Design and Optimization of a Low-Power RISC-V Processor for NDIR Measurement of CO2 Levels

Ricardo Núñez Prieto

UAB
Universitat Autònoma de Barcelona

Design and Optimization of a Low-Power RISC-V Processor for NDIR Measurement of CO_2 Levels

Microelectronics & Electronic Systems Department
Universitat Autònoma de Barcelona (UAB), Spain

A thesis submitted in fulfillment of the requirements
for the degree of Doctor of Philosophy
in Electronic and Telecommunication Engineering

— September 20, 2023 —

Author:
Ricardo Núñez Prieto

Directors:
Lluís Terés Terés
David Castells Rufas
Narcís Avellana Tarrats

I certify that I have read the dissertation "*Design and Optimization of a Low-Power RISC-V Processor for NDIR Measurement of CO_2 Levels*" and agree that it adequately fulfills all requirements as a dissertation for the degree of Doctor of Philosophy.

Universitat Autònoma de Barcelona
Departament d'Enginyeria Electrònica
Programa de Doctorat en Enginyeria Electrònica i de Telecomunicació

This work was mainly carried out at Institut de Microelectrònica de Barcelona - Centre Nacional de Microelectrònica, IMB-CNM (CSIC).

To my three precious sunshines,
Erik, Júlia, and Marta:
You are the joy, the pride,
and the love that light up my life.

Abstract

In various fields like environmental monitoring, smart sensors, and the Internet of Things (IoT), the increasing demand for low-power devices has led to growing interest in the development of high-performance and energy-efficient processors. In this ongoing quest, the RISC-V architecture emerges as a promising solution. In recent years, the RISC-V architecture has gained popularity as an open-source and customizable alternative to proprietary instruction set architectures (ISAs).

This thesis presents the comprehensive design, implementation, and performance analysis of RisCO2, a custom-designed soft-core RISC-V processor. It is specially optimized for energy-efficient IoT devices, with a focus on its feasibility for Non-Dispersive Infrared (NDIR) CO_2 sensors in environmental monitoring and air quality control. The research builds on the extensible and modular architecture of the RISC-V instruction set to develop a processor core that is tailored for low-power applications. RisCO2 has also been integrated into the PULPino System-on-Chip (SoC), providing a comprehensive system for evaluating its energy efficiency

The study employs an iterative design process using an FPGA implementation that incrementally incorporates architectural modifications aimed at reducing both power consumption and computation time, all without sacrificing performance. The process began with a basic RV32I core, allowing us to understand the fundamental characteristics of the RISC-V ISA as well as to acknowledge the flexibility provided by its modular design. As the project progressed, we selectively incorporated more specialized extensions and functional units into the processor.

We have placed heavy emphasis on the design's verification and validation. Through Register-Transfer Level (RTL) simulations, we confirmed the processor's compliance with RISC-V ISA specifications and its functional integrity. We further ensured consistency in instruction execution by comparing RTL simulations in Vivado with RISC-V ISA simulations in

SEGGER Embedded Studio. In subsequent design stages, we engaged in a meticulous process to prune extraneous hardware logic and simplify RTL modules. These steps streamlined the processor's architecture, aligning it more closely with the unique requirements of CO_2 sensing applications while also reducing both area and energy consumption.

The application used for benchmarking the processor employs the well-known digital quadrature demodulation technique to extract information from the sensor's digital samples. It then calculates the CO_2 concentration based on the Beer-Lambert law, which governs the behavior of light absorption in gases.

Additionally, the study conducts a comparative analysis with established RISC-V reference processors: Ri5cy, CV32E40P, Zero-riscy, and Micro-riscy. This comparison aims to assess how RisCO2 performs when integrated into real-world SoC frameworks like PULPino, providing a more comprehensive perspective on the design of energy-efficient processors. To perform this assessment, distinct projects were created targeting each of these reference processors. Through this approach, the processor's performance, resource utilization, and power consumption were scrutinized in detail.

Following implementation, in-depth power breakdown analyses were conducted using switching activity data from circuit nets, offering precise estimates of power consumption. This examination provided insights into how power consumption is distributed across the various processor modules. Finally, synthesis and area utilization were analyzed using TSMC's 65nm process technology library, utilizing the Cadence Genus synthesis tool for the assessment.

The results show that RisCO2 achieves a significant reduction in energy consumption while delivering performance comparable to that of the reference processors. These findings highlight the capability of custom RISC-V processors like RisCO2 to serve as effective solutions in gas concentration sensing applications. Such processors not only offer meaningful energy savings but also meet the requirements for efficient, low-power edge computing. Moreover, the implications of this study extend to the design of energy-efficient processors in diverse applications, including the IoT, wearable technology, and mobile devices.

In summary, this work contributes to the ongoing efforts aimed at improving processor energy efficiency and promoting sustainable computing. It serves as a reference point for future investigations, offering both a methodological approach and empirical data that can guide the development of high-performance, low-power processors.

Acknowledgements

First and foremost, I extend my gratitude to my thesis directors: Prof. Dr. Lluís Terés at the Institute of Microelectronics of Barcelona (IMB-CNM), Prof. Dr. David Castells at the Autonomous University of Barcelona (UAB), and Dr. Narcís Avellana at NVISION Systems & Technologies. Their invaluable mentorship made this journey not only possible but also enlightening. I am especially grateful for their willingness to offer their time, expertise, and guidance throughout my research. The opportunity to enroll in this unique Industrial Ph.D. program has been a transformative experience, and I owe a significant part of that to them.

Additional thanks are also due to Prof. Dr. David Castells, who contributed to the development of main software tools used in this study. His expertise in setting up and performing the power measurements on the designed system was invaluable to the successful completion of this thesis.

My appreciation to Ricardo Martínez at IMB-CNM for his work in generating the ASIC design synthesis results and for the floorplanning and the chip layout of the system.

I want to express my thanks to the Barcelona Supercomputing Center (BSC) for providing the FPGA cards essential for our power measurements. Particular recognition is due to my workplace supervisors at BSC, Óscar Palomar and Francesco Minervini, for their understanding and support, enabling me to harmonize my academic and professional commitments.

Finally, my deepest gratitude goes to my wife and children, whose love and support have been my cornerstone through all challenges. I also want to extend heartfelt respect to my parents and brothers for their unconditional esteem and for always being there for me.

Ricardo Núñez Prieto
Bellaterra, September 2023

List of Acronyms

ADC Analog-to-Digital Converter. 67, 70, 71

ALU Arithmetic-Logic Unit. 65

ASIC Application-Specific Integrated Circuit. 10

BRAM Block RAM. 87, 91

CO_2 carbon dioxide. xi, 1, 5, 7–10, 12, 14, 15, 17, 25, 46, 48, 49, 51, 63, 65, 67–70, 73–76, 78, 81, 83, 84, 88, 90

CSR Control and Status Register. 60, 61

DSP Digital Signal-Processor. 60, 87, 91

DVFS Dynamic Voltage and Frequency Scaling. 34

EX Execution. 57, 59, 60, 65

FF Flip-Flop. 87, 91

FMA fused multiply-add. 60, 65

FPGA field-programmable gate array. 1, 7, 9–11

FPU Floating-Point Unit. 60, 63, 65

GPR General Purpose Register File. 64

HDL hardware description languages. 9

ID Instruction Decode. 57, 58, 64

IF	Instruction Fetch. 56, 64
IoT	Internet of Things. 1, 3, 4, 26, 32, 67
IPC	Instructions per Cycle. 61
IR	Infrared. 67
ISA	instruction set architecture. 1, 6–9, 18, 19, 60, 88
LED	Light-Emitting Diode. 67
LPF	low-pass filter. 71
LSU	Load-Store Unit. 61
LUT	Look-Up Table. 87, 91
MEM	Memory Access. 57, 65
NDIR	non-dispersive infrared. 12, 13
PC	program counter. 65
RISC	Reduced Instruction Set Computer. 20, 50
RTL	Register-Transfer Level. 36, 88, 89
SAIF	Switching Activity Interchange Format. 7, 11
SES	SEGGER Embedded Studio. 83, 84
SoC	System-on-Chip. 50
VFI	Voltage-Frequency Island. 36
WB	Write-Back. 57, 65
WSN	Wireless Sensor Network. 50, 67

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

The development of energy-efficient processors is becoming increasingly important as the demand for low-power devices continues to grow in various fields, such as the Internet of Things (IoT), wearables, and mobile devices. In particular, battery-powered devices require processors that can operate at low power levels to extend battery life.

This work explores the energy efficiency of RISC-V processors and their potential for use in battery-powered devices as well as devices that use energy-harvesting techniques. Energy-harvesting techniques refer to the process of capturing energy from one or more renewable energy sources and converting them into usable electrical energy [1]. The RISC-V architecture is an open-source instruction set architecture (ISA) that has gained popularity in recent years due to its flexibility, scalability, and low power consumption.

Our primary objective is to assess the applicability of RISC-V processors in low-power CO_2 concentration sensing, which is a critical application in environmental monitoring and control. The evaluation includes performance and energy consumption metrics of a custom-designed RISC-V processor in comparison to existing RISC-V references, using a field-programmable gate array (FPGA) implementation.

The outcomes of this work offer valuable insights into the development of processors tailored for low-power applications across various domains. Moreover, the methodologies employed can serve as a practical guide for future design and implementation efforts in the area of energy-efficient processing units.

1.1 Background and Motivation

RISC-V is a free and open ISA that was developed at the University of California, Berkeley, in 2010 [2]. It is designed to be simple and modular, making it easy to customize for specific applications. The ISA is divided into two parts: the base ISA and the optional extensions. The base ISA includes the essential instructions required for executing a broad range of programs. In contrast, the optional extensions are designed to offer hardware-accelerated support for advanced functionalities, such as floating-point operations, vector processing, bit manipulation, and atomic operations. While these extensions do not introduce functionalities that could not be implemented through software, they do significantly enhance performance by reducing computational time and overhead.

The RISC-V ISA presents a flexible architecture that can be tailored for various implementations, including those that prioritize low power consumption. The ISA was designed with energy efficiency in mind, making it particularly beneficial for battery-powered and IoT devices. Here are some of these key aspects:

1) Large Register File: RISC-V has a large register file with 32 registers, which can reduce the frequency of memory accesses, thus conserving energy. Fewer memory operations not only save power but also enhance performance by reducing data transfer times.

2) Orthogonality of Operations: The RISC-V ISA's design allows for any register to be used in any operation, providing greater flexibility in instruction scheduling. This can lead to more efficient use of hardware and therefore lower energy consumption.

3) Simplified Instruction Encoding and Decoding: RISC-V has a simple and regular instruction set, making the instruction decoding process less complex. A simpler decoding logic can be faster and consume less energy. The compactness of certain instructions can also lead to energy savings during instruction fetch operations.

4) Low-Power Instructions: Instructions like WFI (Wait For Interrupt) can put the processor into a low-power state until an interrupt is received, contributing to energy savings. This is particularly useful in event-driven or interrupt-based systems, where the processor needs to be idle most of the time, waiting for an event to occur.

5) Modular Design with Extensions: One of the key advantages of RISC-V is its extensibility. Custom instructions tailored for specific workloads can

be added, potentially reducing the number of cycles required to complete operations and, hence, the energy consumed.

In addition, RISC-V is highly scalable, which means that it can be used in a wide range of applications, from small embedded systems to large-scale data centers. Another advantage of RISC-V is its flexibility. The open-source nature of the ISA means that it can be customized to meet the specific needs of different applications. This makes it an attractive option for developers who want to create custom processors for their products.

As previously discussed, the flexibility of the RISC-V ISA allows for implementations that are well-suited to meet the energy-efficiency requirements of IoT devices. As the IoT industry aims to achieve a 10-year battery lifespan for these devices [3], power efficiency becomes crucial. RISC-V's design emphasis on energy efficiency aligns perfectly with this goal. By leveraging the scalability of the RISC-V architecture, IoT developers can create power-efficient solutions that meet the long battery life expectations. Additionally, the flexibility inherent in RISC-V allows developers to tailor the processor design to the specific needs of IoT applications, further optimizing power consumption and extending battery life.

