based upon the demands of application software while guaranteeing multi-year battery lifetime on standard batteries.

To tackle this problem, we introduce Chimera, a flash FPGA based IoT device. We show that Chimera achieves its goals of adaptability and flexibility without sacrificing a long lifetime through the implementation and evaluation of a number of archetypal IoT applications. We demonstrate the capability of Chimera to tailor FPGA design dynamically depending on the change in application

[1]https://www.particle.io
[2]hhttps://www.electricimp.com
[3]http://www.libelium.com
[4]https://www.versasense.com

requirements and perform an application-level analysis to show the potential of utilizing an FPGA on battery-powered platforms. Our results show that: *(i.)* Chimera achieves more than one year of battery lifetime using four standard AA/LR6 batteries, (ii.) battery lifetime can be extended by up to 30% by customising the FPGA image to suit application requirements, *(iii.)* the costs of online or offline reconfiguration are acceptable even in the absence of partial reconfiguration support (worst case 0.002% of available battery life) and *(iv.)* executing processing-intensive algorithms (AES and SHA3) on the FPGA with optimised hardware accelerators achieves better performance in terms of power consumption, execution time and throughput compared to the software implementation on traditional processors.

The preliminary results of Chimera were published in our previous work [2], to which this paper provides the following extensions: *(i.)* additional information on the design of Chimera at the electrical and FPGA level, *(ii.)* a secure remote attestation case study to motivate the need for security accelerators and *(iii.)* a cryptographic hash function case study to provide a state-of-the-art comparison against existing solutions.

The remainder of this paper is structured as follows: Section 2 introduces the background and pertinence of our work. Section 3 describes design analysis as well as the hardware & software details of the platform. Section 4 explains the evaluation and the results. Section 6 discusses directions for future work, Section 5 highlights related work and Section 7 concludes.

2 BACKGROUND

2.1 Field Programmable Gate Arrays

Programmable logic devices offer a number of advantages for the IoT. The ability to redefine the hardware functionality of an already deployed platform offers the promise of a tailored architecture for each task, allowing more agility and efficiency in a domain where resources are constrained. Among the various families of programmable logic devices, Field-Programmable Gate Arrays (FPGA) have become dominant due to the re-programmability, unlike Complex Programmable Logic Device (CLPD) and Programmable Array Logic (PAL). The configuration or programming of an FPGA begins with a digital design of the required functionality in a hardware description language (HDL) such as VHDL or Verilog. Various functionalities from basic digital XOR gates to complete processor or micro-controller designs can be implemented [11]. These designs are translated into register transfer level (RTL), then to a netlist, and finally to a bitstream that is used to program the FPGA. Various limitations have prevented FPGAs from being applied in IoT systems. Among them, the lack of systematic tool-chain support and, in particular, the high power consumption [56] of FPGAs remain key challenges, which this paper aims to address.

2.2 Flash FPGAs

FPGAs typically implement combinatorial logic through Look Up Tables (LUT), which hold the system configuration. These LUTs are primarily built out of volatile memory (SRAM) and require a constant electric charge. SRAMs are used in FPGAs to reduce the read/write overhead and increase the overall performance since they are widely used in high-throughput and/or computing-intensive applications. However, in case of a power-down or a reboot, the Look-Up-Table is loaded from an external non-volatile memory or programmed via an external device due to the volatile nature of SRAM memories. Therefore, SRAM FPGAs are not utilized in battery-powered devices and classified as high-end computing platforms. Recent advances in Flash-based FPGAs are increasingly becoming competitive with S-RAM models. As Flash FPGA is based upon non-volatile memory, no power is used to maintain the hardware design, enabling the chip to be powered down and restarted

Table 1. Cost of reconfiguration for involved modules

	Processing Unit	Wi-Fi Module	Bluetooth Modules	Non-Volatile Memory	Elapsed Time	Cost
Reconfiguration over Wi-Fi	7mA	Rx:50mA	-	Writing:10μA	45sec	0.4925mAh
Reconfiguration over Bluetooth	7mA	-	Rx:5.4mA	Writing:10μA	36sec	0.2635mAh
Offline Reconfiguration	7mA	-	-	Reading:20μA	31sec	0.2325mAh

without reprogramming. Together, these features enable low cost and low power flash-based FPGA designs [21].

2.3 Requirement Analysis

We identify five key requirements in contemporary IoT systems which can be address with a flexible and re-configurable hardware platform.

- **Adapt to changes in application requirements**: Emerging IoT applications have specific requirements concerning networking, system security and edge analytics capabilities. This diversity in requirements demands end-devices which can be adapted rapidly to a new set of tasks through local or over-the-air updates.
- **Increase the reuse of IoT devices**: In contemporary IoT environments, such as smart building and factories change is inevitable as new machinery is added and rooms are re-purposed over time. However, these changes shorten the average life-cycle of an IoT device, causing substantial electronic waste as IoT devices are replaced. We advocate that IoT devices must be capable of evolution to prevent their replacement in such scenarios.
- **Tackle contemporary security issues**: Discoveries of new security flaws in IoT devices has become a consistent theme over the last few years. Implementing countermeasures is a costly operation that often requires manual intervention or even end-device replacement in the case of low-cost wireless MCUs. On the scale of a smart hospital or the reorganization of a university these costs could drastically increase costs.
- **Manage hardware and software heterogeneity**: The resource constrained nature of the IoT has created a diverse landscape with vast application specific solutions. This diversity formed a unique domain where there is no standard protocol and technology adopted. We believe that a reconfigurable IoT platform offers the potential to tackle this heterogeneity.
- **Efficient task execution**: IoT devices may be required to implement different applications over their lifetime, each of which demand different edge processing. While the tasks such as encryption (SHA3, RSA), signal processing (FFT, DFFT) or data analysis (machine learning, neural networks) can be still performed on this devices, execution times and thus power consumption may increase drastically. A reconfigurable platform that can run such tasks on optimized hardware accelerator is needed to optimize the device lifetime and performance.

3 ARCHITECTURE

The design of the Chimera platform aims for three objectives. First, Chimera must enable the reconfiguration of its hardware and software functionality, both locally and over-the-air, while minimizing the energy cost. Secondly, Chimera aims to design a power-efficient device with minimal active and sleep power consumption, which can last for multiple years on standard battery pack. Finally, the Chimera platform must be easy to use, with an open and accessible tool-chain in order

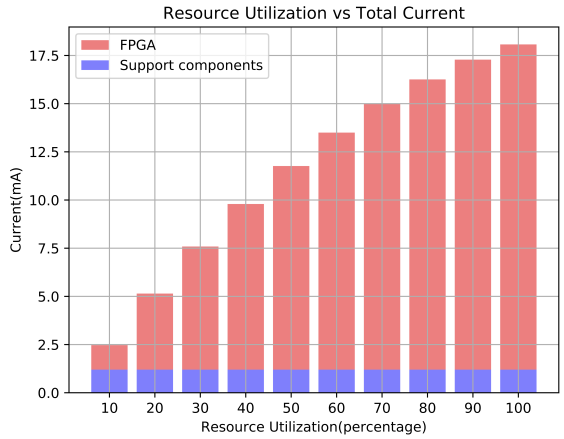


Fig. 2. Influence of resource utilization on power consumption

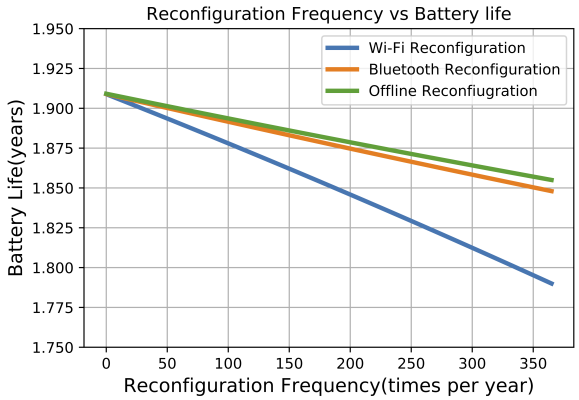


Fig. 3. Effect of Reconfiguration Frequency over battery life.

to minimize thresholds for its adoption. The following section presents a more in-depth analysis of our goals and how they have been achieved in our hardware and software design.

3.1 Design Analysis

As discussed previously, the primary goal of Chimera is to offer low-cost dynamic hardware and software reconfiguration. Using a Flash FPGA as the main computing unit is therefore a logical choice due to their cost, performance and power efficiency. After analyzing the market and comparing various products, we selected an IGLOO Nano FPGA from Microsemi [34]. Figure 2 shows the power consumption results of this platform plotted against increasing resource utilisation. The various components necessary to operate the system and their associated current consumption is presented in Table 1 for reference. These results show low power consumption overall and a logarithm-shaped evolution as resource usage increases, confirming that this FPGA is good choice to be used as core unit of our architecture.

