A Low-Power Hardware Platform for Smart Environment as a Call for More Flexibility and Re-Usability

Stéphane Delbruel, Emekcan Aras, Wouter Joosen, Danny Hughes

A Low-Power Hardware Platform for Smart Environment as a Call for More Flexibility and Re-Usability

Emekcan Aras, Stéphane Delbruel, Fan Yang, Wouter Joosen and Danny Hughes
imec-DistriNet, KU Leuven
3001 Leuven, Belgium
firstname.lastname@cs.kuleuven.be

Abstract

This paper presents Chimera, a low-power platform for research and experimentation with reconfigurable hardware for end-devices in the Internet of Things. Through an architecture based on a Flash FPGA, Chimera aims to offer flexibility in the usage via live over-the-air hardware and software reconfiguration. This adaptability enables low-cost in-situ deployment of upgrades without the need for physical access to devices as well as the capability to absorb a temporary heavy calculations in response to events. Chimera offers a step forward in pushing computation to the edge, by offering a modular hardware embedded platform that is able to switch instantaneously between largely passive sensor tasks and computing intensive operations while still retaining multi-year battery lifetimes. This paper reviews the factors which have thus far prevented Field Programmable Gate Arrays (FPGAs) from being used as the primary processors in IoT end-devices and demonstrates through evaluation that the Chimera hardware and software platform work together to overcome these barriers.

Categories and Subject Descriptors

C.3 [**Special-Purpose and Application-Based Systems**]: Real-time and embedded systems

General Terms

Performance, Design, Measurement, Experimentation

Keywords

Reconfigurable Sensor Node, Sensor Network, FPGA

1 Introduction

The Internet of Things (IoT) is being deployed in an ever growing range of applications; from industrial monitoring, through smart buildings to wearable devices. A strategic domain in this deployment comprises low-cost battery powered wireless sensors meant to enrich the understanding and control of a given environment. Generic IoT platforms such as Spark.io, Electric Imp, Libellium and VersaSense enable the rapid roll-out of IoT applications in this domain, without the need to develop specialized hardware[16]. This deploy-and-forget approach in populating an environment with multi-years battery life sensors is appreciated thanks to its low cost and convenience. At least until an hardware or software security update must be applied to a remotely deployed platform, or a reworking of tasks is temporary needed on a given sensors platform. The need for repurposable end-devices is constantly aggravated by the extension of the cloud computing paradigm to the edge of the network. However Micro Controller Units (MCUs) used in these platforms are often incapable of supporting all but the simplest in-situ processing power intensive analysis, which prevents them from being used to develop IoT applications including high speed signal processing or multimedia. The need for energy efficiency on these resource constrained devices have also prevented them to offer on-the-fly software updates. In short, these platforms do only one thing but do it efficiently by being tailored for their task.

Different leads have been investigated to extend the flexibility and the ability to scale up to task for these platforms. Among the promising ones, one involving generic platforms and a second turned towards FPGAs stood up. While flagship 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. For the lead standing on FPGA-based platforms, allowing for hardware and software updates, previous research and products [2, 3, 10] have been developed in that way to extend the flexibility of an infrastructure far away for its core components. Alas, their role stop at the gateway level and their cost and power consumption prevent them to be spread deeper in the infrastructure. Chimera attempt to narrow the gap by providing a deeply re-configurable FPGA-based hardware platform that is capable of both high-speed signal analysis and multi-year battery lifetime.

Extensive prior work on the application of FPGAs in embedded applications has shown their potential as: multimedia co-processors [26], cryptographic accelerators [20] and flexible network accelerators [29]. As of today, this work has yet to break through into main-stream IoT platforms. We identify two key reasons for this adoption delay: *(i.)* the high

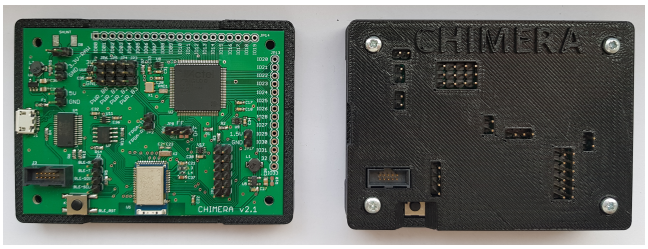

Figure 1. Chimera Platform.

power consumption of most FPGA-based platforms and *(ii.)* the lack of systematic support for FPGA use in contemporary IoT tool-chains. Chimera addresses this problem by providing a fully open-source FPGA-based platform that is capable of achieving multi-year battery life on standard AA batteries, allows over-the air reconfiguration of both software and hardware resources and provides a tool-chain that can optimize and reconfigure the FPGA cores transparently based upon the demands of application software.

We demonstrate that Chimera achieves its goals of adaptability and flexibility without sacrificing long lifetime through the implementation and evaluation of a number of archetypal IoT applications. We then demonstrate the capability of Chimera to tailor FPGA hardware dynamically based upon the changing needs of application software. Our results show that: *(i.)* Chimera achieves more than one year of battery lifetime using a couple of pairs of retail store AA/LR6 batteries, *(ii.)* optimizing the FPGA based upon the needs of its application software can increase battery life by up to 30% depending on the application and *(iii.)* the costs of online or offline reconfiguration are acceptable (worst case 0.002% of a couple of pairs of AA battery life) even in the absence of partial reconfiguration support.

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

2 Background

Programmable logic devices has long been an incentive in the embedded networked sensor devices field. The ability to redefine the hardware functionalities of an already deployed platform offers the promise of a tailored architecture for each task, allowing much more agility and efficiency in a domain where resources are constrained. Among the various family of programmable logic devices, Complex Programmable Logic Device (CPLD) and Field-Programmable Gate Array (FPGA) are the two dominant branches, with the major difference being for the FPGAs to be built upon lookup tables instead of sum of products. The configuration or programming of an FPGA mostly starts with a digital design of required functionality by using a hardware description language (HDL) like VHDL or Verilog. Various functionalities from basic digital XOR gates to complete processor or micro-controller designs can be implemented [4]. 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 to be suited for constrained networked sensor devices among them the ability to describe these devices using Hardware Description Languages, and their power consumption [34].