To gain a comprehensive understanding of the battery lifespan challenge, according to EnABLES [4], an EU-funded project that is urging researchers and technologists to take action to ensure that batteries outlive the devices they power, about 78 million batteries powering IoT devices will be dumped globally every day by 2025 if nothing is done to improve their lifespan. Currently, the majority of devices are designed to have a lifespan of more than ten years, yet the batteries that fuel them typically last for two years or even less. This discrepancy leads to the recurring need for battery replacements, resulting in significant economic and environmental implications. The production and disposal of hundreds of millions of batteries on a daily basis pose notable challenges in terms of cost and sustainability.

To make matters even more concerning, a recent report from 2019 by Everactive, Inc., a spin-off company from the Massachusetts Institute of Technology (MIT) specializing in batteryless IoT devices using energy-harvesting techniques, highlights the daunting reality. Even if the IoT industry manages to achieve its ambitious goal of a 10-year battery lifespan, the projected number of daily battery replacements in a potential trillion-device world (as predicted by IBM in 2012 [5]) would be astonishing. To be precise, it would exceed a quarter of a billion battery replacements every day.

The situation becomes even more worrisome if the industry falls short of this goal and settles for a two-year battery lifespan. In such a scenario, every individual on the planet, with a population of approximately 7.4 billion, would need to replace a battery every five days. Even in the best-case scenario where all batteries reach their full 10-year life expectancy, the need to replace over a quarter of a billion batteries daily to power 1 trillion IoT devices is undoubtedly an impractical proposition.

Due to this pressing challenge, companies like Everactive are dedicating their efforts to research and develop batteryless, self-powered sensors. These sensors utilize ambient energy sources to operate, such as low-level indoor solar, the thermoelectric effect, vibrations from piezoelectric materials, and even radio waves present in the environment. Remarkably, these sensors not only collect a wide range of data using multiple sensors but also process, analyze, and wirelessly transmit that data, all within the constraints of a batteryless power budget.

In conclusion, the main motivation behind this thesis work lies in the urgent need for energy-efficient processors tailored for low-power IoT applications. As the IoT industry strives to achieve extended battery lifespans and reduce power consumption, the role of efficient processors becomes paramount. With its focus on low power consumption, scalability, and customization, RISC-V emerges as a compelling solution to address these challenges.

By exploring the potential of RISC-V in enabling energy-efficient processors, this thesis aims to contribute to the advancement of sustainable IoT systems. Through hands-on design and advanced optimization strategies, we seek to empower IoT developers with the means to create power-efficient solutions, prolong battery lifespans, and reduce environmental impact. The engineering efforts in this work aspire to shape the future of low-power IoT applications by harnessing the full potential of RISC-V processors.

1.2 Thesis Goal

This work is conducted within the framework of an industrial Ph.D. program, and as such, it merges both academic research and practical design and implementation. The primary objective is to design, implement, and evaluate an energy-efficient RISC-V processor tailored specifically for low-power signal-processing applications in the IoT space. While the academic component aims to explore and potentially introduce novel methodologies for achieving energy efficiency, the industrial aspect mandates a focus on practical, real-world applications and solutions. The study responds to the growing demand for efficient processors capable of performing signal-processing tasks while minimizing power consumption.

Due to the lengthy cycle associated with ASIC production, which exceeds the duration of a typical Ph.D. thesis, FPGA design serves as a practical proxy for hardware implementation. This approach enables a more agile route to a complete and testable design. Furthermore, we will estimate power consumption through simulations carried out in both FPGA and ASIC environments.

Our work extends beyond processor design to scrutinize the efficient allocation of FPGA resources, affecting the feasibility of realizing RISC-V processors on hardware platforms with constrained resources. This involves a meticulous examination of how the processor utilizes resources and considers the available resources on the hardware platform. The goal is to achieve the best possible performance while operating within the given limitations. By carefully managing resource utilization, our work aims to optimize the overall performance of the RISC-V processor when deployed on resource-limited hardware platforms.

To demonstrate the feasibility of using custom RISC-V processors for low-power IoT applications, our focus narrows on a proof-of-concept application centered around the measurement of carbon dioxide concentrations using optical sensors. This specific application serves as a representative case to showcase the effectiveness of the designed RISC-V processor in handling signal processing tasks with minimal power consumption.

Target Application: Low-power CO_2 Concentration Sensing

Typically, low-power CO_2 concentration sensing applications are of interest because they have important implications for environmental monitoring and control since CO_2 is a greenhouse gas that contributes to climate change, and accurate monitoring of CO_2 concentrations is essential for assessing the impact of human activities on the environment and identifying potential sources of pollution.

Moreover, CO_2 sensors are used in a variety of applications, such as industrial safety, where it is crucial for detecting leaks in gas pipelines and preventing accidents, also in indoor air quality monitoring and control for ensuring a healthy and comfortable indoor environment, as high levels of CO_2 can cause headaches, dizziness, and other health problems, and can even be used as a low-cost tool to assess ventilation and risk of contagion of COVID-19 for high occupancy spaces [6]. In many of these applications, the sensors are battery-powered and require low-power processors to extend battery life. This is especially relevant in remote or hard-to-reach locations where battery replacement or recharging may not be feasible.

Our strategy aims to transform a conventional gas sensor into a smart sensor by incorporating components that enable functionalities beyond simple measurement. Typically, a smart sensor integrates elements for data conversion, data processing, and communication within a single unit. Our primary focus is on optimizing the energy efficiency of the processor, which handles tasks such as signal processing from the sensor and periodic computation of gas concentrations.

Because smart sensors have the capability to analyze data locally, another goal of ours is to significantly reduce the overall energy footprint. This reduction will primarily come from minimizing the need for constant communication with a central system, which can save energy in some applications.

Consequently, the design and implementation of energy-efficient processors for low-power CO_2 concentration sensing emerge as a critical industrial application with the potential for significant societal impact, contributing to sustainable computing. Moreover, leveraging low-power processors may unlock new device functionalities and applications previously hindered by energy limitations.

1.3 Main Challenges

This section delves into the principal challenges that arise in the quest for an energy-efficient and high-performance processor design. The development of a customized RISC-V processor demands a meticulous optimization process to achieve the desired balance between power efficiency and computational capabilities. The section examines the intricacies involved in designing and optimizing the processor for energy efficiency, considering specific ISA features, custom instructions, and hardware accelerators to enhance signal-processing capacities.

A more comprehensive approach to tackle these challenges is expounded later in Chapter 3, "Methodology", where an in-depth exploration of the design process and optimization techniques for energy-efficient processors is presented.

1.3.1 Design and Optimization of the Processor for Energy Efficiency

One of the key challenges of this work is the development of a customized RISC-V processor from the ground up, with a specific focus on optimizing energy efficiency while maintaining high performance. It is worth emphasizing that the processor's design will be highly optimized for the target application, as mentioned in the previous section, which involves processing the digital signal of low-power optical CO_2 sensors. Therefore, our focus is on creating an application-specific design for the processor. This involves a thoughtful examination of the ISA features, exploring the potential utilization of custom instructions and hardware accelerators to enhance the processor's efficiency in handling signal-processing tasks.

The initial design of this processor started with a basic RV32I architecture featuring a 32-bit instruction set, an in-order execution model, and a five-stage pipeline. To improve performance, incremental enhancements were implemented, starting with the inclusion of integer multiplication and division modules. Subsequently, a floating-point unit was integrated, further augmenting the processor's capabilities. Additionally, the adoption of optional extensions offered by the RISC-V ISA (specifically the 'E' and 'Zfinx' extensions) played a crucial role in optimizing resource utilization and maximizing energy savings. These extensions significantly reduced the processor's resource requirements while simultaneously enhancing its energy efficiency.

By carefully considering the ISA features and leveraging specific functional units, our aim is to create a highly efficient RISC-V processor that strikes an optimal balance between energy efficiency and performance, catering to the specific requirements of signal-processing tasks.

1.3.2 Estimation of Processor's Power and Energy Consumption

Accurately measuring the power consumption of the processor during application runtime poses a significant challenge. This can be achieved through two approaches: 1) Real-world power measurements, using a power analyzer connected to the FPGA configured with our processor design; 2) Post-route simulations to collect switching activity data across the entire circuit. These simulations enable precise power consumption estimates, as they provide detailed information on the switching activity of individual transistors over time. This data is stored in Switching Activity Interchange Format (SAIF) files, a format developed by Xilinx. By leveraging SAIF files, power analysis tools can assess the average power consumption of the entire circuit or specific design sections. This enables designers to identify power-hungry areas and make informed decisions to optimize power consumption in their designs.

1.3.3 Evaluation of Processor's Performance

Finally, an additional significant challenge is to assess the performance of our RISC-V processor in comparison to other well-known reference processors within the RISC-V community, such as Ri5cy or Zero-riscy, designed by the PULP Team from ETH Zürich. For this purpose, we employed the PULPino platform, also developed by the same group. In order to conduct a comprehensive evaluation, it is essential to utilize a benchmark application that encompasses typical signal processing tasks encountered in CO_2 sensors to ensure their relevance and representativeness in real-world applications.

1.4 Research Approach and Strategy

This section will provide further elaboration on the challenges discussed earlier and describes the approach and strategies employed to address them. It covers not only the specific methods but also the overall conceptual framework or theoretical basis guiding this work, and it aims to offer a concise overview of our practical engineering strategies and potential innovations, which will be substantiated by detailed data and results in subsequent chapters.

1.4.1 Design of a RISC-V Processor

First and foremost, the availability of a RISC-V processor is crucial to meet the objectives laid out earlier. There are two pathways to consider: adopting an existing processor from publicly available options or undertaking the engineering challenge of creating a tailored processor from the ground up. The first option entails carrying out a comprehensive analysis of the processors available and selecting the most suitable one for the target application but requires an extensive knowledge of the ISA in order to make a well-informed decision. Moreover, certain processors utilize a Chisel-based RTL description, such as the Rocket core [7] developed by the University of Berkeley. Chisel is a relatively new hardware description language that significantly differs from the widely used Verilog. Therefore, introducing modifications to enhance energy efficiency at later stages could be complex if Chisel is not fully mastered. Furthermore, modifying a pre-existing processor designed by a third party can be time-consuming as it requires understanding the intricacies and interdependencies of the design. The second option, which involved designing a custom processor from scratch and walking the learning curve from the bottom up, provided not only invaluable insights into the RISC-V ISA but also yielded remarkable success. This approach deepened our understanding of the architecture and enhanced our confidence in extending the processor's capabilities by incorporating additional functional units and available extensions offered by the RISC-V ISA.

1.4.2 Soft-Core Processor Approach

We have chosen a *soft-core processor* approach to design our custom RISC-V processor. Soft-core processors are typically implemented using hardware description languages (HDL) like Verilog or VHDL. The processor's architecture and instruction set can be designed and programmed to meet specific requirements or tailored for a particular application. This flexibility allows for easy modifications and adaptations without the need for physical changes in the hardware. Examples of soft-core processors include the Xilinx MicroBlaze [8] and Intel (previously Altera) Nios II [9].

Using a soft-core processor approach for designing our RISC-V core offers several advantages:

- Customization: It allows for customization of the processor's architecture and instruction set to meet the specific requirements of the application. This flexibility enables tailoring the processor to optimize performance and energy efficiency for CO_2 concentration sensing applications.
- Iterative design: It enables rapid prototyping and iteration, allowing designers to quickly make modifications and improvements to the processor's design. This iterative design process helps refine the core and optimize its functionality.
- Hardware resource utilization: Soft-core processors are implemented in programmable logic devices like FPGAs, which provide ample resources for incorporating additional features and extensions. This allows for the integration of hardware accelerators and specialized instructions to enhance the performance and energy efficiency of our processor.

- Debugging and testing: It offers easier debugging and testing capabilities compared to fixed hardware implementations. Designers can use simulation and debugging tools specific to the chosen hardware description language to identify and resolve any issues efficiently.