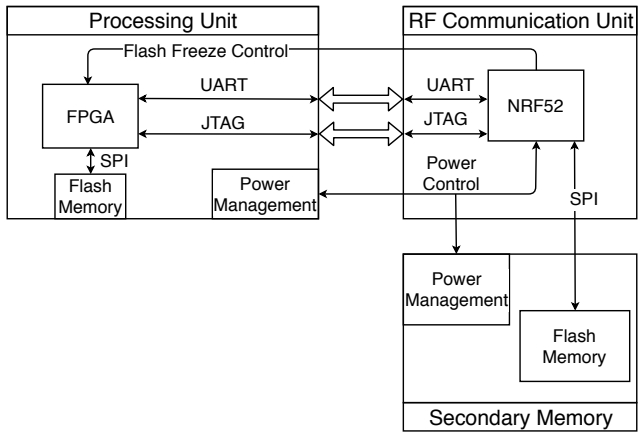


Fig. 4. Hardware architecture of the Chimera platform.

Support for over-the-air reconfiguration is required to support the in-situ evolution of the hardware/software platform. Moreover, the design should minimize the overhead of FPGA updates by locally caching reconfiguration images that can then be updated *offline*. To support this, the Chimera platform must include an on-board memory module to store FPGA images. Various RF modules are tested by performing a reconfiguration process to evaluate and select the most suitable radio module for the platform. In the reconfiguration process, the device requests to download an FPGA image from a connected gateway, stores it locally, and reprograms the FPGA by transferring the image stored in the external Flash to the programming array of the FPGA. Table 1 compares various candidate modules and indicates the benefit of using a Bluetooth-based approach. Figure 3 shows the impact of reconfigurations on battery lifetime for increasing reconfiguration frequencies.

From an application development perspective, Chimera should minimize barriers to entry in terms of programming and managing the platform. Programming on an MCU is a common practice in the application domain that we target. Yet, it is well-known that emulating an MCU in an FPGA is far from trivial. In order to position Chimera as an experimentation platform, a familiar and comfortable development environment should be provided to users. We analysed the implications of implementing a softcore microprocessor in our FPGA (the MSP430, as discussed in Section 3.3). Having an experimental platform that can offer this facility is essential to enable a large community and support.

3.2 Hardware Design

The main blocks of hardware architecture and their interactions are illustrated in Figure 4. The *RF Communication Unit*, embeds a Nordic Semiconductor nRF52832 SoC, which supports: Bluetooth Low-Energy, ANT, and other proprietary 2.4GHz protocols, as well as an ARM Cortex M4 processor core. This block interfaces with the rest of the infrastructure to provide over-the-air reconfiguration images for the FPGA. During this process, the RF Communication Units interacts with the flash memory via a Serial Peripheral Interface (SPI) to store a newly received image. It also allows for direct reconfiguration of FPGA, without the intermediate step of storing the image. The interface used to program the FPGA is a standard JTAG port. In addition to these reconfiguration tasks, the RF Communication Unit manages the duty-cycle and power of the other two units. As shown in Figure 4. It is responsible for powering the units up and down and serving as a sleep mode timer for the entire platform. This choice was guided by the reduced power consumption of the SoC (2μA

Table 2. Details of Images for IoT Applications

Application Image	Resource Utilization	Memory	Hardware Modules	Active Power
Standard Sensor	41%	1kB RAM 1kB ROM	GPIO	14mA
Control and Relay	57%	4kB RAM 4kB ROM	GPIO,UART	17mA
Real-Time Embedded OS	99%	4kB RAM 4kB ROM	GPIO,UART WDT,TIMER	22mA

running a watch-dog timer). Having a timing element allows for the implementation of custom firmware for smarter actions during the active or sleeping phase while draining so little power makes what the RF unit a perfect candidate for handling power management.

The *Main Processing Unit* is centered around an Igloo Nano FPGA from MicroSemi, one of the lowest power FPGAs on the market with a power consumption of 2.2μW (sleep mode), 16μW (flash freeze mode), and 1-30mW (active mode depending on the resource-use). We combined the computing unit with a dedicated 2Mb Flash memory chip to provide support for storing application data, software modules, and sensor logs.

The *Secondary Memory Unit* provides storage for the reconfiguration process of the Main Processing Unit. The current prototype has a memory capacity of 2MB and is able to store up to 4 different reconfiguration images for the Flash FPGA. It draws 2.5-20mA, 2-10mA, and 5μA of dc current respectively for a reading cycle, writing cycle, and sleep mode with a clock speed varying between 20MHz and 120MHz.This allows Chimera to economise on reconfiguration costs by avoiding a wireless transmission every time an image change is requested.

3.3 FPGA Design

3.3.1 The Processor Core. the Chimera FPGA design consists of three hardware modules. First, an open source implementation of the TI MSP430 archictecture, the Neo430 from the OpenCores project [35] is used as main processing unit. Neo430 is based on the MSP430 ISA and provides 100% compatibility with the original instruction set, which supports both GCC and LLVM, while offering an open-source, highly customizable micro-controller core. Second, a power management module; the Flash Freeze core, is included to safely put the FPGA into sleep mode while retaining SRAM and register information. Two cryptographic primitives (AES and SHA3) are also designed as a memory-mapped accelerator that is integrated with Neo430 core. Our design uses memory mapped IO due to its lower resource requirements in comparison to a serial interface or an on-chip bus (UART, SPI, Avalon[5] or AMBA[6]).These modules are combined to develop three FPGA designs used on the platform:

(1) The basic Neo430 core with flash freeze module can be used to implement archetypal IoT applications. We used various hardware configurations with different RAM, ROM memory sizes and sensing peripherals to tailor the CPU design for different IoT applications.
(2) An extended Neo430 core with AES module and a flash freeze module that is suitable for implementing efficient end-to-end secure IoT applications.
(3) An extended Neo430 core with SHA3 and flash freeze module to optimize SHA3 process for applications where establishing a secure communication channel is required.

[5]https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/manual/mnl_avalon_spec.pdf
[6]https://www.arm.com/products/silicon-ip-system/embedded-system-design/amba-specifications

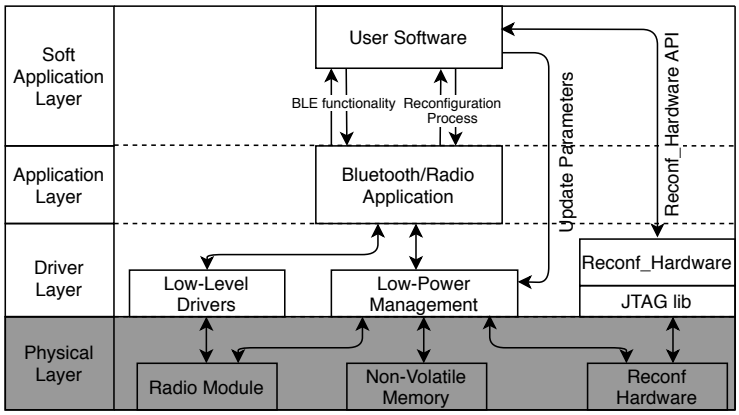


Fig. 5. Software Stack of the Chimera platform.

3.4 Software Design

Chimera is a reconfigurable hardware platform combined with a software architecture that is designed to provide a complete IoT solution for both high-end and low-end applications. The software architecture provides complete abstraction and flexibility to the user application through software interfaces (APIs) for reconfiguration and power management.

3.4.1 Software Stack. The software architecture of the Chimera platform can be seen in the Figure 5 along with supporting hardware modules. Software is shown in white while hardware is shown in gray. The Chimera architecture consists of four layers: hardware, drivers, application and soft-application.

(1) The physical layer is formed by the RF Unit, Memory and FPGA modules as previously described.
(2) The driver layer provides a software interface to these hardware modules, an optimized power management library and a JTAG library to reprogram the FPGA.
(3) The application layer contains the radio/bluetooth application to connect to the back-end for the reconfiguration process. The user application can handle the reconfiguration process through the API. Using radio application API, the user application can request different FPGA images (cores) from the back-end based on the application. In addition, the radio application layer also provides a separate software interface to the soft application layer for bluetooth connectivity to transmit sensor data when it is needed.
(4) The top layer called the soft application layer defines the application implemented in the soft processor core. The soft application layer implements the user software and can manage hardware components without interacting with bluetooth/radio application. The re-configurable hardware is transparent to the user application layer. Therefore, the user software can reprogram the FPGA via the reconf_hardware API depending on the application as shown in Listing 1. An external image from the backend or a local image which is saved in the flash memory, can be used to reprogram the FPGA directly by the user software. The reconfiguration process is detailed in section 3.4.2.

Customizing the FPGA design affects not only the power-consumption but also the software stack. It allows users to bare-strip the platform to the minimum in order to implement a various cores

(FFT module, image-recognition core, etc.) and to fine tune the performance/power parameters based on the application.