One of these problems encountered by the S-RAM based FPGAs is the storage of the design. Contrary to the CPLDs which embed non-volatile memory in the chips, the look-up tables are a form of memory but a non-persistent one when powering down the device. Requiring to power an external memory to store the design of an FPGA and load it at every change of configuration or reboot was a limiting factor which as of today is progressively vanishing with the rise of Flash-based FPGAs. As a reminder, the idea of using flash memory to externally store the design of an FPGA was already in use, but the added costs in consumption, security and integration were high and the attempts to include a non-volatile memory in an FPGA were unconvincing for highly constrained domains. However, recent advances in flash memory and integration allowed Flash-based FPGAs to become more and more relevant. By leveraging Flash properties, design of FPGAs can now offer competitive advantages compared to S-RAM based ones, allowing the upcoming of low cost and low power flash-based FPGAs [8].

Competitive advantages offered by this design of FPGAs are still challenged by traditional preference over simple and efficient MCUs-based design issued by the embedded hardware industry. These designs have a long-lasting record of efficiency in low-power and constrained environment, making them the actual best candidates for being cost-effectively deployed and efficiently perform a simple task for a decade without intervention in an industrial environment [33, 21]. Some domains however tend to be closer to humans than others and as such are more subject to influence and change of habits by them. Repurposing a storage room into a meeting room in an office building is a common and frequent change. Discoveries of security flaws with ties to hardware has been increasingly present in the news over the past year. Implementing an AES core for privacy in the first case, or adding a precise RTC for wireless channel hoping synchronisation in the second are costly operations requiring manual intervention or even end-device replacement among the low-cost wireless MCU-based sensor space, where devices tend to be static and tailored for a particular task. On the scale of a smart hospital or the reorganization of a university these costs could sky-rocket. This is where end-devices based on platforms like Chimera are relevant.

In this context, these devices can now takes place in the embedded sensor systems environment, allowing platforms to be much more flexible yet power efficient [5], thus being able to perform computing-power intensive applications for a given amount of time before being remodelled in power saving devices, collaborative data processing between platforms or being remodelled in a different system at each life cycle.

Table 1. Comparison of platforms with reconfigurable hardware

Platform	FPGA as main processor	Programmable device	Radio module	Dynamic reconf.	Main features	Active power	Sleep power
Cyclops[26]	no MICA2	CPLD	CC2420	no	imaging applications	64.8mW	0.7mW
mplatform[14]	no msp430	CPLD	CC2420	no	real-time applications	5mW-13mW	-
PowWoW[1]	no MSP430	FPGA Igloo Nano	CC2420	no	hardware accelerator	1-30mW	16uW
Cookie[12]	no ADuC841	FPGA Xilinx Spartan3	Zigbee	yes	sensor management	150-240mW	60mW
Sentiof[29]	no Atmel AT32U	FPGA Xilinx Spartan6	CC2520	yes	high-end application	158mW	0.313mW
Marmote[30]	no Smart Fusion	FPGA fabric in SOC	2.4Ghz RF frontend	no	used as SDR	287-852mW	71mW
uSDR[13]	no Smart Fusion	FPGA fabric in SOC	2.4Ghz RF frontend	no	used as SDR	1400 mW	71 mW
IEEE1451 based node[6]	no Telos B[24]	PSOC	CC2420	yes	analog accelerator	62.7mW + telosB power	6.6uW + telosB power

3 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[26] 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 [15] 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 device 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 and processing power-hungry tasks. Another work that utilizes a CPLD is called mplatform[14]. The authors designed a modular 4-module stack platform that is capable of processing real-time data while supporting dynamic reconfiguration. A similar platform making use of reconfigurable hardware is Sentiof[29], 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 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[12]. 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 [1]. 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 and industrial actors [23]. A FPSOC combines a FPGA fabric with several high-performance embedded processors. Gomes et al. [7] 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 [13]. 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 a 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. [31] makes use of FPSOC to prototype a similar platform like Chimera. They

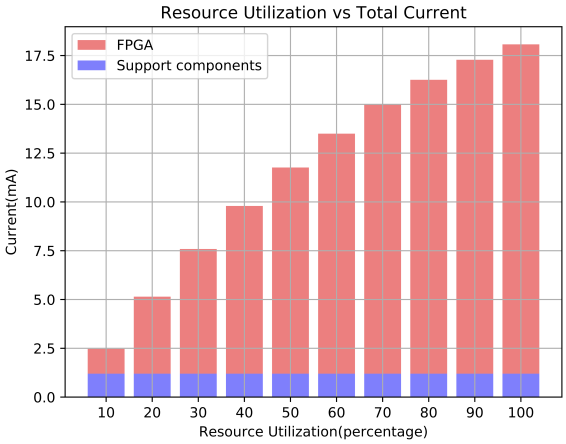

Figure 2. Influence of resource utilization on power consumption

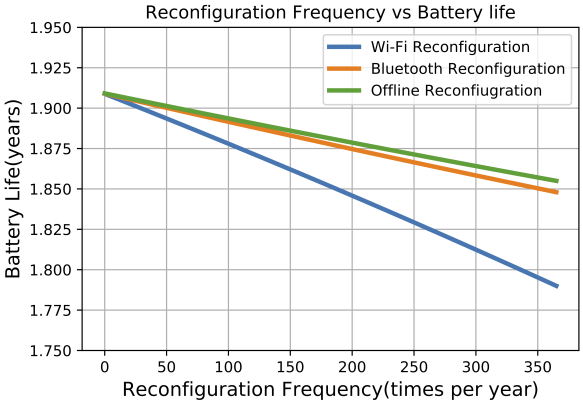

Figure 3. Effect of Reconfiguration Frequency over battery life.

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. [11]. They presented a middleware called RCMW that improves and enables application and tool portability. Another proposition was a complete toolchain called Click2NetFPGA [27], 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 4. As discussed previously, most of them utilize either a 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 having a too high power consumption.