- Scalability: Soft-core processors are highly scalable, making them suitable for a wide range of applications. Implementing our core as a soft processor, it can be easily adapted and scaled for different CO_2 sensing requirements, from low-power IoT devices to more powerful embedded systems.

It is worth mentioning that soft-core processors play a crucial role in the process of reaching the final Application-Specific Integrated Circuit (ASIC) design, serving as an essential development phase for architectural exploration and optimization. They serve as an intermediate stage, allowing us to explore and optimize the processor's architecture before moving on to the ASIC implementation. Once the soft-core processor design is thoroughly validated and optimized on the FPGA, the understanding gained from this iterative process can be leveraged to create the final ASIC design. By starting with a soft-core processor, we can fine-tune our architecture, ensure that it meets performance targets, and reduce the risk of errors or costly redesigns in the final ASIC implementation.

In summary, soft-core processors can harness the advantages of open-source ISAs, such as RISC-V, to optimize our processor architecture for specific applications. With the creative freedom that RISC-V offers, we can implement the best possible architecture tailored to our needs. Through this iterative process of soft-core exploration and optimization, we fine-tune the core's design, ensuring energy efficiency and scalability for low-power IoT applications. As a result, the realization of an energy-efficient and scalable 32-bit RISC-V core for low-power IoT applications becomes feasible.

1.4.3 Hardware Platform

In line with our soft-core processor approach, we have decided to utilize an FPGA as the hardware platform for implementing and evaluating our custom RISC-V processor. This choice was made for several compelling reasons.

Firstly, FPGAs are highly configurable and can be programmed to implement custom logic circuits, making them well-suited for the development of custom processors. This allows us to implement our proposed improvements and evaluate the performance of our processor in a realistic hardware environment.

Secondly, FPGAs are widely used in the development of embedded systems and IoT devices due to their low power consumption and high performance. As such, they serve as an excellent testbed for scrutinizing both the energy consumption and operational capabilities of processors intended for low-power applications.

Thirdly, FPGAs provide a robust platform for prototyping and iterative testing, offering high levels of flexibility and the ability to be reprogrammed multiple times. This adaptability allows us to refine our design continuously and evaluate various processor configurations to maximize performance and energy efficiency.

Lastly, considering the lengthy production cycles associated with ASICs, FPGAs offer a quicker route to a functional hardware implementation, enabling timely iterations and evaluations. This accelerated timeline is particularly beneficial when the goal is to swiftly move from concept to a validated design.

We selected the Nexys-4 development board as our preferred choice, which incorporates the XC7A100T-1CSG324C FPGA device. This general-purpose board was specifically chosen for our design and implementation process. The board has 4.860 Kib of block RAM, 63.400 LUTs, 126.800 FFs, and 240 DSPs.

In addition to selecting the FPGA platform, the choice of a suitable design environment suite is essential as it provides the necessary tools to streamline the design tasks. For this purpose, we utilized the Vivado Design Suite 2020.2 (HLx edition) developed by Xilinx. This comprehensive suite served as our primary tool throughout the design and implementation process. Furthermore, an additional compelling reason for choosing the Vivado IDE is its robust power analysis tool. This tool plays a crucial role in evaluating the power and energy consumption of our processor, providing valuable insights into its efficiency and helping us optimize its performance.

To obtain a detailed estimation of the dynamic power consumed by our processor and derive the energy, we need to obtain the switching activity of all the gates of our circuit when executing our application. Xilinx Vivado allows capturing this activity in post-implementation timing simulation, generating SAIF files that are used to provide detailed power estimates for different regions of the FPGA fabric. SAIF files typically contain information such as the signal names, time values, and switching activity levels for each signal in the design. The switching activity levels are represented as probabilities or counts of signal transitions, indicating how frequently a signal changes state (from 0 to 1 or vice versa) over a given time interval. While Vivado is capable of accurately obtaining switching activity for certain sections of the design, it relies on a probabilistic approach to estimate power consumption for other components, such as memories.

1.4.4 Benchmark Program for CO_2 Concentration Measurement

To effectively measure and evaluate the performance and capabilities of our custom RISC-V processor, the utilization of a benchmark program is essential. A benchmark program is a standardized software application or set of tasks specifically designed to assess and gauge the performance and capabilities of a computer system or a specific hardware component. It offers a consistent and reproducible workload that enables fair comparisons between different systems or components operating under identical conditions. As previously discussed, the selected application to showcase the effectiveness of the proposed processor is CO_2 concentration sensing. We have thoroughly justified our choice of this application and its relevance in previous sections.

CO_2 Sensors Based on Infrared Absorption Spectroscopy

Gas concentration sensing encompasses various methods, and our focus is on the optical techniques that exploit the phenomenon of light absorption by the target gas. When the frequency of light aligns with the vibrating electrons' structure in materials, absorption occurs. As a result, different gas molecules exhibit distinctive absorption profiles. Many gases of industrial and environmental significance exhibit frequency absorbance peaks in the near-infrared (NIR) and mid-infrared (MIR) spectrum regions, including CO, CO_2, NO_2, NH_3, among others. This characteristic has fostered extensive research and development in systems operating within this frequency range.

After thorough consideration, we have determined that our benchmark application will be tailored to a specialized gas sensor that utilizes non-dispersive infrared (NDIR) spectroscopy. This advanced sensor technology accurately measures gas concentration by analyzing the intensity of light reaching the photodetector. The detected light intensity is directly influenced by the gas quantity present in the sample, allowing for precise quantification of gas concentration levels.

In contrast to alternative methods such as chemical sensors, optical techniques based on infrared (IR) absorption spectroscopy offer fast measurement times in addition to high sensitivity. Moreover, these optical techniques are highly suitable for miniaturization, making them ideal for integration into IoT devices. An additional advantage of NDIR sensors is their non-reactivity with the target gas, which is particularly advantageous in harsh industrial environments. Specifically, NDIR sensors utilizing a source-detector configuration with LED and photodiode components offer the added benefit of low power consumption. This characteristic is particularly beneficial for battery-powered IoT devices, ensuring efficient energy usage.

A commonly employed technique in optical gas sensors involves modulating the light source used to interact with the sample gas. This modulation typically occurs at a much lower frequency compared to the sampling frequency used by the analog front end connected to the photodetector. The primary reason for modulating the light source in NDIR sensors is to improve the signal-to-noise ratio and enhance the accuracy of gas concentration measurements. Modulation allows the sensor to distinguish the desired signal from ambient noise and interference. By modulating the light source, the processor can extract the desired signal at the modulation frequency, effectively filtering out any noise or interference that does not share the same modulation characteristics. This modulation technique helps to enhance the sensitivity, stability, and overall performance of NDIR gas sensors.

Benchmark Application and Dataset Generation

The demodulation technique employed to extract the signal from the sensor's sampled data is based on asynchronous quadrature demodulation. This process occurs on the RISC-V microprocessor, utilizing a specialized software algorithm that involves various algebraic operations, such as floating-point addition, multiplication, division, and square root, as well as transcendental functions like trigonometric functions (sine and cosine) and natural logarithm.

The asynchronous quadrature demodulation technique is a method used to extract information from a modulated signal, typically used in digital signal processing. It involves demodulating a signal by sampling it at specific time intervals, which are not synchronized with the signal's carrier frequency. Instead of using a fixed clock, this technique uses variable sampling times based on the signal's characteristics. It is especially useful when dealing with signals that may experience frequency drift or variations over time or when synchronization with a fixed clock is challenging or impractical, making it a flexible and efficient approach for demodulating signals in various applications, such as communication systems, radar, and digital signal processing. The technique captures both the in-phase and quadrature (90-degree phase-shifted) components of the modulated signal, which contain valuable information about the original signal's frequency and phase.

One significant advantage of employing digital demodulation instead of analog demodulation is its capability to operate with a significantly lower signal-to-noise ratio [10]. Digital systems are inherently less susceptible to waveform distortion, enhancing the accuracy of signal extraction. Moreover, digital demodulation can be efficiently implemented in software, making it adaptable for execution on our application-specific processor. This versatility enables precise and efficient signal processing, contributing to improved overall performance.

Regarding the digital samples extracted from the gas sensor's output signal, which are typically obtained through an ADC in the analog front-end, we have devised a streamlined approach. Instead of dealing with real-time sensor readings during testing, we incorporated the digital sample values into a file. This file is then compiled together with the application algorithm, allocating specific space in the data memory. During run-time, the processor can directly access the stored samples. This approach simplifies data management and facilitates seamless integration of the samples into the application.

The advantage of using this method is twofold. Firstly, it simplifies the testing process by eliminating the need for physical gas sensors connected to ADCs in a laboratory setup. Additionally, benchmark instruments that provide precise control over CO_2 concentration volumes are no longer required. Instead, we opted for a software simulation approach using Python scripts in an interactive computational environment (Jupyter Notebook) [11]. Through these scripts, we simulate the modulation process by combining the sensor signal corresponding to a specific gas concentration with a carrier wave having a designated modulation frequency.

The composite signal undergoes quantization and sampling at a specific frequency, all accomplished through software. The resulting data is then incorporated into a C header file. Subsequently, on the processor side, the demodulation process takes place, extracting the CO_2 concentration value from the samples. This extracted value is then compared with the designated value utilized in our Python script, which was initially employed to generate the samples. By performing this comparison, we can validate the accuracy and effectiveness of the demodulation process implemented in the processor. This iterative approach allows us to fine-tune and optimize our processor for precise and efficient CO_2 concentration sensing applications.

1.4.5 Processor Implementation in PULPino SoC for Performance Comparison

As part of this effort, we aim to assess the capabilities and efficiency of our custom RISC-V processor, optimized for low-power CO_2 concentration sensing applications. To achieve this objective, we turn to the PULPino SoC platform [12], which provides a suitable environment for implementing our processor and conducting performance comparisons.

The PULPino SoC, developed by the PULP Team from ETH Zurich [13], serves as an excellent reference platform due to its flexibility, scalability, and extensive use of the RISC-V ISA. With its customizable nature, the PULPino SoC allows us to integrate our optimized processor and tailor it to our specific application domain. Furthermore, leveraging the platform's integrated peripherals and memory hierarchy, we can thoroughly evaluate the processor's performance and energy efficiency in realistic scenarios.

By utilizing the PULPino SoC and comparing our custom processor with other existing processor designs, including reference processors like Ri5cy and Zero-riscy, which are provided as part of the PULPino SoC, we aim to validate our processor's capabilities, assess its performance, and demonstrate its potential for low-power CO_2 concentration sensing applications.

In this comparison, we utilize several key figures of merit to assess the performance and efficiency of each processor. These metrics include the FPGA resource utilization, the number of instructions needed to complete the algorithm, the application execution time, and the estimated average power and energy consumption. By analyzing these essential parameters, we gain valuable insights into the strengths and weaknesses of each processor design, enabling us to make informed decisions in optimizing our custom RISC-V processor to achieve the desired energy efficiency and performance targets.

1.5 Overview of the Thesis Structure

The structure of this thesis is designed to provide a comprehensive understanding of the design and optimization of energy-efficient processors for IoT applications using the RISC-V ISA. This section provides an overview of the main sections and chapters of the thesis.

The introductory chapter sets the stage, explaining the background and motivation for this work. It underscores the importance of developing energy-efficient processors for IoT applications that require efficient processing and low power consumption. The chapter also discusses the battery problem in the potential 3-trillion IoT device scenario predicted by IBM and presents RISC-V as a viable solution to such challenges. The primary aim of this project, focused on crafting efficient low-power processors, is clearly articulated, along with the selected target application as a case study for the designed processor.

The second chapter of the thesis is a review of the current state of the art, which provides an overview of the RISC-V architecture and related work. It explains the RISC-V instruction set architecture, including its design principles, features, and advantages over other architectures such as RISC, MIPS, and ARM. The chapter also describes the ISA extensions available so far, explaining the advantages and drawbacks and remarking on the ones that have been implemented in our processor. Additionally, the chapter lists and describes some of the most known processors used in the RISC-V community, commenting on the supported architecture, pipeline stages, bit-width of the data registers, target applications, etc.