Listing 1. Reconf_Hardware API

```
//RECONF_HARDWARE API

//initialize the FPGA hardware and low-level drivers
void reconf_hardware_init();

//scans device id for the FPGA hardware
uint32_t reconf_hardware_scan_idcode();

//erases a core image from the internal flash memory
uint8_t reconf_hardware_erase_core_from_mem(uint8_t *
    p_flash_memory_address, uint32_t size);

//Saves an externally received core to the internal flash memory. Should
    be used with relevant BLE/RADIO APIs
uint8_t reconf_hardware_save_core_to_mem(uint8_t *p_flash_memory_address,
     uint32_t size);

//FPGA reconfiguration functions
uint8_t reconf_hardware_erase_FPGA();
uint8_t reconf_hardware_program_FPGA(uint8_t *p_memory_address, uint32_t
    size);
uint8_t reconf_hardware_verify_FPGA(uint8_t *p_memory_address, uint32_t
    size);
```

3.4.2 Reconfiguration Process. One of our main purpose is to develop a platform which is able to adapt its hardware and software to the ever-changing and heterogeneous nature of the IoT. For power consumption related reasons, we wanted to design a system which is able to perform online (remote) as well as off-line reconfigurations of the FPGA. To that extent, we articulated the three units of the platform (Main Processing Unit, RF Communication Unit and Secondary Memory Unit) in a way allowing us to perform both types of reconfiguration in a power efficient manner. It is important to note that a complete reconfiguration requires a time that is linear to the size of the bit-stream and during this process the entire chip is inoperable. The radio module is in charge of receiving the new FPGA image when it is needed and to initiate the reconfiguring process. This image consists of a bitstream representing the new hardware and software design of the FPGA and intended for the Programming Array, inside the FPGA. The bitstream can be obtained from two different manners, depending on the type of reconfiguration.

- **Online Reconfiguration:** the MCU receives a given bitstream through the radio module , and uses the SPI interface to forward and store it in the dedicated flash-memory for images. When the image transfer is finished and stored in memory, the RF module is shut down and the MCU will read the flash-memory and initiate the transfer of the bitstream towards the programming array of the FPGA. If the image does not need to be stored locally and simply programmed into the FPGA as soon as possible, the MCU can skip the write and read sequence in the memory and directly reprogram the FPGA with the bitstream received by the RF unit. It is worth mentioning that this method will not allow the actor to later

retrieve and store the newly implemented image. This image is immediately implemented and operational but will not be available for further analysis or once it has been rewritten by another reconfiguration cycle.
- **Offline Reconfiguration:** the MCU will select a desired image in memory and will read it bit by bit and forwards them along to the FPGA. The dedicated memory can store up to four images, allowing the platform without having to request a dedicated reconfiguration image from a distant server. This mode allows to perform faster reconfiguration at a cheaper energy cost than in the Online mode due to the inactivity of the radio module.

The extent of the reconfiguration process goes beyond low-level drivers, the soft-processor which is part of the FPGA's image can be reconfigured in Online mode on a hardware level. As the soft processor core includes an optional bootloader, we can remote update the code run by the emulated MCU on the FPGA.

3.4.3 Energy Management. To optimize the power consumption in resource-constrained devices, one cannot rely solely on the hardware design or the software design. Chimera aims to fine tune both software and hardware parameters by applying low-power policies. Following strategies are implemented to reach multi-year battery life;

- The power lines of major components in the platform are isolated from each other and can be turned off and on by individual power switches. Therefore, the software has a fine grained control over the power of Chimera modules and it is optimized through the Low-Power management driver where power consumption of all major components including the RF Communication Unit, the Main Processing Unit and the Secondary Memory Unit can be controlled. To that extent, the RF communication unit implements configurable low power timers to adjust the active periods of the hardware components. The Low-Power management driver supports different power modes by optimising the active cycles of various hardware components by default. Additionally, the user software can configure the low-power timers to fine-tune the power consumption depending on the application.
- As mentioned previously, Chimera utilizes a Flash FPGA to reduced the power consumption not only in active period but also in sleep period. The FPGA used in the platform has a special feature called flash freeze in which FPGA consumes substantially low power while retaining all the information in the memories and registers with rapid recovery to active (operating) mode. While Flash Freeze mode is active, no power is consumed by the I/O banks, clocks or JTAG pins and the FPGA consumes around 5μW. Activating this mode controlled by a low power timer which is programmed by RF communication Unit and follows low duty cycle principles.
- The low-power management driver also implements a low duty cycle policy to reduce the power consumption aggressively. Duty cycling is a widely used policy in wireless sensor devices to reduce the overall power consumption [59]. The sensor nodes constantly switch states between active mode, where the sensor node collects information, processes, stores or sends it, and sleep mode, where all the indispensable elements are reduced to the lowest power consumption by disabling unused features, and the rest of the components is in low power modes or turned off. Low Power Management library applies the duty-cycle principle mainly on the Processing Unit (FPGA) and the RF communication Unit via two different low-power timers. Optionally, the user software can adjust or disable duty-cycle period depending on the application by configuring these timers.

3.4.4 Toolchain and Environment. To develop software and FPGA modules for Chimera, we used the standard Microsemi Libero SoC Design Software. Low level drivers and MCU applications are

Table 3. Composition details of the different cores used in the evaluation scenario.

	Resource Utilization	Memory	Hardware Modules
Base core	50%	1kB RAM 1kB ROM	GPIO, UART
AES core	99%	2kB RAM 2kB ROM	GPIO, UART, AES

developed using a modified version of Arduino toolchain. For the soft application layer the standard msp430-gcc tool chain is used. The aim is to provide a familiar code base with a simple tool-chain to reduce the software development complexity for users.

4 EVALUATION

Our evaluation of the Chimera platform has three objectives: *(i.)* evaluating the low-level power characteristics of Chimera, *(ii.)* evaluating the performance of Chimera in a realistic remote attestation case-study and *(iii.)* comparing with existing solutions to validate energy and performance benefits.

4.1 Power Consumption and Reconfiguration Characteristics

This scenario-based evaluation focuses on a smart office building that is equipped with a pervasive deployment of Chimera devices. These devices may be re-purposed over time in order to support a range of application scenarios. As the application, we optimise the configuration of deployed Chimera devices to match changing application requirements that were not anticipated at design-time.

The scenario begins with a simple smart office sensor telemetry scenario which measures and transmits environmental parameters such as temperature and humidity from laboratories, meeting rooms and offices via an IoT gateway. Sensor devices are optimised to achieve multi-year battery life in order to minimise the time and cost of performing battery upgrades. Following a change in corporate data protection rules, the data collected by sensors is re-classified as private and confidential during working hours from 8AM to 8PM. An extra layer of security therefore needs to be applied to ensure the confidentiality of the sensor data. Running AES in software is power inefficient and, as these security requirements were not considered at design time, no AES hardware accelerator is present. To accommodate this change, the Chimera configuration is reconfigured at runtime to introduce hardware support for AES during working hours, reverting to the basic sensor telemetry configuration at all other times.

4.1.1 Methodology. We evaluate the power consumption of the above mentioned smart building scenario based upon current trends in the active cycle and evolution of the battery charge over two days.

As described above, we tested three different configurations on the end-devices. One with the basic sensor telemetry core, the second with an added AES core that encrypts data prior to sending it, and a third which reconfigures itself every 12 hours to switch between the two previous configurations. The details of the FPGA images corresponding to these configurations can be found in Table 3. The duty cycle of the application is set to 1%, which we believe is representative for wireless sensor network applications [25]. The duration of the active cycle wherein sensor data is collected, processed and transmitted takes 200ms, which gives a 20 second sleep period based on the 1% duty cycle. In the second case, an AES core is permanently added for encryption of the collected data. The active cycle duration expands to 400ms to encompass the necessary duration to

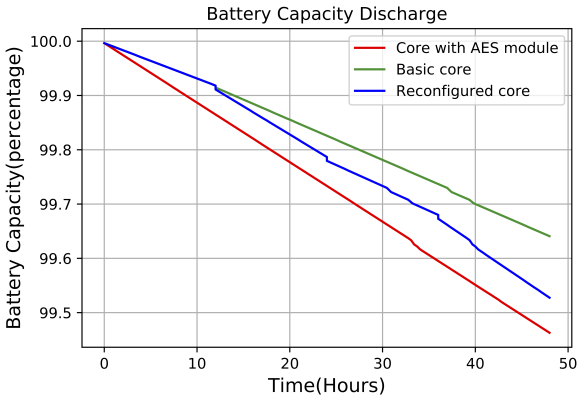


Fig. 6. Battery Charge Trend for three different scenarios.

perform AES calculation then sending the encrypted data before going back for a 40s cycle of sleep. Each device is connected to an external coulomb counter to measure current consumption and battery use. For detailed current traces, a digital multimeter with data-logging capability was used. The platforms use a pair of retail store standard AA/LR6 batteries with a total charge of 4800mAh. We run these experiments for the scenario duration of 48 hours and analyse the collected results in the following subsection.