4 Architecture

The design of the Chimera platform takes place in the wireless embedded sensor network domain and as such is oriented towards three goals. We want to propose a platform that can have its hardware and software functionalities reconfigured both locally and over-the-air at a minimal power cost. We want to design a power efficient device with minimal active and sleep power consumption, able to last multiple years on a couple of standard retail batteries. We want our platform to be easy to use with an open and accessible toolchain, in order to facilitate the work for actors more familiar with other domains. We present in the following section a more in-depth analysis of our goals as such as their implications on hardware and software design.

4.1 Design Analysis

As mentioned before, our primary goal is to offer low-cost dynamic hardware and software reconfiguration on our platform. Using a FPGA as main computing unit is the logical choice to be able to perform this task, especially flash-based ones for their cost and performances. After analysing the market and comparing various products, our choice has been set on an IGLOO Nano FPGA from Microsemi [18]. The wide range of capacities of that FPGA had yet to be assessed in terms of power consumption to determine its viability in performing our goals. In our design process, we developed a test version of Chimera in order to assess the impact of resources use in the FPGA. We created a heavy computation task and we extended gradually the percentage of resource array used in the FPGA. The measured consumption results are presented in Figure 2. The various components necessary to operate the system and their associated current consumption is represented for reference. These results display a modest power consumption and its logarithm-shaped evolution under the increase of resource usage thus confirmed us the choice of this particular FPGA as core unit of our architecture.

We then analysed our needs in order to match the frame in which reconfiguration should occur. The support of online reconfiguration should take place over the radio module used for sensor data communications. Moreover, the ability

Table 2. 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

for the device to embed and provision reconfiguration images for previously planned reaction to a particular event is a very interesting feature. In order to support this offline reconfiguration process and thus saving on radio transmission costs when switching between predefined configuration, our platform must include an onboard memory module dedicated to store FPGA images. We again used our dummy platform to perform current consumption of the reconfiguration process and analyse the choice of communication modules. This process requests then download an FPGA image from a connected gateway, store it locally, then performs an offline reconfiguration, meaning transferring the image stored in the external Flash to the programming array of the FPGA. The comparison between various candidates modules are presented in Table 2 and indicates clear preference for a Bluetooth based approach in our design. In Figure 3, we observe these effects on the lasting battery life of our dummy platform, as well as the influence of the evolution of reconfiguration frequency over time.

We then focussed on practical considerations in terms of handling the platform and its ability to be flexible in its usage, especially for the application development part. Programming on a MCU is considered easy and very common in the application domain that we target. Emulating one in a FPGA is well understood as a significant loss of performance, but as Chimera aims at being also an experimental platform, giving the ability to actors to do so in a controlled environment was an interesting feature. Having the capability to transpose a familiar and comfortable development environment is a must for any actor who is willing to discover and apprehend a new platform. Our aim was to analyse the conditions of implementing a softcore microprocessor in our FPGA, especially an iconic one in the embedded world, the Texas Instruments MSP430. For openness and customization reasons we chose the NEO430 [19] project, based on the MSP430 ISA and providing 100% compatibility with the original instruction set. This project is open, highly customizable and have a reduced footprint. We implemented it with an UART, a SPI module, one hardware Timer, a GPIO module supporting 16 I/Os, 4kB of RAM and 4kB of ROM as well as an internal bootloader (2kB ROM) with a serial interface. Having an experimental platform that can offer this facility is in our opinion a good incentive for academics and industrial actors, thus comforting us in our choice of design hardware.

4.2 Hardware Design

After having analysed how to reach our design goals and expressed the related needs in the previous subsection, we designed a final hardware version of Chimera where the power consumption has been optimized to achieve the above mentioned objectives with a minimal cost. We designed the platform around three main blocks, the radio transceiver, the core FPGA, the embedded additional memory. These blocks and their interactions are illustrated in Figure 4.

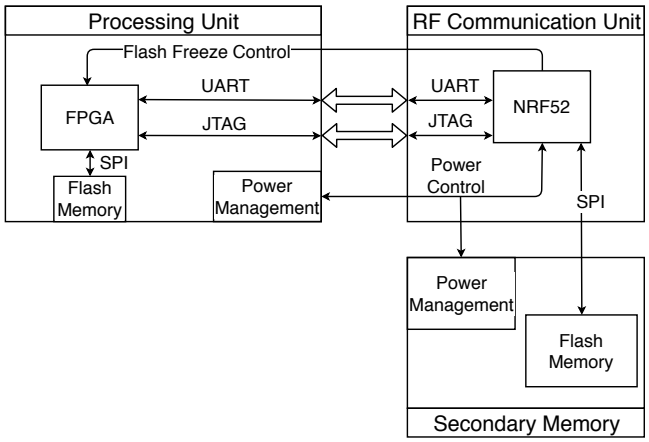

Figure 4. Hardware architecture of the Chimera platform.

The first main block, labelled RF Communication Unit, revolves around a Nordic Semiconductor SoC nRF52832 transceiver supporting Bluetooth Low-Energy, ANT and 2.4GHz proprietary communication protocols. In charge of the radio interface with the outside world, this block in his daily usage interfaces the communication of sensors values with the rest of the infrastructure and is also responsible for receiving over-the-air the communicated reconfiguration images for the FPGA. Interacting with the memory during this process via a Serial Peripheral Interface (SPI), this module stores in it the newly received image, or in response to an external order reads the memory to retrieve the designated configuration image that has to be loaded in our flash-based FPGA. This module further allows for a direct reconfiguration of the FPGA, without storing it beforehand in the onboard memory, and funnel it directly into the FPGA's internal programming array. The interface used in this process between the SoC and the FPGA is a JTAG port. Besides the previous mentioned tasks this module and its computing unit in its core are responsible of the duty-cycle and the power management of the other two units. As represented in Figure 4, the RF communication Unit is responsible to powering up and down the other units, and as such act as the sleep

mode timer of the entire platform. Such choice has been guided by the ability of the SoC to have a reduced power consumption inferior at 2μA when operating in minimal mode. Having a timing element able to implement custom firmware for smarter actions during the awakening or sleeping phase while draining so little power makes him the perfect candidate to deport the power management to.