Chapter three, the Methodology, outlines the processor's design strategies and techniques aimed at enhancing energy efficiency. It rationalizes the choice of a pipelined architecture and delves into the technical aspects like the impact of pipeline stages, hazard management, and data forwarding. The chapter concludes with the various methodologies employed to estimate the power and energy efficiency of the designed processor.

The fourth chapter, Results and Analysis, presents the performance metrics related to the processor's power and energy consumption. It also offers a comparative study of the designed processor against other RISC-V processors, focusing on parameters like power consumption, resource utilization, and other relevant metrics.

The concluding fifth chapter synthesizes the major contributions and findings of this work. It also identifies limitations and suggests avenues for future enhancements to the processor's performance.

Overall, this thesis contributes to the development of energy-efficient processors for IoT applications, a pivotal aspect for the industry's growth and sustainability.

Chapter 2

Review of the State-of-the-Art

In this chapter, we review cutting-edge developments and knowledge pertinent to our project. The exploration encompasses several key areas, shedding light on the state-of-the-art in various domains.

Firstly, we embark on an insightful journey through the realm of the RISC-V architecture and its related work. Understanding its historical evolution, design principles, and distinct features empowers us to grasp its advantages over conventional architectures. Additionally, we explore the extensions that empower RISC-V to embrace innovative techniques, making it a preferred choice for modern computing systems.

Next, we delve into the realm of energy-efficient processor design techniques. This section presents an array of state-of-the-art strategies employed to achieve optimal energy consumption in contemporary processors. From low-power design methodologies to dynamic voltage and frequency scaling, we uncover a multitude of approaches adopted to enhance energy efficiency, enabling us to comprehend the evolving landscape of processor design.

Finally, we venture into the domain of optical gas sensors and their applications. By unraveling the underlying principles of optical gas sensing, like the Beer-Lambert law, we gain insight into their operational mechanisms. Further exploration leads us to discover the various types of optical CO_2 sensors and their diverse applications, from industrial processes to environmental monitoring and beyond.

This comprehensive review of existing literature serves as the groundwork for our subsequent analyses, offering crucial background for the empirical and design-focused activities carried out in this work.

2.1 Overview of the RISC-V Architecture and Related Work

The RISC-V architecture has emerged as a revolutionary force in the domain of processor design, redefining the landscape of open-source Instruction Set Architectures. As a versatile and extensible ISA, RISC-V has captured the attention of researchers, engineers, and developers alike, driving a wave of innovation and exploration in the field of computer architecture.

This section presents an in-depth overview of the RISC-V architecture and related work, delving into its historical evolution, design principles, key features, and advantages over other well-established architectures such as RISC, MIPS, and ARM. Additionally, we will explore the diverse extensions that empower RISC-V to accommodate diverse applications, expanding its capabilities beyond conventional boundaries.

The subsections within this section provide a comprehensive glimpse into the fascinating journey of RISC-V, showcasing its adaptability and scalability in different computing environments. From its historical roots to its present-day prominence, the RISC-V architecture stands as a testament to the collaborative spirit of the open-source community and its potential to drive innovation in the world of processor design.

2.1.1 History of the RISC-V ISA

The RISC-V ISA was designed at the University of California, Berkeley. The project was initiated in 2010 by Professor Krste Asanović, along with his colleagues Andrew Waterman, Yunsup Lee, David Patterson, and others. They aimed to create an open, free, and extensible ISA that could foster innovation, research, and development in the field of computer architecture. The team's goal was to offer a versatile and efficient ISA that could be customized and extended to suit various applications and target platforms. Over time, RISC-V gained traction and evolved into a widely used and influential open-source architecture, attracting interest from both academia and industry.

The RISC-V Foundation, established in 2015, is a non-profit organization that now oversees the development and promotion of the RISC-V ISA, ensuring its continued growth and standardization, and it has become a significant player in the processor ecosystem, with many companies and institutions contributing to its advancement and adoption.

2.1.2 Features and Design Principles

The RISC-V ISA was designed with several key principles in mind. First and foremost, it aimed for simplicity, ensuring that the instruction set is easy to implement, understand, and extend. It follows a modular approach, allowing for different optional extensions to be added as needed for specific applications without affecting the core ISA. This modularity ensures flexibility and adaptability for a wide range of use cases.

Another significant aspect of the RISC-V ISA is its support for both 32-bit (RV32) and 64-bit (RV64) address spaces. This flexibility enables the architecture to cater to different target platforms and application requirements. Moreover, the RISC-V ISA adheres to a load-store architecture, where data transfer between memory and registers can only occur through specific load and store instructions. This approach reduces instruction complexity and enhances pipelining, leading to improved overall performance.

The instruction set of RISC-V is designed to be orthogonal, meaning that instructions have a consistent format and can operate on any register. This orthogonality simplifies the instruction decoding process and enables more efficient code generation. Furthermore, RISC-V reduces the number of addressing modes, which contributes to the overall simplicity and ease of implementation.

2.1.3 Advantages Over Other Architectures

RISC-V is an open-source ISA that has gained popularity due to its several advantages over other architectures like RISC, MIPS, and ARM. Here is an extensive list of the key benefits of RISC-V:

- Open Source: One of the most significant advantages of RISC-V is its royalty-free open-source nature, meaning anyone can access, use, modify, and implement it without any licensing fees or restrictions. This openness has encouraged widespread adoption and collaboration in both academic and commercial settings.
- Customizability: This is another of RISC-V's key highlights. Designers can add optional instruction extensions tailored to specific application needs, leading to more energy-efficient and performance-optimized designs.
- Modularity: RISC-V's modular design allows users to select the exact set of instructions they need for their specific application, making it

highly customizable and efficient.

- Scalability: RISC-V offers different instruction widths like 32-bit and 64-bit. This feature provides flexibility for various applications and target platforms, making it suitable for both low-power embedded systems and high-performance servers.

- Simplicity and Elegance: RISC-V follows the Reduced Instruction Set Computer (RISC) philosophy, which aims to keep the instruction set simple and easy to decode, execute, and pipeline. This simplicity leads to efficient and power-conscious hardware implementations and straightforward compiler design.

- Flexibility for Specialized Instructions: RISC-V provides a mechanism for adding application-specific instructions, which can significantly accelerate specific tasks and improve overall performance for specialized workloads.

- Standardization and Ecosystem: RISC-V is developed and maintained by the RISC-V Foundation, which ensures a well-defined and standardized ISA specification. This has led to the growth of a rich ecosystem of tools, compilers, simulators, and development boards, making it easier for developers to work with RISC-V.

- Research and Academia: RISC-V's openness and simplicity have made it a popular choice for research and academic projects. It enables researchers to explore novel architectural ideas and experiment with new designs without the limitations of proprietary architectures.

- Reduced Vendor Lock-In: Since RISC-V is an open standard, it reduces the risk of vendor lock-in that might occur with proprietary architectures. Organizations can switch between different RISC-V implementations without significant architectural changes.

- Community-driven Innovation: The open nature of RISC-V encourages a collaborative and diverse community of developers, leading to continuous innovation and improvement of the architecture.

- Security Considerations: RISC-V's simplicity and modularity make it easier to implement security features and extensions, allowing designers to incorporate robust security mechanisms into their processors.

Overall, RISC-V's open-source nature, flexibility, and simplicity have made it a compelling choice for a wide range of applications, from low-power Internet of Things (IoT) devices to high-performance computing systems.

Its growing popularity and active community support have solidified its position as a competitive alternative to other established architectures like RISC, MIPS, and ARM.

2.1.4 RISC-V Standard Extensions

RISC-V offers a modular approach, allowing designers to tailor the processor to specific needs while maintaining compatibility with a core set of instructions. The architecture separates its features into 'unprivileged' and 'privileged' specifications. The unprivileged specification outlines the base integer instruction set and standard extensions like floating-point operations, atomic instructions, and others. These can be implemented or omitted depending on design goals such as energy efficiency, performance, or area optimization. On the other hand, the privileged specification details advanced functionalities, essential for system-level tasks like interrupt handling and virtual memory management, which are accessible only in privileged operating modes. This segregation into privileged and unprivileged instructions allows for more secure and efficient system designs. For comprehensive details on these specifications and extensions, RISC-V International maintains an official website, and interested readers can refer to it for more information [14].

Within the RISC-V ecosystem, the term 'ratified' holds special significance. Ratified extensions have undergone rigorous review and testing and have received official approval from RISC-V International. They are considered stable and are recommended for widespread adoption, ensuring that any implementations based on them will be both compatible and maintainable over time.

Among the notable ratified extensions are:

- RV32I/RV64I - Base Integer Instruction Set: The RV32I and RV64I extensions are the base integer instruction sets for 32-bit and 64-bit RISC-V architectures, respectively. These are the basic integer instruction sets, which include essential operations for integer arithmetic, logical operations, and control flow.
- RV32E - Base Integer Instruction Set (embedded): The RV32E extension is a variant of the base integer instruction set that is designed for embedded systems with limited resources. It has a smaller register file (16 registers instead of 32) and a smaller instruction set.
- M - Standard Extension for Integer Multiplication and Division: These

extensions add hardware support for integer multiplication and division instructions.

- A - Standard Extension for Atomic Instructions. The A extension in the RISC-V architecture adds atomic instructions designed to facilitate the construction of multi-threaded applications. This extension is particularly important for implementing synchronization primitives like locks and semaphores in a multithreaded environment. The atomic operations read, modify, and write back a memory location atomically, ensuring that no other instruction can access the given memory location during the atomic operation.

- C - Standard extension for Compressed Instructions: The C extension adds support for 16-bit compressed instructions, which can significantly reduce code size while maintaining performance. These compressed instructions can be used in place of their corresponding 32-bit instructions without any loss in functionality.

- F/D - Standard Extensions for Single-Precision and Double-Precision Floating-Point: These extensions add support for single-precision and double-precision floating-point arithmetic and manipulation.

- Zfinx/Zdinx - Standard Extensions for Floating-Point in Integer Registers: Theses extensions provide similar instructions to those in the standard floating-point F and D extensions but which operate on the integer `x` registers instead of the floating-point `f` registers.

- Zicsr - The Zicsr extension stands for Control and Status Registers and is part of the standard RISC-V instruction set extensions. This extension adds instructions for reading and writing Control and Status Registers (CSRs). These are special-purpose registers used to configure or query various system settings or statuses. In the context of an operating system, this extension is crucial for tasks like process isolation, interrupt handling, and general system configuration.

- Zifence - The Zifence extension stands for Instruction-FENCE and provides a way to ensure instruction stream synchronization. The `fence.i` instruction is often used in this extension. In multiprocessor systems, when software updates an instruction in memory (such as self-modifying code or dynamic code loading), `fence.i` guarantees that subsequent instruction fetches see the new value. This is crucial for ensuring that all cores in a system have a consistent view of the instruction memory.

- G - Shorthand for the base and above extensions: This is not an extension itself but rather a shorthand notation to indicate that all of the above extensions (M, A, F, D) are included.

Each set of extensions can be added independently, allowing for customizing the ISA to suit specific application needs, and they are backward compatible with the base integer instruction sets (RV32I/RV64I). By providing this modular and extensible approach, RISC-V offers flexibility and scalability while maintaining simplicity and ease of implementation.

Among the extensions that have not yet been ratified, the RV128I ISA stands out as particularly noteworthy. Designed as a future addition to the RISC-V architecture, RV128I aims to extend the integer support to 128 bits. This would mark a significant departure from the existing RV32I and RV64I ISAs, which provide 32-bit and 64-bit integer support, respectively.

However, it is crucial to understand that RV128I is largely theoretical at this stage and not yet actively implemented or standardized. The extension faces numerous complexities, such as the need for vastly larger memory subsystems and modifications to the instruction format to handle 128-bit immediates or addresses. Due to these challenges, RV128I remains unratified.

2.1.5 RISC-V Processors Available in the Public Domain

There is a wealth of publicly available RISC-V processors that offer open-source RTL and cater to low-power requirements, allowing for free usage, modification, and distribution in compliance with their respective licenses.