4.1.2 Results. This overall performance of the platforms is illustrated in Figure 6, where the battery discharge curves of the platforms are represented. As described in the Design Analysis Section 3.1, the resource utilization of the FPGA core has a major effect on the power consumption of the processor. Even though the platforms have a similar duty-cycle of 1%, there is a significant difference in the power consumption of each configuration. By the end of our evaluation scenario, the basic configuration used the 0.36% of the battery (1.5 years of battery life). The AES-accelerated core finished the run after using 0.54% of the battery (1 year battery life); a one third decrease in battery life compared to the baseline core. As can be seen from this evaluation, utilizing the extra modules in the design even when it is not needed has significant battery life costs. The third platform, the alternates between the two configurations as needed. It finished with 0.48% of battery charge (1.2 years battery life). By re-configuring every 12 hours, battery charge is conserved in comparison to the permanent use of AES core, even when considering the costs of reconfiguration (which can be seen in the blue curve every 12 hours). For the purposes of this evaluation FPGA images were loaded onto flash (i.e. offline reconfiguration).

Figure 7 provides a detailed view of the current trend during the active cycle. Sensor reading, serial transmission to the RF Communication Unit and Bluetooth communication are the main tasks that affect the active cycle of the baseline core represented in green. The graph shows the following tasks for basic core; the main processing unit reads the sensor data through the Analog to Digital Converter (ADC), this is not visible in the power profile due to the low power ADC, sends the data via serial interface (UART) to the RF unit (A1), the RF unit then initiates the Bluetooth transmission of sensor packet (A2)and busy-waits for evaluation purposes (A3).

On the second core (shown in red in Figure 7), current consumption rises higher during the period A0 due to the AES computation. The AES core is then turned off, and the RF Communication Unit is then turned on for data transmission, as in the first use case. The rest of current activity is identical to the first use case. The effects of an extra AES module are clearly visible.

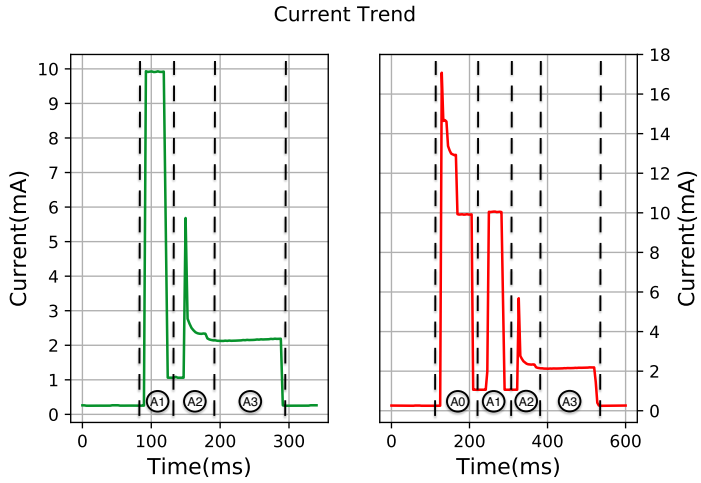


Fig. 7. Current trend during an active cycle. Sensor core (green) vs Sensor core + AES core (red)

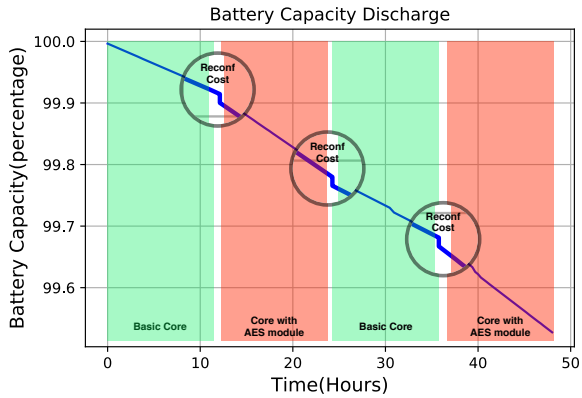


Fig. 8. Battery Charge Trend during reconfiguration.

Figure 8 shows the power consumption of reconfiguration between the basic core and the AES core. Chimera offers two types of reconfiguration process; offline and online as described in Section 3.4.2. In Offline reconfiguration, one of the core images previously saved in the secondary memory unit is programmed into the FPGA array using the JTAG interface. In online reconfiguration, an image is directly loaded directly onto the FPGA over the network using JTAG with no intermediate storage. The current profile of both reconfiguration processes is shown in Figure 9. The Offline process starts by erasing the previously used image iduring the period denoted A1. In period A2, the new image is read from the memory and programmed in the FPGA. Period A3 verifies the newly programmed image and during period A4, the FPGA's clock is programmed and the core is run for verification purposes. Online reconfiguration increases the duration and overall power consumption of reprogramming due to the use of the RF communication to receive

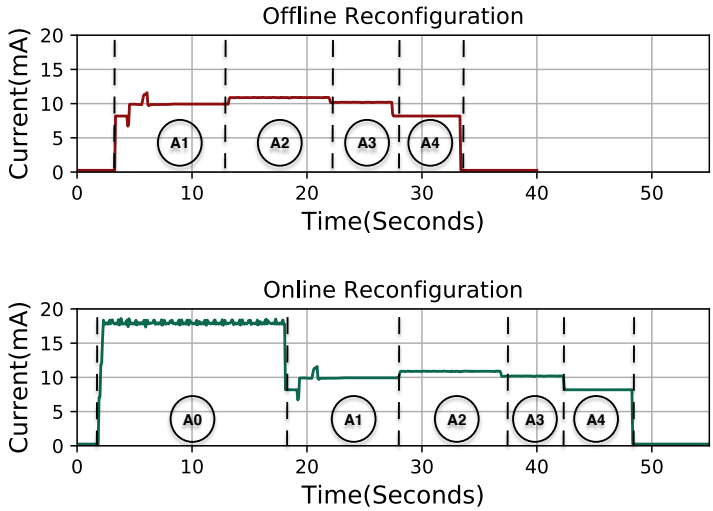


Fig. 9. Current trend during reconfiguration.

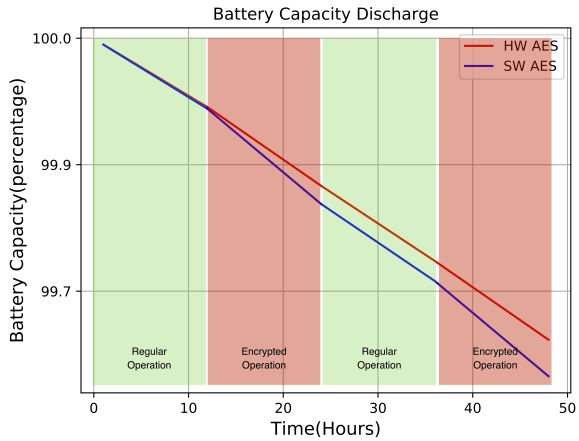


Fig. 10. Battery Charge Trend comparison on hardware and software AES implementation

the new FPGA image. This can be seen in period A0, where the RF Communication Unit wakes up, downloads the reconfiguration image and stores it in the secondary memory unit. The overall power consumption of reconfiguration is 61μAh and 95μAh for the Offline and Online processes respectively.

Figure 10 shows the evolution of battery consumption of (i) the core with hardware AES (HW AES) and software implementation (SW AES) without any reconfiguration process occurring on the

devices. The software AES functionality is implemented onto the basic core to provide a comparison between two different approaches. What stands out in this figure is the difference between two implementations on encrypted operation. On the one hand, using a similar portion of FPGA resources leads to a similar power consumption profile on regular operation, on the other hand, the significant difference on AES performance influences negatively on power consumption even within a short span of time. Reconfigured core still reach longer battery life while providing AES functionality when it is required despite additional cost of online reconfiguration on battery. These results would seem to suggest that not only reconfiguration but also designing a core according to the application increases efficiency and extends the battery life of Chimera platform.

4.2 Remote Attestation Case Study (AES-128-ECB)

The second aspect of the evaluation is based on support for a remote attestation scenario. In recent years, IoT networks and devices has become important targets for various cyber-attacks that are causing a growing amount of financial and, in some cases [30], physical damage.

Remote Attestation provides a secure software mechanism to detect and mitigate misbehaviour of compromised devices remotely [1]. A trusted entity, the *verifier*, regularly monitors the current state of a remote untrusted device, the *prover*, by computing a secure checksum of its software image and current state. To generate this secure checksum, the prover must has the memory with a cryptographic function such as: SHA, AES, RSA [51]. As these cryptographic operations are inefficient when implemented in software and they must operate over the whole program memory, they are good targets for hardware acceleration.

In this scenario, the network owner decides to enhance their security by deploying remote attestation support. We therefore show how Chimera can support this by providing optimised support for remote attestation by evaluating the performance impact of deploying hardware acceleration for remote attestation in comparison to realising this functionality in pure software. Key performance factors include: performance, power consumption and effect of overhead on Bluetooth throughput.