The second block consists of the Main Processing Unit. As discussed previously, recent advances in Flash technology have turned the tables on FPGA's architectures, allowing the Flash-based ones to become competitive both in cost and power consumption in the low-power domain against more traditional platforms such as the SRAM-based ones. Our main processing unit is centred around an Igloo Nano FPGA from MicroSemi. It is one of the lowest power FPGAs in the market with a power consumption of 2.2μW, 16μW and an interval of 1-30mW respectively for sleep mode, flash freeze mode and run mode depending on the routed design complexity. As our main processing unit has been designed to support reassignments in tasks and their varying complexity, we adjunct the computing unit a dedicated external memory. This memory is a non-volatile flash-based one with a capacity of 2MB. It aims providing support for information storage as it be in case of a loss of connection between the rest of the infrastructure, or a reassignment to perform signal processing operations. In some extreme cases such as implementing convolutional neural networks on embedded FPGA platforms [25], having an external memory to store and process information is mandatory.

The third block is designated as Secondary Memory Unit. The role of this block is fully tied to the reconfiguration process. It is mainly composed of an external flash-based memory, solely in charge of storing the reconfiguration images for the Main Processing Unit. This low-power unit is managed by the RF Communication Unit for awake and sleep cycles as well as for any read or write access to the memory. As of today, the memory has a capacity of 2MB, and is able to store up to 4 different reconfiguration images. Its power characteristics of 2.5-20mA, 2-10mA, and 5μA respectively for a reading cycle, writing cycle and sleep mode with a clock speed varying between 20MHz and 120MHz allows Chimera to economises on the reconfiguration in avoiding to request a wireless transmission every time an image change is requested. Moreover, when no reconfiguration process is initiated, this memory is powered off by the RF Communication Unit, preventing the undue power consumption of 5μA from the sleep mode and only powering the power management unit for a nominal consumption of 3nA.

4.3 Software Design

Chimera is a reconfigurable hardware platform combined with a software architecture designed to provide a complete IoT solution for both high-end and low-level applications. The hardware platform is build on a radio module, a low-power Flash-based FPGA used as the main processing unit and a radio transceiver supporting multiple protocols. Instead of using separate modules to create a complete platform, Chimera integrates programming, computation and communication onto a single board. It introduces new features to improve the versatility and the energy efficiency

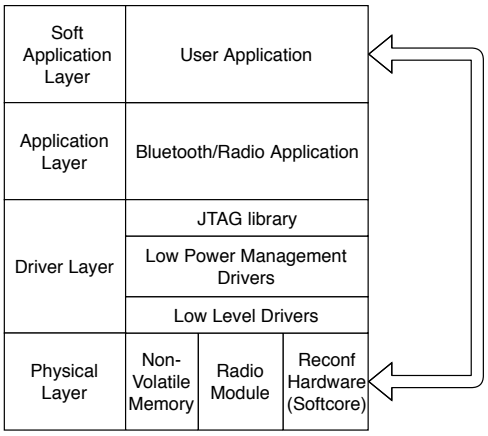

Figure 5. Versatile Software Stack of the Chimera platform.

while delivering the abstraction features of FPGAs for software developers.

4.3.1 Software Stack

The versatile architecture of the platform can be seen in the Figure 5 and consists of four layers: hardware, drivers, application and soft-application layer. The physical layer is formed by the three previously detailed hardware units so the driver level contains required low-level drivers to interface properly with these elements. The bluetooth/radio application layer contains the bluetooth application to connect bluetooth gateway and programmer application to program and reconfigure the FPGA when it is needed. As mentioned previously, a soft processor core is implemented into the FPGA to be used as a main processor and when doing so it adds another extra layer on top of the common software stack. This top layer called the user application layer defines the application implemented in the soft processor core. The advantage provided by the flexibility of this platform is to allow any layer to be entirely customizable. Removing an hardware component in the FPGA will influence not only the power-consumption but also the software stack. This great adaptability allows Chimera's users to bare-strip the platform to the minimum in order to implement a large FFT module or an AES core, but also to fine tune the power management drivers, adapt the number of I/Os actually handled or the interactions between hardware components.

4.3.2 Toolchain and Environment

In order to run the platform properly and efficiently we need to develop and program both hardware and software from bottom to top. To assist in coping with the complexity of that task we used a set of tool-chains and development platforms. Digital designs for a given FPGA are developed and turned into a bit stream by using a development environment, often provided by the FPGA vendor, in our case Microsemi provides one called Libero SoC Design Software. Low level drivers and MCU applications are developed by using integrated development environment that can be obtained from Atmel's website free of charge. For the soft application layer we used the free TI msp430-gcc compiler tool chain since the implemented soft-core is 100% compatible

with the original instruction set of the MSP430. Having a platform with a simple tool chain with enough abstraction to allow academics students to develop in an Arduino-like environment some simple tinkering exercises, while in the same time allowing industrial actors to deploy a multiple hardware-configurations platform with tailored power consumption profiles is to us one of the keys to ease low-power FPGAs in the low-power domain.

4.3.3 Reconfiguration Process

One of our main purpose is to develop a platform 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 able to perform both remote (online) reconfiguration of the FPGA as well as offline one. To that extend, 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 type 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. This bitstream can be obtained from two different manners, depending on the operated type of reconfiguration.

In Online mode, the MCU receive through its radio channel the given bitstream, and use its SPI interface to forward and store it in the dedicated flash-memory for the storage of images. When the image transfer is finished and stored in memory, the RF module is then 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 is not destined to be stored locally and simply implemented in the FPGA as soon as possible, the MCU has the ability to skip the write and read sequence on the memory and directly reprogram via JTAG the FPGA with the bitstream in the course of its reception by the radio module. 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 have been rewritten by another reconfiguration cycle.