These processors can be categorized into two main groups:

1. General-Purpose Processors: Designed with a broad application scope, these processors are versatile and can serve various use cases. Notable examples include VexRiscv and SweRV.

2. Application-Specific Processors: Tailored for specific applications, these processors address the unique demands of particular domains, such as IoT. Examples in this category include Ri5cy, Zero-riscy, and Rocket, which are optimized to meet the requirements of IoT devices.

Some of the well-known RISC-V processors available in the public domain include:

VexRiscv [15] is a 32-bit configurable RISC-V soft-core processor written in SpinalHDL and developed by C. Papon in 2019. It is designed for

FPGA implementation and intended for use in embedded systems and supports various configurations for custom instruction sets and peripheral interfaces. Due to its low power consumption and high performance, it is particularly well suited for FPGA-based embedded applications, such as real-time control and data processing. It has a pipeline with a configurable number of stages, from 2 to 5 stages, and provides support for the RV32I[M][F][C] instruction set.

SweRV [16] is a 32-bit, 9-stage, dual-issue, superscalar, mostly in-order pipeline with some out-of-order execution capability that supports the RV32IMC_Zicsr_Zifence ISA. The SweRV processor was developed by Western Digital and it is intended for a wide range of applications, including storage devices, embedded systems, and data centers.

Ri5cy and **Zero-riscy** processors are two open-source low-power RISC-V processors designed specifically for embedded systems and IoT applications, both written in SystemVerilog by the PULP team from ETH Zurich.

Ri5cy [17] is a 32-bit, 4-stage, in-order processor with a small and efficient microarchitecture. Ri5cy aims to provide a balance between performance and power efficiency, making it suitable for resource-constrained embedded systems. It supports the RV32IMC[F] instruction set, which includes the integer, multiplication, and compressed instruction set extensions. This processor has been maintained by the PULP platform team until February 2020, when it was contributed to OpenHW Group and developed further under the codename **CV32E40P** [18]. It supports the RV32IMC[F][Zfinx] instruction set.

Zero-riscy and **Micro-riscy** [19] were designed to be a simplified version of Ri5cy to demonstrate how small a RISC-V CPU core could actually be. They are intended for ultra-low-power applications where power consumption is critical, such as IoT devices and wearable electronics. Zero-riscy features a 2-stage, in-order pipeline with a small footprint and optimized power efficiency. It supports the RV32IMC instruction set and includes various power-saving techniques, such as clock gating and dynamic voltage and frequency scaling (DVFS). This processor is now maintained and developed further by lowRISC, a not-for-profit company based in Cambridge, and the processor is now known by the codename of **Ibex**. And Micro-riscy differs from Zero-riscy in that it supports the embedded RV32EC instruction set and drops the integer multiplication and division instructions.

Rocket [7] is an open-source RISC-V processor written in Chisel and developed at the University of California, Berkeley. A 5-stage, in-order scalar core that implements the RV32G and RV64G ISA, it has a highly configurable and extensible design that serves as a platform for research and development in education and industry projects. Its open-source nature and flexible design make it a popular choice for exploring new ideas in computer architecture, prototyping novel processor features, and building custom processor designs tailored to specific applications or domains.

It is worth mentioning that among the available processors, we have carefully selected a set of reference processors for benchmarking and comparison with our own design. The chosen processors include Ri5cy, Micro-riscy, Zero-riscy, and CV32E40P, all of which are compatible with the PULPino SoC and can be seamlessly integrated into the platform. Our objective is to design a processor that shares the same pinout configuration as these references, allowing us to integrate it into the PULPino SoC and conduct a comprehensive performance evaluation. By running our specific target application of measuring CO_2 concentrations in the ambient environment, we aim to analyze the efficiency and effectiveness of our processor compared to the established reference designs.

2.1.6 Proprietary RISC-V Processors

While the RISC-V ISA is open-source and encourages collaboration and innovation, it also allows for the development of proprietary RISC-V processors. Proprietary processors based on the RISC-V ISA offer companies the flexibility to design customized solutions tailored to their specific requirements for internal use or specific markets without revealing the full details of their implementations to the public. The design details of these processors are often kept confidential and not publicly disclosed, providing a level of intellectual property protection and allowing companies to differentiate their products in the market.

Some examples of well-known proprietary RISC-V processors include:

SiFive Core IP (RISC-V Coreplexes): SiFive®, a leading RISC-V processor IP provider, offers various proprietary RISC-V processor cores designed to cater to different applications. Examples include the E21 series (32-bit cores) and E51 series (64-bit cores). These cores can be customized to meet specific performance, area, and power requirements for applications such as AI accelerators, IoT devices, and high-performance computing [20].

AndesCore: Andes Technology provides a range of proprietary RISC-V cores known as AndesCore™. Their offerings include the N25 and N45 series (32-bit cores) and the NX25 and NX45 series (64-bit cores). These cores are designed for applications in automotive, industrial, and IoT domains [21].

Codasip IP cores: Codasip specializes in customizable processor IP cores based on the RISC-V ISA. Their proprietary cores, such as Bk3 and Bk5, allow customers to tailor the processor to their specific application needs, offering a balance between performance and power efficiency [22].

In conclusion, proprietary RISC-V processors offer companies the ability to design customized solutions tailored to their specific needs while capitalizing on the benefits of the open-source RISC-V ISA. By keeping certain design details confidential, companies can differentiate their products and cater to a diverse range of applications in various industries.

2.2 Review of Energy-efficient Processor Design Techniques

Energy efficiency has become a paramount concern in modern processor design, driven by the escalating demand for low-power computing solutions in diverse applications like mobile devices, IoT devices, and battery-operated systems. In this section, we explore a selection of key energy-efficient processor design techniques, representing the current state-of-the-art in the field. These techniques have been developed and embraced to tackle the pressing need for power-conscious computing. It is important to note that the techniques presented here are not an exhaustive list of all the methods applied in our processor design; rather, they reflect the cutting-edge approaches available today to achieve optimal energy efficiency while maintaining high-performance computing capabilities. As the field continuously evolves, novel techniques and innovations will continue to shape the landscape of energy-efficient processor design, enabling further progress in the realm of power-conscious computing.

2.2.1 Power Analysis and Energy Efficiency Considerations

To understand power consumption in Complementary Metal-Oxide-Semiconductor (CMOS) circuits, it is crucial to first detail the parameters influencing it. This will allow us to pinpoint potential areas for power reduction. It is worth considering how these parameters might vary based on the specific application in question. In the standard analysis of digital integrated circuits, as discussed in [23], power dissipation in a CMOS circuit arises from two main components: static power P_{sta} and dynamic power P_{dyn}.

Dynamic power dissipation occurs due to the charging and discharging of the output capacitance C_L for each CMOS gate in the circuit. This capacitance encompasses the drain diffusion capacitances of the NMOS and PMOS transistors that constitute the gate, the capacitance from the connecting wires, and the input capacitance of the gates that receive the output (fan-out gates). A single switching cycle, which includes both low-to-high and high-to-low transitions, consumes a consistent energy amount sourced from the supply voltage V, amounting to $C_L \cdot V^2$. Thus, for accurate power consumption calculations, the frequency with which a circuit's logic gates switch must be factored in. The switching activity of a complex circuit inherently relies on the characteristics and statistics of the input signals. This consideration can be encapsulated in the formula for dynamic power consumption as:

$$P_{dyn} = \alpha \cdot C_T \cdot V^2 \cdot f_{clk} \tag{2.1}$$

where α is the probabilistic switching activity factor, C_T is the total capacitance of the circuit based on the number of transistors, f_{clk} is the maximum possible event rate of the inputs (which is often the clock rate), and V is the supply voltage.

The static, often referred to as steady-state, power dissipation in CMOS circuits is primarily attributed to the static or leakage current, denoted as I_{leak}. This current flows between the supply rails even when there is no switching activity taking place. It can be mathematically represented by the relation:

$$P_{sta} = V \cdot I_{leak} \tag{2.2}$$

In an ideal scenario, the static current of a CMOS gate should be zero. This stems from the operational behavior of CMOS gates where PMOS and NMOS devices are never simultaneously active during steady-state operations. However, in the real-world setting, a leakage current exists. It primarily flows due to the reverse-biased diode junctions of the transistors. Although this leakage current has historically been a minor contributor to power dissipation, its significance is growing with the evolution of semiconductor technology. As chips continue to scale down in size, following the trajectory of Moore's law, this leakage current becomes a more pronounced concern. The increasing influence of the leakage current introduces additional challenges to maintaining energy efficiency, especially as semiconductor devices move towards smaller technology nodes.

An important consideration in our analysis is the omission of direct-path power consumption. This type of consumption arises during the dynamic transition of transistors. CMOS gates are driven by input signals with a finite slope that produces a temporary direct short-circuit current path formed between the supply rails. During this transient phase of switching, both NMOS and PMOS transistors might conduct simultaneously, leading to a surge in power consumption, albeit for a very brief duration.

This short-circuit power consumption can be effectively mitigated at the transistor's physical design level. One established approach to address this involves matching the rise and fall times of the input and output signals, as explained in [24]. While making the input and output rise times of a gate identical might not render the best results for every individual gate, it is instrumental in keeping the overall short-circuit current within permissible bounds. Furthermore, as semiconductor technology has advanced, the short-circuit current has shown a tendency to reduce with decreasing supply voltage. This trend implies that in modern deep submicron technologies, the relative significance of short-circuit power dissipation is diminishing.

In summary, the comprehensive power consumption of a CMOS circuit encompasses several components. It can be formally captured by the equation:

$$P_{tot} = P_{dyn} + P_{sta} = \alpha \cdot C_T \cdot V^2 \cdot f_{clk} + V \cdot I_{leak} \tag{2.3}$$

Since we are interested in how to enhance the energy efficiency of our RISC-V processor, we turn our attention to a processor-centric approach presented in [25]. This source introduces an energy efficiency metric, termed G or *greenness*. Beyond encapsulating the parameters highlighted in the previously discussed power consumption equations, G integrates another vital computing performance measure: operations per cycle (OPC). Simply put, greenness represents the number of operations executed every second for each consumed unit of electrical power:

$$G = \frac{Op}{T}\frac{1}{P} \tag{2.4}$$

Therefore G is expressed often expressed in operations per second per Watt or one of its multiples. For instance, the well-known Green500 list [26], the ranking of the most energy-efficient supercomputers in the world, uses GFLOPS/W, *i.e.* the number of billions of floating-point operations per second that can be executed with one Watt of electrical power.

The number of operations per time unit can be more appropriately expressed as operations per cycle (OPC) as follows:

$$\frac{Op}{T} = OPC \cdot fclk \tag{2.5}$$

In the context of RISC-V we can conveniently use OPC as a synonym for IPC (instructions per cycle). This is because the instruction set is designed such that all instructions share the same latency. This consistent latency ensures a uniform basis for comparing processors across various benchmark applications. With this in mind, and by combining Equations 2.3, 2.4 and 2.5 we can reformulate G as:

$$G = \frac{IPC}{\alpha \cdot C_T \cdot V^2 + \frac{V \cdot I_{leak}}{f_{clk}}} \tag{2.6}$$

From the above equation, it is evident that only IPC and clock frequency exhibit a direct relationship with enhanced energy efficiency. Interestingly, though it might seem counter-intuitive, higher clock frequencies can lead to more energy-efficient designs. This is because quicker instruction execution reduces the impact of static power. Such a correlation was previously highlighted in [27], but its significance is often overlooked or misinterpreted.

By following, we will categorize a range of energy-efficient design strategies based on the individual parameters highlighted in Equation 2.6. By understanding the influence and role of each parameter, we aim to explore how its strategic manipulation can enhance the energy efficiency of processor designs. This approach will not only provide a systematic framework for assessing energy-efficient techniques but will also offer insights into the interdependencies between these parameters. The main goal is to find out how alterations in one aspect can potentially benefit, or conversely, impact the overall energy performance of a processor, allowing designers to make more informed decisions in their optimization efforts.