4.2.1 Methodology. We implement a simple remote attestation scheme where a sensor device regularly computes a checksum on different part of memory and verifies it with a trusted entity via BLE communication. The scenario was tested with; *(i)* the basic core extended with a hardware AES accelerator and *(ii)* with the basic core along running a software AES implementation. We used the Electronic Codebook mode (ECB) of AES with a key size of 16 bytes and plain text size of 16 to 1024 bytes to evaluate both hardware and software primitives.

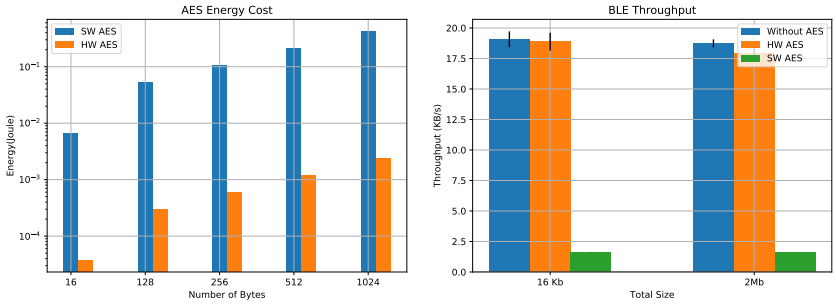
We also investigate how the the execution time of these primitives affects radio (BLE) packet throughput for those networks that require encrypted communication.

Each device used same batteries as described in the previous part of evaluation and a digital multi-meter was again used to measure the power consumption profile of the platforms. These experiments run on the platforms with different memory sizes (16b, 128b, 512b, 1024b, 16Kb, 2Mb) repeatedly (10-20 times depending on the experiment). In order to show the impact of FPGA configuration on battery-life, we used the same configuration as the smart build scenario in battery charge trend (48 hours period with 1% duty-cycle and 4800 mAh battery).

4.2.2 Results. To quantify the impact of FPGA configuration on performance (execution time), we first added a custom clock-cycle counter the the CPU core to measures the number of clock-cycles elapsed between two points in the software. As can be seen in Table 4, the HW and SW AES implementations have completely different characteristics in terms of execution time, code size and extra hardware resources. The hardware AES module consumes more power due to higher usage of FPGA resources as discussed in Section 3.1. However, HW AES drastically reduces code

Table 4. AES implementation comparison

	Execution Time (cycles)	Code Size (bytes)	Extra FPGA Resources (D flip-flops)	Extra Memory Blocks (kbit blocks)
HW AES	978	932	2437	none
SW AES	112628	2100	none	16



(a) Energy consumption of AES (b) BLE throughput

Fig. 11. Application level performance characteristics of different AES implementations

size; the memory footprint of the software implementation is more than two times larger. Due to the need for extra program memory, SW AES uses 94.61% of available FPGA resources, while HW AES utilizes 99.86%, resulting in similar active power consumption. Although both platforms have a similar power profile in active mode, the difference in execution time is the major influence on power consumption. The results in Figure 11a show that HW AES consumes two orders magnitude less energy than SW AES regardless of number of bytes being encrypted.

As shown in the Figure 11b, the poor execution time of software AES significantly reduces BLE throughput. The graph shows the throughput in Kb/s when encrypting and streaming 16kB (left) and 2Mb (right), which averages 19.06 Kb/s and 18.7 Kb/s respectively. Compared to sending unencrypted data, HW AES only causes only a 1% drop in throughput, whereas the software implementation causes a drop of over 90%.

4.3 Cryptographic Hash Function Case Study (SHA-3)

The third and final part of the evaluation focuses on a Cryptographic Hash Function (CHF) case study. A CHF is a one-way function that maps data from an arbitrary size domain into a fixed size co-domain. These algorithms are widely used in information-security to verify data integrity through the creation of digital signatures and Message Authentication Codes (MAC). However, running a CHF algorithm is costly in terms of power consumption and execution time, which has prevented their widespread adoption in the IoT. In this section, we evaluate the performance of Chimera against popular IoT platforms using a CHF case study using SHA3 [14].

4.3.1 Methodology. We evaluate the impact of the SHA3 case-study on three separate platforms based on power consumption and execution time over a secure sensor task to provide a state-of-the-art comparison. The scenario was tested with *(i)* three commodity microcontrollers (MSP430 [23], Atmega328 [10] and At32u4 [9]) to provide a benchmark for a common sensor device, *(ii)* an SRAM based FPGA platform (using the Xilinx Spartan6 FPGA) as a benchmark for the FPGA

Table 5. Execution time of SHA3 task on different platforms.

Platform	Execution Time (cycles)	Throughput (kB/s)
FPGA	282	113475
at32u4	6275	5099
atmega328	6279	5096
msp430	8500	3764

based platforms [4][49][28] and *(iii)* the Chimera platform. We use the Keccak [7] implementation of SHA3 [14] in software on the TelosB mote and an open source SHA3 hardware accelerator [22] on both FPGA based platforms.

The evaluation task is divided into three sub-tasks; *(i)* a sensor task in which the data is collected via an analog sensor, *(ii)* a SHA3 task which is used to establish a secure connection (as described in [47]) and *(iii)* a sleep task for the purposes of duty-cycling. As before, all three devices implement a 1% duty cycle. The experiments run 100 times and the average execution time and energy values are used in the results.

4.3.2 Results. Table 5 shows the results of a pure computational test on three different popular micro-controllers and an FPGA. All processors were clocked at 20 MHz. Since the FPGA utilizes a combinational logic unlike the other micro-controllers, it is far faster than the software based approaches (results are identical for the flash and SRAM based FPGAs at the same clock speed).

Figure 12 shows the energy consumption for a complete wireless application. As seen in the figure, the energy profile is different for each platform and for each task. As expected, the TelosB mote is the most power-efficient platform during the sensor task and the sleep period since it is specifically designed for low power sensing operations. While Chimera consumes more power than TelosB, it consumes significantly less power than the SRAM FPGA based platform due to the superior power profile of flash FPGAs. In addition, in the sleep period, the power saving features of Chimera (as described in Section 3.4.3, in combination with the use of non-volatile memory in the flash FPGA result in a significantly lower power profile for Chimera than the flash FPGA. On the other hand, TelosB is extremely *inefficient* when performing the *SHA3 Task* due to its limited processing capacity. The execution time for the SHA3 task is considerably longer compared to the FPGA approaches, which causes the TelosB mote to consume more power compared to these platforms during SHA3 task. As can be seen from the results, Chimera bridges the gap between low power devices and high-end platforms by providing a low power energy profile while allowing hardware accelerated operations on demand with over-the-air updates.

4.4 Discussion

The evaluation results demonstrate that Chimera meets the requirements highlighted in Section 2.3. Specifically, Chimera significantly increases the performance of complex tasks such as encryption, signal processing, or data analysis while providing a multi-year battery lifetime. In addition, it provides low-power hardware/software updates to adapt the platform to suit the changes in application requirements. As shown in the evaluation, hardware design can be reconfigured with a small cost of battery life, which directly affects the life-cycle of an IoT device.

We wish to highlight four known limitations in the current version of the platform that we plan to address in the future;

[7] https://keccak.team/index.html

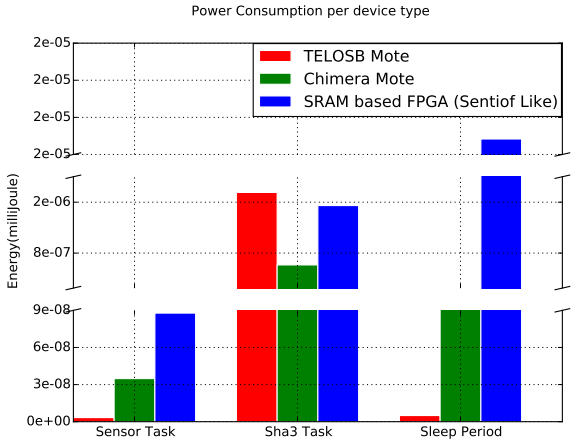


Fig. 12. Energy Profile of different task during the CHF use-case.

- *The number of FPGA images* that can be used in offline reconfiguration is limited due to limited memory. Chimera supports up to four images in the current version. Therefore, it covers a limited set of functions without online reconfiguration. We will use larger flash memory in the next version of the platform to cover a broader set of functions and thereby a broader range of applications.
- *Limited set of functions (SHA3, AES, FFT, DFFT)* are supported in the current version. Adding a new set of functions is relatively simple, depending on the use-case. There are various open-source projects available online [8], which can be implemented into the Soft-Core (neo430), which supports various interfaces (Wishbone bus, SPI, I2C, Memory-mapped I/O) to connect different hardware modules.
- *The battery life of the platform depends on the online reconfiguration frequency.* The application developer needs to make a trade-off between battery-life and adaptability. Authors of [28] show that FPGAs with partial reconfiguration feature reduces the reconfiguration cost on battery drastically. We would like to investigate the possibility of using a partially configurable FPGA in the next version of the platform to reduce the reconfiguration cost even further.
- *The FPGA has limited resources* in terms of logic elements and memory (Chimera utilizes AGLN250V2 FPGA from MicroSemi that has 3K logic elements). Implementing a larger processor that can support general-purpose operating systems (such as embedded Linux) is not possible with the current version of the platform. However, as future work, we would like to develop a platform with a larger FPGA (AGLE3000 with 35K logic elements) from the same vendor to provide support for larger processor architectures.