In Offline mode, the MCU will select the desired image in memory and will then read it bit by bit and forwards them along to the FPGA. The dedicated memory can hold up to four different images, allowing the platform to adapt autonomously to a non-scheduled but planned event, without needing to request a dedicated reconfiguration image from a distant server. This mode allows us to perform faster reconfiguration at a cheaper energy cost than in the Online mode due to the inactivity of the radio module. At last, to understand that the extend of the reconfiguration process goes way higher than just low-level drivers, lets remember that 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 have an optional bootloader which is also part of that image, we can by extend reconfigure in Online or Offline mode the soft-application layer, thus performing remote dynamic update on the code run by the emulated MCU in the FPGA Programming Array.

4.3.4 Images for IoT Applications

To illustrate the capacities of Chimera we developed three different images suited for archetypal IoT applications. These images are three different configurations of the soft-core processor saved into the dedicated non-volatile Flash memory.

The first one is a standard sensor application. Common sensor applications such as temperature-humidity measurements, sun-light tracking or motion detection only have two tasks; reading the sensor data and sending it to a gateway via wireless link to be relayed to an application server. These applications do not need to process data on the end-device thus allowing the end-device to not have additional hardware modules such as communication interfaces for extra hardware module, secondary timer module or requiring external memory to process the information. Therefore, the configuration was made according to applications needs to optimize the resource utilization for low power consumption.

The second one is a remote control and relay application. Recent developments in IoT aimed to empower an actor by easing control and interactions with the surrounding environment in its daily life. Smart home and home automation applications are widely implemented and offers remote control for embedded systems and electronics devices such as light, heating, entertainment systems, ventilation, appliances, etc. These applications are more complex than the first passive sensor application and need more capabilities such as various communication interfaces and a larger memory. In order to support required functionality, the resource utilization in this configuration is higher than for the first one.

The third configuration is a real-time embedded OS application. Among the domain of embedded applications high-end ones such as industrial temperature control, vehicle tracking or unmanned robotic missions have real-time requirements demanding the use and reliance of real-time embedded operating systems. Use of an embedded operating system brings a lot of advantages in terms of reliability and portability, and since the complexities in embedded applications are increasing various research and products tends to have a glimpse as these solutions [9, 35]. Therefore, we implemented FreeRTOS on top of the Chimera platform to demonstrate such applications can be developed using this platform. In order to implement FreeRTOS, extra timers and watchdog-timers have been added. The resource utilization of our main processing unit in this case is close to 100% and the power consumption reflects it by being superior to the consumption of the two previous configurations (active power consumption of the first image: 14mA, the second image: 17mA, the third image: 22mA).

The versatility of the needs in resources of these three different configurations implemented successfully, each one reflecting a common situation in the IoT end-devices domain, represents the ability of the Chimera platform to adapt and perform complex tasks while being already employing a large amount of its own resources to emulate an MCU architecture. Getting rid of that handicap, voluntarily introduced

to give a glimpse of the performances of our platform, help to understand the flexibility and the possibilities embodied by Chimera.

4.3.5 Energy Management

To achieve multi-year battery life in the resource-constrained devices domain targeted by our platform, one cannot rely solely on the hardware design or the software design. It is only by a fine tuning of both, and a seamless application of low-power policies through both that costless multi-year battery life can be achieved. The agility of our platform combined with its fully customizable hardware-software interaction allows us to compete along more static tailored and dedicated hardware for that goal. In order to reach this goal, 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 low power management drivers where all major components including the RF Communication Unit, the Main Processing Unit and the Secondary Memory Unit can be switched off, on or switched to low-power modes at run time. In addition, Chimera bases its activity pattern on low duty cycle principle. Duty cycling, between a period of activity and a period of deep sleep, is a common concept and widely spread in wireless sensor network applications to reduce the overall power consumption[36]. The sensor nodes constantly switch states between active mode and sleep mode, the active mode being the activity mode where the sensor collects information, process it, store or send it to the rest of the infrastructure, and the sleep mode being a period of activity where all the indispensable elements are reduced to the lowest power consumption by disabling unused features, and the rest of the components is turned off or put asleep. The ratio between the active time and the total time is called duty cycle[28]. Low-power wireless sensors typically operates on a low duty cycle in order to spare the battery as much as possible without compromising their ability to interact with their environment. Therefore, all the major components are in low power mode for most of the time. Once an event happens, has it be a scheduled measurement or a reaction to an unplanned event, they wake up quickly, process the event and returns to low power mode again.

5 Evaluation

Our evaluation of the Chimera platform focuses on the power consumption and reconfiguration characteristics to demonstrate its low-power characteristics. For a better comprehension, we integrate the evaluated performances in a use case scenario which comprises a series of different tasks. We later on detail the evaluation strategy and methodology then present and discuss the results.

5.1 Scenario

We consider a scenario in a smart building, where for cost and maintenance reasons only one type of platform is deployed in each room. As the usage changes, we consider adapting the deployed Chimera platforms to match the upcoming needs unplanned during the deployment phase. Sensors are deployed to perform simple passive wireless sensor applications, but the repurposing of a given room might trigger a need for platform reconfiguration, as it be for security or privacy reasons. Chimera offers a modular embedded platform able to switch between a regular sensor task and more computing intensive operations. The implication of our platform in this scenario aims at analysing the cost of these changes and assess their impact on the low-power capabilities of the platform.

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

The standard task for a wireless sensor device in IoT or WSNs is to collect environmental parameters such as temperature, humidity, light intensity and send the related data to a centralized server via a network gateway or a neighbour node for relay purposes. These sensor devices are designed to achieve multi-year battery life due to costly manual maintenance or for their deployment in remote areas, thus they have very limited resources and stick to a low-power consumption model. In our scenario, the data collected by sensors in laboratories, meeting rooms or offices during the working hours might be a sensitive information that as to remains private. An extra layer of security needs to be applied to ensure the security of the sensor data. However, running encryption algorithms such as AES all-the-time directly affects and decreases battery-life drastically, while manufacturing end-devices with additional dedicated hardware elements *"just in case"* is not an option in regards of the added costs. As an evaluation application, we implemented 3 sensor devices inside of our university building. These sensors are dedicated to environment data measurements, but on the course of a two days meeting event, we consider one of the rooms as a sensitive meeting room during working hours on a 8am-8pm basis, and a second one as a full-time sensitive office dedicated to experiments. The platform in the meeting room starts with default sensor image and re-configures itself in every 12 hours to a bigger core with AES module to perform encryption process. Every night it then re-configures itself to the default sensor image to save energy. The platform in the sensitive experiments room is reconfigured with the same large AES core at day one, and stay continuously in this configuration until 48 hours later and the end of the meeting days. The third platform is deployed in an unaffected room of the building by these events, and serves as a baseline comparison on the consequences and costs of that changes.