Techniques to Increase IPC

Instruction Level Parallelism (ILP): One of the fundamental techniques to enhance processor performance while maintaining energy efficiency is exploiting Instruction Level Parallelism. This involves the execution of multiple instructions in parallel, maximizing the utilization of functional units and execution resources. Simply put, it is like having multiple tasks being worked on simultaneously rather than in a sequence. The direct advantage of this is an increase in the number of instructions executed per cycle, leading to faster computations using the same amount of energy.

Achieving high ILP can drastically reduce the time processors spend idling or waiting for previous instructions to complete. When processors finish tasks faster, they can enter low-power states more quickly or tackle other tasks more efficiently, leading to energy savings. A review of strategies that are commonly used to achieve ILP can be found in [28] and [29]. Some of these strategies include:

Pipelining: This involves breaking instructions into multiple stages, with each stage being processed in parallel. It is akin to an assembly line in a factory where different parts of a product are assembled simultaneously. This technique augments not only the ILP but also facilitates an increase in the clock frequency. Consequently, it is elaborated upon in greater depth within the clock frequency parameter category.

This technique also allows increasing the clock frequency and therefore it is explained in more detail in the next parameter category.

Superscalar Execution: Here, multiple execution units are used, allowing more than one instruction to be executed in parallel. A processor can dispatch multiple instructions to appropriate execution units in one cycle.

Out-of-Order Execution: Processors equipped with this capability can execute instructions as soon as their inputs are ready, rather than strictly adhering to their original program order. This flexibility means that even if one instruction is stalled, others can proceed, maximizing the use of available resources.

Speculative Execution: To avoid stalls, processors guess or speculate the outcome of conditional branches and execute instructions based on those guesses. If the speculation is correct, time is saved; if not, the actions are rolled back.

Loop Unrolling: This is a vital optimization technique where loops in a program are expanded to execute multiple iterations in a single loop cycle. By reducing the overhead of loop control instructions, processors can execute more useful instructions in parallel. For instance, a loop set to execute four times could be unrolled to execute all iterations in just one or two cycles, accelerating the computation and benefiting from increased parallelism.

With the integration of ILP and these accompanying strategies, microprocessor designers can achieve significant advancements in both performance and energy efficiency, setting new standards in computational capabilities.

Application-Specific Optimization: For specialized applications with unique requirements, application-specific optimizations are employed to tailor the processor's design to the specific workload. These optimizations exploit the idea that not all tasks are created equal. By molding a processor to the peculiarities of a particular job, one can shave off superfluous functionalities, reducing circuit complexity, and power wastage. A microprocessor, thus tailored, can perform its intended task more swiftly, increasing the number of instructions executed per cycle and saving energy. There exist several prominent methodologies that emphasize application-specific optimization:

<u>Application-Specific Integrated Circuits (ASICs)</u>: These are non-reconfigurable hardware entities crafted explicitly for specialized tasks. The precision and specificity of ASICs eliminate superfluous circuitry, subsequently leading to reduced parasitic capacitance and minimized dynamic power dissipation [30]. For instance, in the domain of cryptographic operations, especially in proof-of-work algorithms central to blockchain validation, ASICs are valued for their computational throughput and energy efficiency.

<u>Hardware Accelerators or Co-processors Attached to the Bus</u>: These computational units provide specialized hardware for specific tasks, enabling the main processor to offload complex operations and reduce its power consumption. These co-processors can be dynamically activated when needed and deactivated when not in use, optimizing overall energy efficiency. An illustrative example is Google's Tensor Processing Units (TPUs). Specifically designed for machine learning computations, TPUs demonstrate superior computational efficiency in their domain when placed side by side against general-purpose CPUs or GPUs, leading to energy savings. An extensive review of hardware accelerators, their classification, trends, and challenges can be found in [31].

<u>Instruction Set Customization</u>: By customizing the instruction set architecture for particular tasks, a processor can execute code more efficiently. Processors that implement domain-specific work by setting a dedicated instruction set are known as ASIPs (Application-Specific Instruction Set Processors). A survey on the design process of ASIPs and the generation of the instruction set for a particular application is detailed in [32]. A case in point is the ARM's NEON technology. This advanced SIMD (Single Instruction, Multiple Data) architecture is crafted to optimize tasks inherent to media processing, providing faster, energy-efficient operations.

<u>Custom Instructions</u>: Custom instructions are tailored instructions added to the processor's instruction set to accelerate specific functions commonly used in the target application. By executing these functions

more efficiently, custom instructions contribute to energy savings and performance improvement. Custom instructions, along with the required hardware logic to execute them, can be conceptualized as custom functional units or tightly-coupled co-processors. In contrast, hardware accelerators can be viewed as loosely-coupled co-processors.

Compiler Optimizations: Compiler optimizations play a pivotal role in enhancing the performance of software applications without necessitating changes to the source code. By translating high-level code into machine code, compilers have the crucial task of making this transformation as efficient as possible. This translation, when optimized, can lead to significant improvements in execution speed, power consumption, and memory usage. For instance, compilers can take advantage of SIMD (Single Instruction, Multiple Data) instructions available on a processor to parallelize operations and boost performance. In this seminal paper [33], the authors introduce an innovative methodology tailored for contemporary compilers that target general-purpose architectures.

A few more concrete cases that illustrate this theme further can be found in drones, for example, whose processors, often ASICs, are strictly optimized for flight control and real-time video capture, ensuring both extended flight durations and seamless video streams. Similarly, gaming consoles include processors fine-tuned for the intricacies of graphics rendering and real-time processing, thus facilitating an immersive gaming experience characterized by fluidity and high frame rates. Lastly, in the domain of the IoT, edge devices like security cameras employ processors optimized for continuous surveillance, achieving proficient motion detection with a minimized energy footprint.

In essence, application-specific optimizations are refining the microprocessor design process, aligning architecture with specific workloads. This alignment ensures that processors not only perform tasks more quickly but also consume lesser energy, a dual advantage in our increasingly digital world.

Techniques to Increase the Clock Frequency

Pipeline Optimization: In microprocessor design, pipelining is a primary technique to enhance performance by allowing multiple instructions to be processed concurrently at different stages. One of the main goals of pipelining is to increase the clock frequency, thus improving the throughput of the processor. The underlying concept of pipelining resembles an assembly line in a factory where different stages of a product are handled simultaneously but in different phases of production. By breaking down instruction processing into smaller tasks (stages), each task can be completed in a shorter clock cycle, thereby potentially increasing the clock rate. At the same time, with multiple instructions in different stages of execution, the CPU can handle more instructions at once, increasing the overall instruction throughput.

However, designing an efficient pipeline is non-trivial. Ensuring that instructions flow seamlessly through the pipeline stages demands careful consideration of potential hazards. While extending the pipeline by adding more stages can increase the clock frequency, a longer pipeline means more stages where data dependencies (data hazards) can cause pipeline stalls. These stalls occur when subsequent instructions are dependent on the results of previous ones, causing the pipeline to wait. Also, as we increase the number of stages, the overhead related to managing the pipeline, such as checking for data hazards or control hazards, also increases. The ideal pipeline depth for a microprocessor, both from theoretical and simulation perspectives, is meticulously examined in [34]. In response to these inherent challenges, advanced techniques have been developed:

Instruction Reordering: Compilers and hardware can sometimes reorder instructions to ensure that dependent instructions are spaced apart, giving the first instruction enough time to produce results before the second instruction needs them.

Pipeline Hazard Handling: Techniques such as data forwarding (or hazard forwarding) are used to reduce data hazards. In data forwarding, the result is forwarded directly from one pipeline stage to another without waiting for it to be written back to a register.

Branch Prediction: To mitigate control hazards, modern processors employ branch predictors. They guess the outcome of a branch based on historical data and patterns, allowing the processor to fetch and execute instructions from the predicted path. Modern branch predictors use sophisticated algorithms to achieve high accuracy rates, ensuring that the pipeline remains filled with useful instructions. However, no predictor is perfect. When a prediction is wrong, the processor needs to flush the pipeline of the wrongly fetched instructions and start fetching from the correct path. This flushing process results in a pipeline "bubble" or stall, wasting cycles. In processors with very long pipelines, the cost of a misprediction is even higher because more stages are affected, making an efficient branch predictor crucial for sustaining performance. In Smith’s seminal work [35], a variety of branch prediction techniques are comprehensively discussed. However, the quest for increased accuracy persists. Illustrative of this ongoing pursuit is the innovative branch predictor detailed in [36], which exploits neural learning methodologies.

In conclusion, properly optimized pipelines significantly diminish the occurrence of pipeline stalls, ensuring the processor's resources are utilized to their maximum potential. By minimizing stalls and maximizing the concurrent processing of instructions, the energy expended per instruction is reduced, leading to more energy-efficient microprocessor designs.

Techniques to Decrease the Supply Voltage V

Dynamic Voltage and Frequency Scaling (DVFS): Dynamic Voltage and Frequency Scaling enables a processor to adapt its voltage and frequency in real-time based on the computational demands of the task at hand. The essential concept lies in providing just enough power to meet the workload requirements, avoiding excess consumption. By dynamically scaling voltage and frequency, the processor can match its performance to the application's requirements, leading to significant energy savings. During low-demand tasks, the processor can run at lower frequencies and voltages, reducing both dynamic and static power. A detailed exploration of this approach and its application within the demanding environment of data centers is provided in [37].

Considering again the relationship formulated in Equation 2.6, it is clear that energy efficiency G is inversely proportional to the square of the operating voltage and the frequency of operation. By reducing the voltage, even slightly, we can achieve a significant increase in energy efficiency. Lowering the voltage reduces the switching power in the transistors, leading to less energy use. However, this comes with a trade-off in processing speed, as lower voltage typically means slower transistor switching and thus lower operational frequency. Voltage and frequency are closely intertwined in a processor's operation. Lowering the voltage requires lowering the frequency to maintain stable operation while raising the voltage allows for higher frequency (and thus performance). This interplay must be managed carefully to achieve the desired balance between performance and power consumption.

Power Gating: Power gating refers to the process of completely turning off the power to specific areas of a microprocessor or an integrated circuit (IC) when they are not in use. This is done by using special transistors known as power switches or power gates to disconnect the power supply to certain sections of the chip. Unlike Dynamic Voltage and Frequency Scaling (DVFS), where the voltage is scaled down, power gating cuts off the power entirely to non-operating parts, also bringing the benefit of reducing leakage power to almost zero. An exploration of the potential of architectural techniques to achieve these reductions, specifically through the power gating of processor execution units, can be found in [38].

This technique is particularly useful in designs where various functional blocks or modules have different operating schedules, and not all of them are required to be active simultaneously. By shutting down unused sections, power gating ensures that only the parts of the chip actively performing tasks are powered, leading to significant energy savings.

However, power gating does bring its challenges. Turning off a section of the chip means that it will take some time to turn it back on when needed, introducing latency. Proper management of the transitions between on and off states is essential to minimize this latency and ensure that the power-gated sections are ready when needed. This requires careful coordination between hardware and software, considering both the performance requirements of the application and the power-saving goals. Another concern in power gating is managing the state of the power-gated sections. When power is completely cut off, the state of the power-gated area (e.g., the values in its registers) may be lost. Strategies to preserve or restore this state upon reactivation must be part of the power gating design.

Power gating has found extensive applications in low-power devices such as mobile phones, tablets, and IoT devices, where battery life is a critical concern. By selectively powering down parts of the chip when they are not required, these devices can offer extended battery life while still delivering the performance users expect.

In summary, power gating represents an effective power management strategy by completely turning off power to unused parts of a chip. Its implementation must be carefully managed to balance the benefits of reduced power consumption with considerations for performance, latency, and state preservation.

Techniques to Decrease the Switching Activity Factor α

Clock Gating: Clock gating works by selectively disabling the clock signal to certain parts of the circuit when they are not in use. In a typical digital circuit, the clock signal is continuously toggling, driving the sequential elements of the design, such as flip-flops and registers. This constant toggling contributes to dynamic power consumption, even when certain parts of the circuit are idle or not in use. By implementing clock gating, the clock signal can be disabled to specific areas of the circuit, preventing them from toggling and thereby reducing dynamic power consumption. This is typically achieved through the addition of control logic that observes the operational conditions and appropriately gates the clock signal.