As presented in Section 3.1, resource utilization of the FPGA directly affects power consumption; in return reconfiguring the FPGA depending on the application requirements can optimize power consumption. The Offline reconfiguration process costs 0.0013% of the battery life, and the Online reconfiguration via Bluetooth costs 0.002%. As mentioned previously, up to four different FPGA images can be stored on the platform. Depending on the application environment, these images could cover any possible use case and can be used without an Online reconfiguration and the associated costs. Table 6 represents comparison of different platforms used in wireless

[8]https://opencores.org/

Table 6. Comparison of different platforms.

Platform	Processing Unit	Radio module	Active Current	Sleep Current
Cookie[28]	Xilinx Spartan3 + ADuC841	2.4Ghz RF transceiver supports Zigbee	45mA-72mA	18mA
Sentiof[49]	Xilinx Spartan6 + Atmel AT32U	2.4Ghz RF transceiver Supports Zigbee	47mA (max)	95uA
Telos-Tmote[42]	MSP430F1611	2.4Ghz RF transceiver Supports Zigbee	18.8mA (max)	5.1uA
MicaZ[33]	Atmel ATmega128L	2.4Ghz RF transceiver	18.8mA (max)	10uA
SunSpot[37]	Atmel ARM920T	2.4Ghz RFtransceiver supports Zigbee	35mA (max)	520uA
Chimera	Igloo Nano250	2.4GHz transceiver supports BLE and ANT	19mA (max)	250uA

sensor networks. Chimera consumes less active power than any reported FPGA based platforms [28],[49],[31],[46],[17],[29]. Although Chimera consumes more power in sleep mode than traditional MCU based sensor platforms, its repurposable hardware capabilities as an experimental hardware platform while still providing multi-year battery life offers an interesting trade-off.

As mentioned above in the limitations, the frequency of online and offline reconfiguration directly affects power consumption. The need for reconfiguration depends on the various factors, such as fixing a security bug, repurposing a device (as discussed in the use-cases), or providing new functionality to the existing hardware. Typical IoT applications such as sensors logging applications are updated 5 to 10 times per year depending on the vendor (Google NEST thermostat had 60 updates in the last nine years [20], or another IoT platform is updated 47 times in the last five years [38]). As presented in Section 4.1, Chimera provides multi-year battery life with an update frequency two times per day. This clearly shows that the platform can support typical reconfiguration frequency for IoT devices without compromising the device lifetime. Besides, these updates do not require hardware reconfiguration always. Typically, software updates are used for updating device drivers, fixing a software bug, or configuring the sampling time of a sensor, which are smaller in size (order of kilobytes [55]) compared to hardware updates (order of 100 Kilobytes 3.4.2) in the platform. Using BLE for reconfiguration drastically reduces the cost of the process in terms of power consumption.

The proposed platform provides different hardware architectures to support the different application requirements, as was mentioned in the application level evaluation. Recent literature provides various applications running on the edge devices (such as gateways and routers) or cloud servers along with distributed algorithms. Although these devices are powerful in terms of processing power, applications which require low latency, such as fast signal analysis or real-time control applications, are still challenging to perform on cloud servers or edge gateways due to challenging low-latency requirements. In addition, the edge or cloud servers process the data not at the edge of the network from the user perspective (the edge devices mostly placed between the sensor-devices and distant/cloud servers). Chimera aims to enhance the processing capabilities of battery-powered devices as close to the source of data as much as possible.

As shown in the evaluation, using a hardware accelerator for complex algorithms (such as AES, FFT, analyzing ECG signals, floating-point accelerator implementations) reduces power consumption drastically in resource-constrained devices compared to software implementation [52],[15],[7],[8]. The results also show that running such algorithms on hardware extends the battery life %50 in a multi-year application on resource-constrained devices. Besides, performing these processing-intensive algorithms directly affects the throughput of sequential tasks in the single-core sensor devices. Application-specific hardware architectures can reduce overhead, thus allowing

these devices to be utilized in sensor applications and in multimedia or real-time applications, which might open up a new dimension on information processing in sensor networks and edge computing. In essence, Chimera combines the best of high-end FPGA based platforms and low-power embedded devices.

However, there is still a gap between reconfigurable hardware platforms, such as Chimera, and software support for developers. Even though we tried to overcome this challenge using and configuring an open-source toolchain, there are still hardware and software packages missing to support different applications and algorithms as well as different FPGA vendors. Nevertheless, we believe that experimental platforms on reconfigurable hardware domain will help the open-source community to bridge this gap entirely in the near future.

5 RELATED WORK

Over the years, several academic research results have been published and tools have been proposed to introduce FPGAs in resource-constrained environment. None of these explored leads used FPGAs as main processing units but as accelerators for complex algorithms such as AES encryption, error correction or fast Fourier transform. Among them, Cyclops [46] is one of the very first application of using reconfigurable hardware on resource constrained embedded devices. It is developed as an extension for MICA Motes [32] as a smart vision sensor card that can be controlled by a host (in this case the MICA Mote) to bridge the performance gap between the resource constrained embedded devices and CMOS imagers, to that extend it utilizes a CLPD to accelerate image capturing process. Although it was specifically designed for embedded devices, the presence of the CPLD in the Cyclops platform still makes image capturing computing intensive task. Another work that utilizes a CPLD is called mplatform [31]. The authors designed a modular 4-module stack platform that is capable of processing real-time data while supporting dynamic reconfiguration. Moreover, A similar platform making use of reconfigurable hardware is Sentiof [49], a complete platform designed to be used in smart camera applications. It utilizes a high-end FPGA and a radio chip based on the IEEE 802.15.4 standard. The FPGA in this platform is meant to be used as an accelerator for different tasks in the image processing pipeline. Since all three platforms uses high-end CPLD and/or FPGA, respectively Xilinx XC2C256, XC2C512 and a Spartan-6 FPGA, their uses are unsuitable outside very specific applications, in particular for the resource constrained nature of battery powered wireless sensors.

Another research direction is targeting resource constrained end-devices in order to cope with the heterogeneous nature and constant change in IoT and applications. Krasteva et al. states that FPGAs can be used not only in high-end applications but also in low power sensor applications [28]. Cookie is a reconfigurable wireless sensor platform that utilizes a high-end FPGA with dynamic reconfiguration features to update hardware or software in run-time. However, remote reconfiguration is very power demanding and they reported the used radio technology (Zigbee) not suitable for remote reconfiguration. Moreover, this platform is not an embedded one, made to operate via a fixed power supply or an USB interface. A second work in that domain is a platform called PowWow [4]. Different from other platforms, it uses a low-power Flash FPGA, very similar to the one used in Chimera. Low-power by design, dynamic voltage and frequency scaling features are provided to minimize the power consumption. However in that case, it cannot be reprogrammed once it is deployed. These platforms were specifically developed for low-power IoT/WSN applications, yet their high power consumption during application, power demanding reconfiguration and non-versatile characteristics limits them to be used in the application domain targeted by Chimera.

In the semiconductor field, extensive research both in academia and industry has been pursued on new reconfigurable architectures and devices during the last decade. Field Programmable System on Chips (FPSOC) are one of the recent products in the market popular among academics

Table 7. Comparison of platforms with reconfigurable hardware

Platform	FPGA as main processor	Programmable device	Radio module	Dynamic reconf.	Main features	Active power	Sleep power
Cyclops[46]	no MICA2	CPLD	CC2420	no	imaging applications	64.8mW	0.7mW
mplatform[31]	no msp430	CPLD	CC2420	no	real-time applications	5mW-13mW	-
PowWoW[4]	no MSP430	FPGA Igloo Nano	CC2420	no	hardware accelerator	1-30mW	16uW
Cookie[28]	no ADuC841	FPGA Xilinx Spartan3	Zigbee	yes	sensor management	150-240mW	60mW
Sentiof[49]	no Atmel AT32U	FPGA Xilinx Spartan6	CC2520	yes	high-end application	158mW	0.313mW
Marmote[50]	no Smart Fusion	FPGA fabric in SOC	2.4Ghz RF frontend	no	used as SDR	287-852mW	71mW
uSDR[29]	no Smart Fusion	FPGA fabric in SOC	2.4Ghz RF frontend	no	used as SDR	1400 mW	71 mW
IEEE1451 based node[17]	no Telos B[42]	PSOC	CC2420	yes	analog accelerator	62.7mW + telosB power	6.6uW + telosB power

and industrial actors [40]. An FPSOC combines an FPGA fabric with several high-performance embedded processors. Gomes et al. [19] states that these devices can be used as an edge device to enable offloading computation from server side. Another work which makes use of FPSOC is a platform called μSDR [29]. The platform utilizes a flash based FPSOC device to scale down SDRs (software defined radios) in size, cost and power. Although the design and the main goals are similar to Chimera, the platform is specifically developed as an SDR. Due to not having Flash Freeze technology to clock-gate the FPGA fabric, even in deep sleep mode, this platform draws substantially higher current than ours and is thus not suitable for resource-constrained battery powered environment. On the other hand, Tanaka et al. [52] makes use of FPSOC to prototype a similar platform like Chimera. They state that accelerating tasks by using FPGA optimizes power consumption since tasks will be processed way faster than conventional CPUs. This vision aims to compensate the high power consumption of most FPGA-based platforms and allows them to be used in battery powered ones. However, FPSOCs are not as power efficient as Flash FPGAs or low-power microcontrollers yet. Therefore, it is hard to achieve multi-year battery life while having FPSOC hardware in IoT embedded devices.