5.2 Methodology

We evaluate the impacts of this scenario on the three platforms based on two power consumption markers; current trends in active cycle and evolution of the battery charge over these two days of meeting. As mentioned previously, we deployed the experimental setup in our university building to

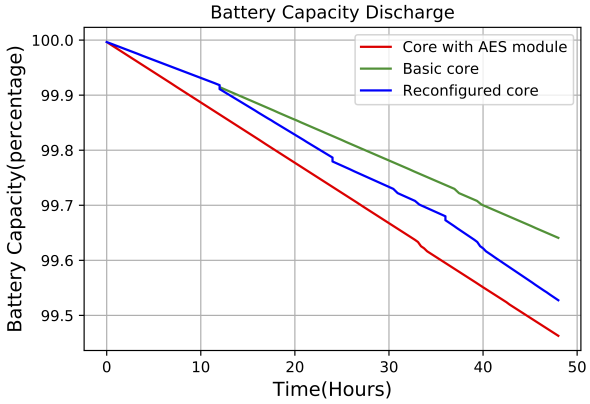

Figure 6. Battery Charge Trend for three different scenarios.

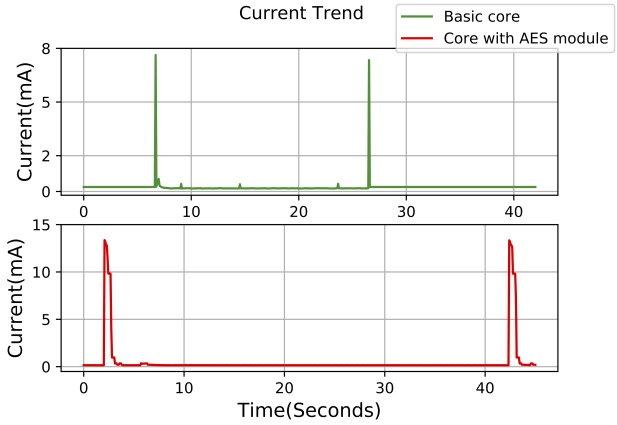

Figure 7. Current trend in a complete cycle operation.

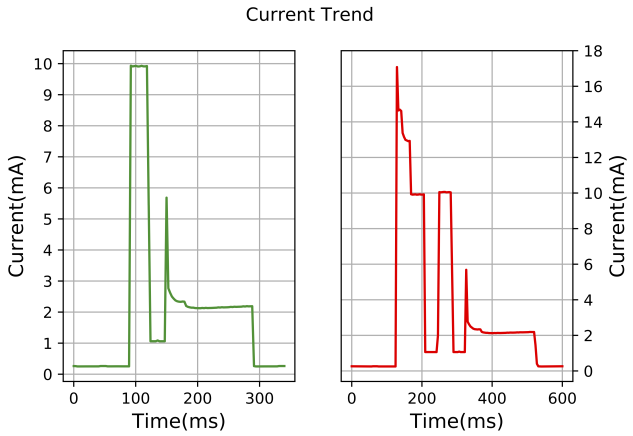

Figure 8. Current trend during an active cycle. Sensor core (green) vs Sensor core + AES core (red)

measure temperature and humidity. We tested three different configurations on the end-devices. One with the basic core sending sensor data, the second with an added AES core encrypting data and sending it, and the third one which reconfigures itself every 12 hours to switch between the two previous configurations, the AES supporting one and the non-supporting one. The details of the FPGA images corresponding to these configurations can be found in Table 3, detailing the core composing units and their associated resource utilisation. All the three platforms alternate between active cycle where measurements are done, then encrypted or not before being sent to the rest of the infrastructure via a BLE connection to a gateway, and a sleep cycle where platforms sleep in minimum energy mode. As the three platforms perform different tasks through the duration of the scenario, they don't draw the same current and don't have similar consumption profiles. Therefore we evaluate the battery charge during the application to get more accurate view of the impact on power consumption.

Duty cycle in the evaluation scenario is arbitrarily set at 1%, mimicking a commonly observed set-up in LPWAN space. In our case, we set up the activity cycle duration for the first configuration at 200ms. This window of activity allows for collecting sensor data, processing it and performs a successful transmission to the rest of the infrastructure. The sleep cycle duration is by consequence of 20s, in regards to the duty-cycle set at 1%. For the second case in the scenario, where an AES code is added for permanent encryption of the collected data, the active cycle duration is set at 400ms to encompass the necessary duration to 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 the current consumption and discharge profile of the batteries. For specific events and periods of time, a digital multi-meter with data-logger capability is employed. The platforms use a couple of pair of retail store standard AA/LR6 batteries totalling an electric charge of 4800mAh. We run these experiments for the scenario duration of 48 hours and analyse the collected results in the following subsection.

5.3 Results

We collect and analyse the impact of our scenario on the energy consumption of our evaluated platforms. To that extent we focus on the evolution of the remaining battery charge after the duration of our scenario, and its various evolutions to cope with the requirements of the three different use cases. This global perception is illustrated in Figure 6 where the battery discharge curves of our platforms are represented. As identified previously during the Design Analysis in Section 4.1, the resource utilization of the core has a major effect on the power consumption of the concerned processing unit. With all platforms having a similar duty-cycle of 1%, the comparison of the elapsed power consumption between the basic core and the greedy AES one display some large disparities. At the end of our evaluation scenario, the basic configuration core has recorded a remaining battery charge of 99.64%. The heavy loaded core where an AES core has been added finished its run with a remaining battery charge of 99.46% marking a one third increase compared to the baseline core. Utilizing the extra modules in the design even when it is not needed comes with sensible cost of battery life, advocating for reconfiguration process over embedding unused elements susceptible to influence negatively

the overall performances. The third platform alternating between the two remains with 99.52% of battery charge. The reconfiguration process every 12 hours allows to save battery charge compared to the permanent use of AES core in the use case plotted in red, illustrating that the overall reconfiguration costs, depending of course of their frequency, are low enough in this scenario to allow a larger battery life time than the permanently dedicated to encryption platform.