Consider a functional unit within a processor that is only active during specific operations. By employing clock gating, the clock to this unit can be disabled when it is not required, significantly reducing power consumption. The challenge in implementing clock gating is in determining when it is safe to gate the clock without impacting the correct functionality of the system. This requires careful analysis and possibly additional logic to ensure that gating the clock does not lead to incorrect behavior.

The benefits of clock gating are most pronounced in systems where there are considerable idle periods or functions that are not continuously utilized. It is a key technique in modern processors to manage energy efficiency, especially in battery-operated devices where power saving is crucial. In complex systems, clock gating must be handled with caution, as incorrect implementation might lead to synchronization issues or other functional problems. The benefits in terms of energy efficiency need to be balanced with the additional complexity and potential risk to the system's robustness. A comprehensive review of the various clock gating techniques that can be used to optimize power in VLSI circuits at Register-Transfer Level (RTL) level can be found in [39].

In the context of Equation 2.6, clock gating directly affects the switching activity factor by reducing unnecessary transitions, which in turn lowers the dynamic power component of the overall energy consumption. Therefore, it is a vital technique in the toolkit of a designer aiming to improve the energy efficiency of processor designs.

Voltage-Frequency Islands (VFIs): Voltage-Frequency Islands are an advanced power management technique that complements methods such as Dynamic Voltage and Frequency Scaling and clock gating to further reduce power consumption, including the switching activity factor. In a complex System-on-Chip (SoC) or processor, different functional units may have different performance requirements at various times. While some parts of the chip may be heavily utilized, others might be idle or only lightly loaded. VFI takes advantage of these varying requirements by dividing the chip into separate regions or "islands," each of which can operate at a different voltage and frequency.

By recognizing that not all parts of the chip need to operate at the maximum frequency and voltage at all times, VFI enables each island to operate at an optimal voltage and frequency for its current workload. This can significantly reduce both dynamic and static power consumption. Operating an island at a lower frequency reduces the clock switching rate, directly cutting down the switching activity factor. Combined with techniques like clock gating, this can lead to substantial energy savings.

Furthermore, operating at lower voltages reduces not only dynamic power but also leakage power, as leakage is often strongly dependent on the operating voltage. VFI allows fine-grained control over different parts of the chip, providing more flexibility in power management. For example, a processor could have an island dedicated to real-time tasks running at a high frequency, while other islands handling background tasks could run at lower frequencies.

However, implementing VFI requires careful design and analysis. Each island must be properly isolated, and data communication between islands operating at different frequencies must be handled cautiously. Voltage level shifting and synchronization mechanisms may be required. Despite these complexities in implementation, VFI is commonly used in multi-core processors and SoCs where different cores or functional units may have disparate performance needs. This allows each core or unit to operate at an independently optimized voltage and frequency, adapting to the dynamic demands of various applications.

In [40], the authors present a design methodology for partitioning a Network-on-Chip (NoC) architecture into multiple VFIs and assigning supply and threshold voltage levels to each VFI. The method demonstrates the effectiveness of the approach in reducing the overall system energy consumption.

Techniques to Decrease Processor's Total Capacitance and Leakage Current

Removing Logic Associated with Unused Instructions: Removing logic associated with unused instructions can lead to notable improvements in energy efficiency in processors. When certain instructions are not required for a specific application, the logic associated with those instructions can be completely removed or shaved off. This removal has two primary benefits:

1. Dynamic Power Reduction: By eliminating unnecessary logic gates and interconnections, the total capacitance of the chip is reduced. Since dynamic power consumption is directly related to the charging and discharging of capacitive structures within the circuit, this reduction in capacitance

leads to lower dynamic power consumption.

2. Leakage Power Reduction: In addition to dynamic power, the removal of unused logic also reduces leakage current. This reduction is achieved not through power gating but by entirely eliminating the logic itself. As leakage current is the small amount of current that flows even when a transistor is turned off, removing unused transistors and associated structures means that there are fewer sites for leakage to occur, further reducing the overall power consumption.

The strategic elimination of unnecessary logic based on the specific needs of an application is an example of application-specific optimization. By tailoring the processor design to the requirements of the task at hand, both dynamic and leakage power can be minimized, contributing to the overall energy efficiency of the processor. This approach reflects a deeper integration between hardware design and software requirements, emphasizing the need for a holistic view of system optimization.

One illustrative example of this approach is the utilization of a One-Instruction Set Computer (OISC)-based multicore processor designed specifically for energy-efficient streaming data processing. This unique processor consists of application-independent tiny cores and application-dependent optimizable inter-core communications, which effectively execute applications on extensive streams of data in a pipeline fashion. The potential of OISC for processing encrypted data is further investigated by the authors in [41].

Cache Design and Memory Hierarchies: Cache design and memory hierarchies play a crucial role in optimizing the power consumption of modern processors. Memory hierarchies are established to exploit the temporal and spatial locality in program behavior. By keeping frequently accessed data close to the processor, memory hierarchies reduce the need to access off-chip larger, slower memories, which usually have higher capacitance. Since the capacitance associated with memory access is directly proportional to dynamic power, accessing a smaller and closer cache results in energy savings.

Cache size and configuration can be optimized to meet the specific requirements of different applications. Reducing the cache size will reduce the total circuit capacitance, hence lowering the dynamic power consumption. However, there must be a careful balance to ensure that the reduced size does not adversely affect the performance by increasing the miss rate. Techniques like way-prediction allow the processor to access only the relevant parts of a set-associative cache, thus reducing the effective capacitance on a cache access and also reducing energy consumption [42]. By selectively accessing the cache, the switching activity is minimized, further reducing dynamic power.

In some cases, using a scratchpad memory can be more energy-efficient than using traditional caches [43]. Scratchpad memory is a programmer-managed memory structure, where programmers explicitly control what data is placed in the scratchpad. While this requires more effort, it can lead to very efficient usage of the memory space, which can be more power-efficient in certain applications. In contrast, caches are managed by hardware and work transparently to the programmer. While they offer ease of use, caches may include inefficiencies such as unnecessary data being loaded and power consumed due to cache coherence mechanisms. In certain energy-critical applications, a well-managed scratchpad can offer more precise control and can be more energy-efficient than a cache. However, this comes at the cost of increased programming complexity.

In conclusion, energy-efficient processor design is essential for meeting the ever-increasing demand for power-conscious computing solutions. The techniques mentioned above, along with continuous advancements in semiconductor technology, play a crucial role in achieving optimal energy efficiency while maintaining high-performance computing capabilities. By leveraging these techniques intelligently and considering the specific requirements of target applications, designers can develop processors that strike an optimal balance between performance and energy efficiency, driving the progress of power-conscious computing in diverse domains.

2.3 Review of Optical Gas Sensors and their Applications

The development and widespread adoption of gas sensors have ushered in a new era of environmental monitoring and safety across diverse industries [44]–[48]. Among the myriad sensing technologies, optical gas sensors have garnered significant attention due to their exceptional accuracy, sensitivity, and selectivity. These sensors harness the principles of light absorption to quantify the concentration of specific gases, opening avenues for various applications in industrial processes, environmental monitoring, and healthcare [49]–[51].

This section presents a comprehensive review of optical gas sensors and their applications. Through a detailed exploration of the underlying principles and the diverse sensor types available, we aim to shed light on the cutting-edge advancements that drive the efficacy of optical gas sensing technologies in a broad spectrum of real-world scenarios.

2.3.1 Working Principle (Beer-Lambert Law)

The Beer-Lambert law, also known as the Beer-Lambert-Bouguer law, is a fundamental principle in optical gas sensing that enables the quantification of gas concentration based on the attenuation of light passing through a gas sample [52]–[56]. The law is expressed as:

$$A = \alpha \cdot L \cdot c \tag{2.7}$$

Where:

- A represents the absorbance of light by the gas sample. It measures how much incident light is absorbed when it travels through a medium.
- α denotes the molar absorptivity, a constant specific to the gas being measured, indicating its ability to absorb light at a particular wavelength.
- c stands for the concentration of the gas in the sample in ppm.
- L refers to the optical path length, which represents the distance the light travels through the gas.

The Beer-Lambert law provides a direct relationship between the absorbance and the concentration of a gas, allowing accurate determination of gas concentrations through spectrophotometric measurements. In the context of CO_2 concentration measurement, the law plays a pivotal role in the operation of optical CO_2 sensors.

We need to introduce the concept of *transmittance* T, which together with absorbance, are two related but different quantities used in spectrometry. The main difference between absorbance and transmittance is that absorbance measures how much of an incident light is absorbed when it travels through a material while transmittance measures how much of the light intensity is transmitted. Due to the way they are defined, the two are not complementary quantities, that is, adding transmittance to absorbance directly does not give the total incident light.

The mathematical interrelation between transmittance and absorbance is articulated in the subsequent expression:

$$T = \frac{I}{I_0} = e^{-A} = e^{-\alpha \cdot L \cdot c} \tag{2.8}$$

where transmittance T is a unitless quantity defined as the ratio of the transmitted light's intensity I to the incident light's intensity I_0. From Equation 2.8 we can derive the gas concentration c as:

$$c = \frac{ln(I/I_0)}{-\alpha \cdot L} \tag{2.9}$$

To calculate the gas concentration, the sensor system relies on the absorption of infrared light at specific wavelengths that correspond to the CO_2 absorption bands. The photodiode detects the intensity of light transmitted through the gas sample and generates an electrical signal proportional to it. This signal is used to determine the transmittance, defined as the ratio of the transmitted light intensity I to the incident light intensity I_0, where I_0 is obtained during a calibration process with a known concentration of CO_2. By knowing the molar absorptivity α of CO_2 at those specific wavelengths and the optical path length L, the concentration c can be accurately determined.

Various factors can influence the accuracy of the sensor based on the Beer-Lambert law:

1. **Optical Path Length (L):** The length of the optical path within the gas sample affects the amount of light absorbed. A longer path length leads to higher absorbance and, consequently, more accurate concentration measurements.

2. **Temperature:** Temperature variations can impact the sensor's performance, especially if the gas sample's density and pressure change. Compensation mechanisms are employed to account for temperature effects.

3. **Reference Concentration (Fresh Air - 400 ppm):** To ensure accuracy, many CO_2 sensors use fresh air with a known concentration of CO_2 (typically 400 ppm) as a reference point during calibration.

4. **Interference from Other Gases:** The presence of other gases in the sample can interfere with the CO_2 measurements. Cross-sensitivity to other gases is a challenge that sensor designers address to maintain specificity.

5. **Wavelength Selection:** The choice of specific infrared wavelengths is crucial to ensure the most significant absorption bands of CO_2 are targeted, optimizing accuracy and sensitivity.

In optical CO_2 sensors, the most commonly used infrared wavelengths are in the mid-infrared (MIR) range, as this is where CO_2 exhibits its most prominent absorption bands. The two main absorption bands for CO_2 in the MIR region are:

- 4.26 μm: this is known as the strong absorption band and is often used in many CO_2 sensors. It provides a significant and distinct absorption peak that allows for accurate measurements.

- 2.0 μm: this is known as the weak absorption band but is still used in some CO_2 sensors. It provides an additional data point to improve the accuracy of the measurements.

The choice of these wavelengths depends on the sensor's design and application requirements, and they play a crucial role in achieving high accuracy and sensitivity in CO_2 concentration measurements.

By carefully accounting for these specific wavelengths and the rest of the factors mentioned above, optical CO_2 sensors can achieve precise and reliable measurements. The Beer-Lambert law's versatility and robustness make it a cornerstone of optical gas sensing, enabling applications in a wide range of fields, from environmental monitoring to industrial process control.