On the developer side, in order to tackle the lack of systematic support for FPGA use and make them more appealing to work with, new software paradigms/architectures have been investigated and proposed. One proposition is a low-overhead FPGA middleware of Kirchgessner et al. [26]. They presented a middleware called RCMW that improves and enables application and tool portability. Another proposition was a complete toolchain called Click2NetFPGA [48], presented to bridge the gap between digital design of FPGAs and software development. This tool-chain converts Click C++ codes into a VHDL netlist. The essence of both papers is to bring reconfigurable hardware opportunities to the software world, by easing the handling and development processes, making them appealing for actors used to traditional MCU-based platforms. However, these approaches are not designed nor optimized for battery powered or resource constrained devices, making them not suitable for the application domain targeted by Chimera.

A comparison of major platforms with reconfigurable hardware in literature is presented in Table 7. As discussed previously, most of them utilize either an FPGA or a CLPD to accelerate a specific task (image processing, encryption, Fourier transform, etc) and none are using them as the main processing unit. In addition, most of these platforms are unsuitable for a wide range of IoT

applications due to being developed/used as dedicated hardware or/and high power consumption profiles.

6 FUTURE WORK

The initial results of Chimera motivate us to extend the work further. Although we used generic hardware components to support the various tasks, a more cutting-edge trade-off between costs and power performances in sleep and active mode could be investigated further. To that extend, one of the directions for future work on a software level resides in the power saving algorithms. Having a framework that can perform aggressive power savings operations based on an overall comprehension and integration of the task flow automatically would lead to shorter activity cycles and a more tailored approach of the sleep mode regardless of the application, thus increasing the battery lifetime drastically.

Another direction of the future work includes developing high-end applications on the edge. Having a reconfigurable and versatile battery-powered device with a dynamic approach on repurposable elements can reduce the need for specialized hardware in high-speed signal processing applications such as vibration [39], ECG signal [3], magnetic signal [12] analysis or multimedia applications including audio encoding/decoding or video streaming [41, 43, 44] in smart environments. In addition, we would also like to leverage existing literature on data processing in sensor networks and use the platform to perform heavy processing tasks in a distributed environment (such as cooperative sensor networks) to reduce the need for centralized servers. Utilizing the platform in the wireless acoustic sensor networks could be a promising application to perform surveillance based on sounds captured from the environment. Thoen et.al [53] suggest that with specialized hardware, acoustic surveillance applications can be performed in a decentralized environment by battery powered devices. We believe that Chimera is a suitable platform for these applications and will reduce the need for specialized hardware or/and centralized servers for high-speed data processing.

As mentioned in previous sections, we would like to overcome the challenges which prevent FPGAs from being used in IoT and sensor networks, particularly the lack of systemic support and extra complexity introduced by the digital design. One way to achieve this goal is to utilize a processor core with an open-source tool-chain and large community support. RISC-V [18] is a free and open-source reduced instruction set computer (RISC) architecture that is widely used and supported by academia and industry. Designing a RISC-V based processor core or implementing an existing one [54] onto Chimera will bring all the software and tool-chain support and help us to reduce the complexity of FPGA design for software developers.

7 CONCLUSION

This paper presents a flexible and low-power experimental platform with multi-year battery life based on a Flash FPGA designed for the low-power wireless networks domain, called Chimera. We justify the need for such platform with the rise of smart environments such as smart buildings or smart healthcare facilities, where the change in application requirements is inevitable. We identify two main reasons that prevent FPGAs from being used in low-power embedded platforms : *(i)* The high power consumption of SRAM-based FPGAs and *(ii)* the lack of systematic support of FPGAs used in contemporary IoT tool-chains. We address these problems by designing an open-source platform based on a Flash FPGA capable of over-the-air reconfiguration of both software and hardware resources while still achieving multi-year battery life. We detailed and analysed the hardware and software architecture of Chimera platform and presented a physical product as well as various contemporary application case-studies with different requirements to illustrate the limits of Chimera.

We evaluated the platform by focusing on three different case-study and presented the power consumption and reconfiguration characteristics, an application-level analysis, and a state-of-the-art comparison against existing solutions. The results show that: *(i)* Chimera can achieve multi-year battery-life time with four standard AA batteries, depending on the FPGA design and application requirements, *(ii)* using an FPGA to perform computational heavy tasks is more efficient in terms of power consumption and performance, and *(iii)* it directly affects the performance of sequential tasks by increasing the throughput 10 times compared to software implementations. We discussed the applicability and the future work of the platform and showed that it is possible to utilize an FPGA-based, deeply reconfigurable design for a versatile platform in resource-constrained environments. With Chimera, we intend to bridge the gap between low-power static end-devices and high-end FPGAs while providing flexible and adaptable hardware/software architecture as close as possible to the edge of the network.

REFERENCES

[1] M. Ammar, M. Washha, G. S. Ramabhadran, and B. Crispo. slimiot: Scalable lightweight attestation protocol for the internet of things. In *2018 IEEE Conference on Dependable and Secure Computing (DSC)*, pages 1–8. IEEE, 2018.

[2] E. Aras, S. Delbruel, F. Yang, W. Joosen, and D. Hughes. A low-power hardware platform for smart environment as a call for more flexibility and re-usability. In *EWSN*, pages 194–205, 2019.

[3] D. Azariadi, V. Tsoutsouras, S. Xydis, and D. Soudris. Ecg signal analysis and arrhythmia detection on iot wearable medical devices. In *2016 5th International conference on modern circuits and systems technologies (MOCAST)*, pages 1–4. IEEE, 2016.

[4] O. Berder and O. Sentieys. Powwow: Power optimized hardware/software framework for wireless motes. In *Architecture of Computing Systems (ARCS), 2010 23rd International Conference on*, pages 1–5. VDE, 2010.

[5] S. Biookaghazadeh, M. Zhao, and F. Ren. Are fpgas suitable for edge computing? In *USENIX Workshop on Hot Topics in Edge Computing (HotEdge 18)*, Boston, MA, 2018. USENIX Association.

[6] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli. Fog computing and its role in the internet of things. In *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pages 13–16. ACM, 2012.

[7] G. C. Cardarilli, L. Di Nunzio, R. Fazzolari, M. Re, F. Silvestri, and S. Spanò. Energy consumption saving in embedded microprocessors using hardware accelerators. *Telkomnika*, 16(3):1019–1026, 2018.

[8] F. Conti, R. Schilling, P. D. Schiavone, A. Pullini, D. Rossi, F. K. Gürkaynak, M. Muehlberghuber, M. Gautschi, I. Loi, G. Haugou, et al. An iot endpoint system-on-chip for secure and energy-efficient near-sensor analytics. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 64(9):2481–2494, 2017.

[9] U. Controller. Atmega16u4/atmega32u4.

[10] A. Corporation. Atmega328/p datasheet, 8-bit avr microcontroller, 2016.

[11] M. Cummings and S. Haruyama. Fpga in the software radio. *IEEE communications Magazine*, 37(2):108–112, 1999.

[12] J. Ding, S.-Y. Cheung, C.-W. Tan, and P. Varaiya. Signal processing of sensor node data for vehicle detection. In *Proceedings. The 7th International IEEE Conference on Intelligent Transportation Systems (IEEE Cat. No. 04TH8749)*, pages 70–75. IEEE, 2004.

[13] C. Doukas and I. Maglogiannis. Bringing iot and cloud computing towards pervasive healthcare. In *2012 Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, pages 922–926. IEEE, 2012.

[14] M. J. Dworkin. Sha-3 standard: Permutation-based hash and extendable-output functions. Technical report, 2015.

[15] A. Engel and A. Koch. Heterogeneous wireless sensor nodes that target the internet of things. *IEEE Micro*, 36(6):8–15, 2016.

[16] S. A. Fahmy, K. Vipin, and S. Shreejith. Virtualized fpga accelerators for efficient cloud computing. In *2015 IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom)*, pages 430–435. IEEE, 2015.