The power consumption model of the baseline core and the one implementing AES are displayed in Figure 7. Each surge in current consumption represents the awakening of a platform, its task processing profile then the current drops suddenly, due to the re-entry in sleep mode. The difference in delay between the two spikes of activity of these two platforms is due to their respective activity cycle duration, presented in the above section, being respectively of 200ms and 400ms. As their duty-cycle is similar, their sleep cycle duration is thus with a factor 2 difference ratio. We observe that the sleep mode consumption appears to be of two orders of magnitude smaller than the active mode consumption for both platforms. Figure 8 provides us a more detailed view of the current consumption during the active cycle. Sensor reading, serial transmission to the RF Communication Unit and Bluetooth communication are the main tasks that affects the active cycle of the baseline core represented in green. The curve details nicely the wake up of the main computing unit in order to collect the data and communicates it to the RF Communication Unit, the turning off of the computing unit, the wake up of the radio module and it's transmission via BLE, then the busy-wait time to complete the active duration of 200ms before returning to a deep sleep state. On the second use case, in red in Figure 8, we witness a predominant rise of power consumption at the beginning corresponding to the AES computation after having retrieved the sensor measured value. The AES core is then turned off, and the data interface with the RF Communication Unit is then turned on for data transmission, like in the first use case. The rest of current activity is identical to the one in the first use case. The effects of an extra AES module and its proportions are now clearly visible.

As observed previously and illustrated with more details in Figure 9, one Chimera platform has multiple different reconfigurations processes happening during the evaluation scenario, each one having a cost in power, witnessed by sudden drops of battery remaining charge. In this second part of the results analysis, we focus on the cost of reconfiguration for Chimera. The platform has two types of reconfiguration process, one dubbed Offline and the second Online. Different hardware components are involved in each process. In Offline reconfiguration, one of the core images previously saved in the dedicated Secondary Memory Unit is selected then fetched by the RF Communication Unit and forwarded to the FPGA bit by bit towards JTAG interface. The overall process can be divided into four tasks and these four tasks can be observed in Figure 10. The Offline process starts by erasing the previously used image in the FPGA programming Array during the period denoted A1 in this figure. In period A2, the new image is read from the memory and programmed in the FPGA. Period A3 corresponds to the verification of the newly programmed image and period A4 is when FPGA's clock is programmed and the core is run for verification purposes. When we observe the Online reconfiguration process, we witness an increase in duration and overall power consumption. The difference originates from the use of the Bluetooth Low-Energy module to receive a remotely sent FPGA image. In the newly appeared period identified as A0, the RF Communication Unit wakes up, and start downloading the reconfiguration image and storing it bit by bit in the Secondary Memory Unit. This situation leads to an overall power consumption of respectively 61μAh and 95μAh for the Offline process and the Online one. As the evolution of the consumed current stays in the same order of magnitude, this difference is mainly due to duration of the reception and storage of the image, leading the Online reconfiguration process to happen in 45s compared to 31s for the Offline one.

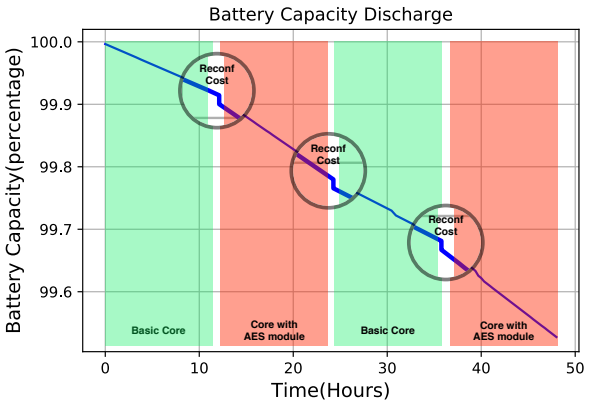

Figure 9. Battery Charge Trend during reconfiguration.

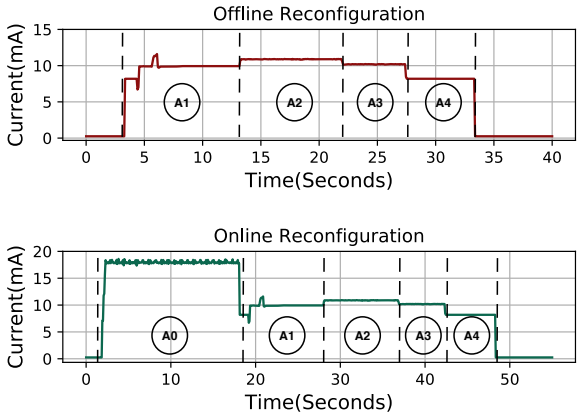

Figure 10. Current trend during reconfiguration.

5.4 Discussion

Evaluation shows that Chimera offers a deeply reconfigurable platform while achieving multi-year battery-life. Resource utilization of the FPGA directly affects power consumption, in return reconfiguring the FPGA to tailor the application needs can help to optimize power consumption. The Offline reconfiguration process costs 0.0013% of

Table 4. Comparison of different platforms.