[17] A. Fatecha, J. Guevara, and E. Vargas. Reconfigurable architecture for smart sensor node based on ieee 1451 standard. In *SENSORS, 2013 IEEE*, pages 1–4. IEEE, 2013.

[18] R.-V. Foundation. Risc-v: The free and open risc instruction set architecture. Online, 2019.

[19] T. Gomes, S. Pinto, A. Tavares, and J. Cabral. Towards an fpga-based edge device for the internet of things. In *Emerging Technologies & Factory Automation (ETFA), 2015 IEEE 20th Conference on*, pages 1–4. IEEE, 2015.

[20] Google. Nest thermostat software update history. Online, 2019.

[21] J. Greene, S. Kaptanoglu, W. Feng, V. Hecht, J. Landry, F. Li, A. Krouglyanskiy, M. Morosan, and V. Pevzner. A 65nm flash-based fpga fabric optimized for low cost and power. In *Proceedings of the 19th ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, FPGA '11, pages 87–96, New York, NY, USA, 2011. ACM.

[22] H. Hsing. Sha3 (keccak) core. Online.
[23] T. Instruments. Msp430 microcontroller datasheet, march 2010, 2010.
[24] Intel. Intel's fog reference design overview. Online, 2017.
[25] X. Jiang, J. Polastre, and D. Culler. Perpetual environmentally powered sensor networks. In *Proceedings of the 4th international symposium on Information processing in sensor networks*, page 65. IEEE Press, 2005.
[26] R. Kirchgessner, A. D. George, and G. Stitt. Low-overhead fpga middleware for application portability and productivity. *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, 8(4):21, 2015.
[27] P. Kocher, D. Genkin, D. Gruss, W. Haas, M. Hamburg, M. Lipp, S. Mangard, T. Prescher, M. Schwarz, and Y. Yarom. Spectre attacks: Exploiting speculative execution. *arXiv preprint arXiv:1801.01203*, 2018.
[28] Y. E. Krasteva, J. Portilla, J. M. Carnicer, E. De La Torre, and T. Riesgo. Remote hw-sw reconfigurable wireless sensor nodes. In *Industrial Electronics, 2008. IECON 2008. 34th Annual Conference of IEEE*, pages 2483–2488. IEEE, 2008.
[29] Y.-S. Kuo, P. Pannuto, T. Schmid, and P. Dutta. Reconfiguring the software radio to improve power, price, and portability. In *Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems*, pages 267–280. ACM, 2012.
[30] R. Langner. Stuxnet: Dissecting a cyberwarfare weapon. *IEEE Security & Privacy*, 9(3):49–51, 2011.
[31] D. Lymberopoulos, N. B. Priyantha, and F. Zhao. mplatform: a reconfigurable architecture and efficient data sharing mechanism for modular sensor nodes. In *Proceedings of the 6th international conference on Information processing in sensor networks*, pages 128–137. ACM, 2007.
[32] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson. Wireless sensor networks for habitat monitoring. In *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, pages 88–97. Acm, 2002.
[33] Memsic. Micaz wireless measurement system. Online.
[34] Microsemi. Igloo nano low power flash fpgas, 2015.
[35] S. Nolting. The neo430 processor. Online, February 2018.
[36] J. Noorman, P. Agten, W. Daniels, R. Strackx, A. Van Herrewege, C. Huygens, B. Preneel, I. Verbauwhede, and F. Piessens. Sancus: Low-cost trustworthy extensible networked devices with a zero-software trusted computing base. In *USENIX Security Symposium*, pages 479–494, 2013.
[37] Oracle. Sun spot. Online.
[38] M. Orehek and A. Zugenmaier. Updates in iot are more than just one iota. In *Internet of Things Software Update Workshop (IoTSU)*, 2016.
[39] S. N. Pakzad and G. L. Fenves. Statistical analysis of vibration modes of a suspension bridge using spatially dense wireless sensor network. *Journal of structural engineering*, 135(7):863–872, 2009.
[40] M. D. V. Pena, J. J. Rodriguez-Andina, and M. Manic. The internet of things: The role of reconfigurable platforms. *IEEE Industrial Electronics Magazine*, 11(3):6–19, 2017.
[41] C. Plessl, R. Enzler, H. Walder, J. Beutel, M. Platzner, and L. Thiele. Reconfigurable hardware in wearable computing nodes. In *Proceedings. Sixth International Symposium on Wearable Computers,*, pages 215–222. IEEE, 2002.
[42] J. Polastre, R. Szewczyk, and D. Culler. Telos: enabling ultra-low power wireless research. In *Proceedings of the 4th international symposium on Information processing in sensor networks*, page 48. IEEE Press, 2005.
[43] S. Pudlewski, A. Prasanna, and T. Melodia. Compressed-sensing-enabled video streaming for wireless multimedia sensor networks. *IEEE Transactions on Mobile Computing*, (6):1060–1072, 2012.
[44] R. Puri, A. Majumdar, P. Ishwar, and K. Ramchandran. Distributed video coding in wireless sensor networks. *IEEE Signal Processing Magazine*, 23(4):94–106, 2006.
[45] A. Putnam, A. M. Caulfield, E. S. Chung, D. Chiou, K. Constantinides, J. Demme, H. Esmaeilzadeh, J. Fowers, G. P. Gopal, J. Gray, et al. A reconfigurable fabric for accelerating large-scale datacenter services. *ACM SIGARCH Computer Architecture News*, 42(3):13–24, 2014.
[46] M. Rahimi, R. Baer, O. I. Iroezi, J. C. Garcia, J. Warrior, D. Estrin, and M. Srivastava. Cyclops: in situ image sensing and interpretation in wireless sensor networks. In *Proceedings of the 3rd international conference on Embedded networked sensor systems*, pages 192–204. ACM, 2005.
[47] M. Rao, T. Newe, and I. Grout. Secure hash algorithm-3 (sha-3) implementation on xilinx fpgas, suitable for iot applications. In *8th International Conference on Sensing Technology (ICST 2014), Liverpool John Moores University, Liverpool, United Kingdom, 2nd-4th September*, 2014.
[48] T. Rinta-Aho, M. Karlstedt, and M. P. Desai. The click2netfpga toolchain. In *USENIX Annual Technical Conference*, pages 77–88, 2012.
[49] K. Shahzad, P. Cheng, and B. Oelmann. Sentiof: an fpga based high-performance and low-power wireless embedded platform. In *Computer Science and Information Systems (FedCSIS), 2013 Federated Conference on*, pages 901–906. IEEE, 2013.
[50] S. Szilvási, B. Babják, P. Völgyesi, and A. Lédeczi. Marmote sdr: Experimental platform for low-power wireless protocol stack research. *Journal of Sensor and Actuator Networks*, 2(3):631–652, 2013.

[51] Y. K. Tan and H. D. Chinh. *Smart wireless sensor networks*. BoD–Books on Demand, 2010.

[52] S. Tanaka, N. Fujita, Y. Yanagisawa, T. Terada, and M. Tsukamoto. Reconfigurable hardware architecture for saving power consumption on a sensor node. In *Intelligent Sensors, Sensor Networks and Information Processing, 2008. ISSNIP 2008. International Conference on*, pages 405–410. IEEE, 2008.

[53] B. Thoen, G. Ottoy, F. Rosas, S. Lauwereins, S. Rajendran, L. De Strycker, S. Pollin, and M. Verhelst. Saving energy in wsns for acoustic surveillance applications while maintaining qos. In *Sensors Applications Symposium (SAS), 2017 IEEE*, pages 1–6. IEEE, 2017.

[54] A. Traber, F. Zaruba, S. Stucki, A. Pullini, G. Haugou, E. Flamand, F. K. Gurkaynak, and L. Benini. Pulpino: A small single-core risc-v soc. In *3rd RISCV Workshop*, 2016.

[55] N. Tsiftes, A. Dunkels, and T. Voigt. Efficient sensor network reprogramming through compression of executable modules. In *2008 5th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks*, pages 359–367. IEEE, 2008.

[56] M. A. M. Vieira, C. N. Coelho, D. Da Silva, and J. M. da Mata. Survey on wireless sensor network devices. In *Emerging Technologies and Factory Automation, 2003. Proceedings. ETFA'03. IEEE Conference*, volume 1, pages 537–544. IEEE, 2003.

[57] M. Yun and B. Yuxin. Research on the architecture and key technology of internet of things (iot) applied on smart grid. In *2010 International Conference on Advances in Energy Engineering*, pages 69–72. IEEE, 2010.

[58] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi. Internet of things for smart cities. *IEEE Internet of Things journal*, 1(1):22–32, 2014.

[59] Y. Zhang, C.-H. Feng, I. Demirkol, and W. B. Heinzelman. Energy-efficient duty cycle assignment for receiver-based convergecast in wireless sensor networks. In *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*, pages 1–5. IEEE, 2010.