Platform	Processing Unit	Radio module	Active Current	Sleep Current
Cookie[12]	Xilinx Spartan3 + ADuC841	2.4Ghz RF transceiver supports Zigbee	45mA-72mA	18mA
Sentiof[29]	Xilinx Spartan6 + Atmel AT32U	2.4Ghz RF transceiver Supports Zigbee	47mA(max)	95uA
Telos-Tmote[24]	MSP430F1611	2.4Ghz RF transceiver Supports Zigbee	18.8mA(max)	5.1uA
MicaZ[17]	Atmel ATmega128L	2.4Ghz RF transceiver	18.8mA(max)	10uA
SunSpot[22]	Atmel ARM920T	2.4Ghz RFtransceiver supports Zigbee	35mA(max)	520uA
Chimera	Igloo Nano250	2.4GHz transceiver supports BLE and ANT	19mA(max)	250uA

the batteries life and the Online reconfiguration via Bluetooth costs 0.002%. Up to four different FPGA images can be stored in the platform. Depending on the application environment, these images could cover any possible use case suppressing the need for an Online reconfiguration and the associated costs. Table 4 represents comparison of different platforms used in wireless sensor networks. Chimera consumes less active power than any reported FPGA based platforms[12],[29],[14],[26],[6],[13]. 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. Designing different hardware modules to enhance the platform features or prioritizing the autonomy duration of the end-device are modular choices offered by our architecture. In essence, Chimera combines the best of high-end FPGA based platforms and low-power application specific ones applied to low-power wireless network domain.

6 Future Work

The initial results of Chimera demonstrates the capacity of this experimental platform to perform in the low-power wireless sensors domain and motivates us to extend the work further. Although we used generic hardware components to support the tasks in it, a more cutting-edge trade-off between costs and power performances in sleep and active mode could be investigated further. One of the main primary future works includes using the platform where specialized hardware is required. Having a deeply reconfigurable and versatile battery powered device in a sensitive area with a dynamic approach on repurposable elements like in smart hospitals can reduce the need for specialized hardware in high speed signal processing tasks or multimedia applications. One of the extended directions for future work on a software level resides in the power saving algorithms. Having a framework able to perform automatically aggressive power savings operations based on an overall comprehension and integration of the task flow, would lead to shorter activity cycles, and a more tailored approach of the sleep mode regardless of the application, increasing drastically the battery life time. On a more performance oriented note, one future work direction could matching our platform with wireless acoustic sensor networks, being promising technology to perform surveillance based on sounds captured from the environment. Thoen et.al[32] suggests that with specialized hardware, acoustic surveillance applications can be performed in decentralized environment by battery powered devices. We believe that the versatile and low power nature of Chimera is suitable for these kind of applications and will increase re-usability of such dedicated devices. We would finally like to leverage existing literature on data processing in sensor networks and use the platform to perform heavy processing tasks in distributed environment (such as cooperative sensor networks) to reduce the need of centralized servers.

7 Conclusion

This paper presents a flexible and low-power experimental platform with multi-year battery life based on a Flash FPGA designed to introduced in the low-power wireless networks domain, designated Chimera. We justify the need for such platform by witnessing the rise of smart environments such as smart buildings or smart healthcare facilities, where traditional static low-power wireless sensors infrastructure are deployed in order to increase the understanding and control on these environments, while being confronted to dynamically evolving operating conditions as it be related to unplanned events or a need repurposable devices to match future uses, all of these being direct characteristics of a close proximity with human interactions. We identify two main reasons that prevents 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 a fully-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 the design analysis leading to hardware and software solutions around which Chimera is built, and present the physical product as well as a standard panel of IoT applications with different resources needs to illustrate the capacities of Chimera.

We evaluated the platform in its operating domain by focusing on its power consumption and presented the current

trend during normal operation, battery capacity discharge and the cost of reconfiguration in three different scenarios. The results show that Chimera can achieve multi years battery-life time with a pair of couple of standard retail store AA batteries, depending on the FPGA design and application requirements. 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 with multi-year battery life. With Chimera we intend to bridge the gap between low-power static end-devices performance and growing trends to extend the flexibility of an architecture as close as possible to its edge.

8 References

[1] 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.

[2] 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.

[3] 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.

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

[5] A. De La Piedra, A. Braeken, and A. Touhafi. Sensor systems based on fpgas and their applications: a survey. *Sensors*, 12(9):12235–12264, 2012.

[6] 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.

[7] 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.

[8] 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.

[9] O. Hahm, E. Baccelli, H. Petersen, and N. Tsiftes. Operating systems for low-end devices in the internet of things: A survey. *IEEE Internet of Things Journal*, 3(5):720–734, Oct 2016.

[10] Intel. Intel's fog reference design overview. Online, 2017.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] N. Matthys, F. Yang, W. Daniels, S. Michiels, W. Joosen, D. Hughes, and T. Watteyne. μpnp-mesh: the plug-and-play mesh network for the internet of things. In *Internet of Things (WF-IoT), 2015 IEEE 2nd World Forum on*, pages 311–315. IEEE, 2015.

[17] Memsic. Micaz wireless measurement system. Online.

[18] Microsemi. Igloo nano low power flash fpgas, 2015.

[19] S. Nolting. The neo430 processor. Online, February 2018.

[20] 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.

[21] Nwave. Smart parking solutions, 2018.

[22] Oracle. Sun spot. Online.

[23] 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.

[24] 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.

[25] J. Qiu, J. Wang, S. Yao, K. Guo, B. Li, E. Zhou, J. Yu, T. Tang, N. Xu, S. Song, et al. Going deeper with embedded fpga platform for convolutional neural network. In *Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pages 26–35. ACM, 2016.

[26] 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.

[27] T. Rinta-Aho, M. Karlstedt, and M. P. Desai. The click2netfpga toolchain. In *USENIX Annual Technical Conference*, pages 77–88, 2012.

[28] R. R. Rout and S. K. Ghosh. Enhancement of lifetime using duty cycle and network coding in wireless sensor networks. *IEEE Transactions on Wireless Communications*, 12(2):656–667, 2013.

[29] 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.

[30] 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.

[31] 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.

[32] 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.

[33] VersaSense. Fabric for the industrial iot, 2018.

[34] 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.

[35] C. Zhang, M. Zhang, Y. Su, and W. Wang. Smart home design based on zigbee wireless sensor network. In *7th International Conference on Communications and Networking in China*, pages 463–466, Aug 2012.

[36] 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.