2.3.2 Types of Optical CO_2 Sensors and Applications

Optical CO_2 sensors are versatile devices used in a wide range of applications, and their design and performance characteristics vary depending on specific requirements. There are various classes of optical gas sensors, each with their unique features and applications. These include Non-dispersive Infrared (NDIR), Photoacoustic Spectroscopy (PAS), Tunable Diode Laser Absorption Spectroscopy (TDLAS), and Spectrophotometry [57].

As the name suggests, in an NDIR gas detection system, optical dispersion of IR radiation is not needed. Therefore, dispersive elements like diffraction gratings are absent in these types of sensors, making them simpler and often more cost-effective. An ample review of NDIR sensors can be found in [58].

In Photoacoustic Spectroscopy (PAS), a gas sample is exposed to modulated light, usually from a laser, which it absorbs. The absorbed energy then causes a localized temperature change, leading to a pressure change within the gas sample. This pressure change manifests as sound waves, which are detected to determine the gas concentration. Known for its high sensitivity and selectivity, PAS is well-documented in the scientific literature, as indicated by numerous studies [59]–[61].

Tunable Diode Laser Absorption Spectroscopy (TDLAS) employs a tunable laser diode to scan a narrow wavelength range across an absorption line of the gas to be detected. The gas concentration is then determined based on the absorption of the laser light. TDLAS is highly accurate and can be applied for both trace gas detection and high-concentration measurements. For more comprehensive insights into the capabilities and applications of TDLAS, influential works by Werle et al., Lackner, and Li can be consulted in [62]–[64].

Spectrophotometry involves shining a light through a gas sample and then measuring the intensity of light absorbed at each wavelength using a detector. The absorbed wavelengths help identify the types of gas present and their concentrations. Spectrophotometry is highly versatile but often requires more complex instrumentation compared to other optical gas sensors. Each of these optical gas sensor technologies offers different advantages and trade-offs in terms of sensitivity, selectivity, and complexity, making them suitable for varied applications. For a deeper understanding and further comparisons, interested readers may consult the referenced studies [65], [66].

In this study, we will focus on NDIR gas sensors, which are both straightforward to implement and highly promising for environmental monitoring. This is largely because many environmentally significant gases exhibit strong absorption characteristics in the mid-infrared regime ($2.5\ \mu\text{m} - 14\ \mu\text{m}$). This absorption is due to the fundamental transitions in molecular rotation and vibration energy states, making it approximately 100 times more effective than absorption in the near-IR region.

The history of NDIR sensors is relatively recent, tracing its advancements to the development of optical bandpass filters in the 1970s. Since then, research and interest in this technology have grown steadily. This technology has earned the trust of the general public, leading to its widespread application across various sectors—ranging from oil and gas industries to coal mines, pharmaceuticals, and automobile manufacturing. Initially mechanical-optical in design, NDIR sensors have evolved into purely electronic systems, thanks to the integration with low-cost microprocessors. Modern NDIR sensors offer numerous advantages, including high selectivity, sensitivity, and low power consumption.

A key consideration in the design and operation of NDIR gas sensors is the selection of the light source, which typically falls into one of two categories: broadband sources like lamps and narrowband sources such as Light-Emitting Diodes (LEDs). Each category presents its own set of unique advantages and challenges:

Broadband Source (Lamps): Broadband sources emit light across a wide range of wavelengths. These sources are often based on incandescent lamps or thermal emitters. One of the main advantages of using lamps is their broad spectral output, which allows them to measure CO_2 concentrations over a wide range. However, they are sensitive to temperature variations, and their power consumption tends to be higher compared to narrowband sources.

Narrowband Source (LEDs): Narrowband sources, particularly LEDs, emit light at specific wavelengths, precisely tailored to the CO_2 gas absorption bands. LEDs offer advantages such as high spectral purity, low power consumption, and reduced sensitivity to temperature changes. This makes them a popular choice for energy-efficient and temperature-stable CO_2 sensors. However, their narrowband nature limits their measurement range compared to broadband sources.

Applications

Optical CO_2 gas sensors serve as crucial tools across a diverse array of applications, each demanding unique performance criteria. Below are some key domains where these sensors are making an actual impact:

Indoor Air Quality Monitoring: Within buildings, offices, schools, and homes, CO_2 sensors are commonly used for gauging indoor air quality. By monitoring CO_2 levels, these sensors contribute to optimal ventilation and air circulation, factors that directly influence the health and comfort of occupants.

Industrial Process Control: In industrial applications like fermentation, where CO_2 production is a key process metric, precise monitoring is essential. Proper control ensures optimal process conditions, thereby enhancing efficiency and product quality.

Agriculture and Greenhouses: CO_2 sensors are vital in agriculture and greenhouse environments. Monitoring CO_2 levels aids in maintaining optimal conditions for plant growth, as CO_2 plays a significant role in photosynthesis.

Safety and Environmental Monitoring: These sensors find use in safety-related applications such as hazard detection in confined spaces and underground mining. They also contribute to broader environmental efforts, aiding in air quality assessments and climate change studies.

Vehicle Emissions Testing: CO_2 sensors are utilized in vehicle emissions testing to monitor exhaust gas emissions and ensure compliance with environmental regulations.

Regarding the measurement range for CO_2 sensors, the required scope varies based on the specific application. For indoor air quality monitoring, sensors usually measure up to 2000 ppm (parts per million). In industrial settings, however, the need may arise to monitor concentrations that can go up to several percent (e.g., 0-5%). In our targeted application, our primary focus lies on ranges suitable for low-power IoT devices that measure CO_2 concentrations pertinent to indoor air quality.

2.3.3 Light Source Modulation in NDIR CO_2 Sensors and Signal Demodulation

In NDIR CO_2 sensors, the light source used to probe the gas absorption is often modulated or chopped, employing either mechanical or electronic techniques. This modulation is crucial to mitigate the impact of thermal background signals, enhance the signal-to-noise ratio, and ultimately improve the accuracy and sensitivity of the gas concentration measurement.

Mechanical Chopping: Mechanical chopping involves physically interrupting the light path using a rotating chopper wheel or a vibrating mirror placed in front of the light source. This causes the light to alternate between the sensor and a reference path (without the gas sample) at a known frequency. The alternating light signal is then detected by the photodetector, resulting in a modulated electrical signal.

Electronic Modulation: Electronic modulation is an alternative approach to achieve the same effect without mechanical components. In this method, the light source's current or voltage is modulated at a specific frequency, generating an intensity-modulated light signal. The modulated light is then directed towards the gas sample, and the resulting signal is detected by the photodetector.

Demodulation Process (Asynchronous Quadrature Demodulation)

After the modulated light interacts with the CO_2 gas sample, it is detected by the photodetector as an intensity-modulated electrical signal. The next step is to demodulate this signal to extract the gas concentration information accurately.

Asynchronous quadrature demodulation is a widely used technique for this purpose. It involves processing the modulated signal through a series of digital filters to separate the in-phase (I) and quadrature (Q) components. These components represent the real and imaginary parts of the signal, respectively.

The demodulation process involves multiplying the modulated signal with a reference sinusoid at the same frequency as the light modulation. This multiplication results in a pair of signals, I and Q, which can then be low-pass filtered to remove high-frequency components. The low-pass filtered signals represent the amplitude and phase of the modulated signal, respectively. A comprehensive review of demodulation techniques and various applications can be found in [67]–[69].

From these amplitude and phase components, the gas concentration can be accurately inferred using the Beer-Lambert law, taking into account the known characteristics of the sensor, such as the optical path length, gas absorption coefficient, and reference concentration.

Advantages of Light Source Modulation in CO_2 sensors

Light source modulation in NDIR CO_2 sensors offers several significant advantages that improve the accuracy and reliability of gas concentration measurements. These advantages include:

1. Enhanced Signal-to-Noise Ratio: It reduces the impact of background noise, resulting in a more accurate and reliable gas concentration measurement. By separating the desired gas absorption signal from noise and interference, the sensor can achieve better signal fidelity and higher sensitivity.

2. Offset of Thermal Background: Modulation enables the system to distinguish between the gas absorption signal of interest and the thermal background signals. The alternate reference path or sinusoidal modulation allows the sensor to accurately isolate the desired signal from temperature-induced variations, leading to precise gas concentration determination.

3. Reduced Interference: Light source modulation helps to mitigate interference caused by ambient light variations and external sources. By modulating the light at a specific frequency, the sensor can filter out unwanted external signals, improving the robustness of the measurement in challenging environments.

4. Improved Sensitivity: It increases the sensor's sensitivity, enabling precise measurements even at low gas concentrations. The ability to amplify and demodulate weak signals enhances the sensor's detection capabilities, making it suitable for applications where high sensitivity is essential.

By leveraging these advantages, light source modulation plays a crucial role in the accurate and reliable measurement of CO_2 concentrations in various applications, including indoor air quality monitoring, industrial process control, and environmental monitoring. These techniques are widely adopted in optical CO_2 gas sensors, ensuring the accurate monitoring of CO_2 levels for diverse purposes.

Advantages of Digital Demodulation

Using digital demodulation instead of analog demodulation offers several significant advantages in various applications, especially in the context of modern electronic systems. Here are some key advantages:

1. Noise Immunity: Digital demodulation is less susceptible to noise and interference compared to analog demodulation. Digital systems can employ error correction techniques, filtering, and signal processing algorithms to effectively reduce the impact of noise and improve the signal-to-noise ratio, resulting in more accurate and reliable measurements.

2. Flexibility and Configurability: It allows for greater flexibility and configurability in signal processing. By implementing algorithms in software, digital systems can easily adapt to different modulation schemes, frequencies, and signal formats without the need for hardware modifications. This makes digital demodulation highly versatile and suitable for a wide range of applications.

3. Ease of Implementation and Debugging: It can be implemented using software-based algorithms, which are generally easier to develop, test, and debug compared to complex analog circuits. This simplifies the design process and reduces development time, making digital demodulation a more practical and efficient solution.

4. Scalability and Upgradability: It can be easily scaled and upgraded

to accommodate changing requirements or improvements in demodulation techniques. Software updates can be applied without the need for hardware changes, allowing for seamless upgrades and advancements in the demodulation process.

5. Cost-Effectiveness: It can be more cost-effective than analog demodulation in many cases. Digital signal processing components are often more affordable and readily available, and the use of software-based algorithms reduces the need for specialized analog hardware components.

6. Signal Analysis and Post-Processing: Digital demodulation provides more opportunities for signal analysis and post-processing. Digital data can be easily stored, analyzed, and visualized, allowing for in-depth examination and extraction of valuable information from the demodulated signals.

8. Adaptive and Intelligent Processing: It allows for the implementation of adaptive and intelligent processing techniques. Advanced algorithms can adapt to changing signal conditions, optimizing demodulation parameters in real-time for improved performance.

In summary, digital demodulation offers superior noise immunity, flexibility, ease of implementation, and scalability compared to analog demodulation. These advantages make digital demodulation the preferred choice in many modern electronic systems, ranging from communication systems and wireless technologies to sensor applications and signal-processing tasks.

2.3.4 Review of Wireless Sensor Network Nodes for CO_2 Sensing

Wireless Sensor Network (WSN) nodes, also known as *sensor motes*, play a crucial role in environmental monitoring and data acquisition. These compact and integrated devices combine sensors and microprocessors, enabling them to monitor various environmental conditions or physical systems. One of the key advantages of WSN nodes is their ability to perform on-edge computing, which reduces the need to transmit large volumes of raw data to centralized data centers. Instead, the onboard processor in a WSN node performs preliminary data processing and filtering, sending only relevant and preprocessed information to the data center.

As depicted in Figure 2.1, a typical wireless sensor node consists of four main components: a sensor, a microprocessor, a radio transceiver, and a power supply. The sensor is responsible for capturing specific environmental parameters, such as CO_2 concentration, temperature, humidity, or light intensity. The microprocessor handles data processing tasks, enabling local decision-making and reducing the power consumption associated with data transmission. The radio transceiver facilitates wireless communication between sensor nodes and enables the formation of a self-organizing network. Finally, the power supply provides the necessary energy to operate the sensor node, and power efficiency is a critical consideration in WSN design.

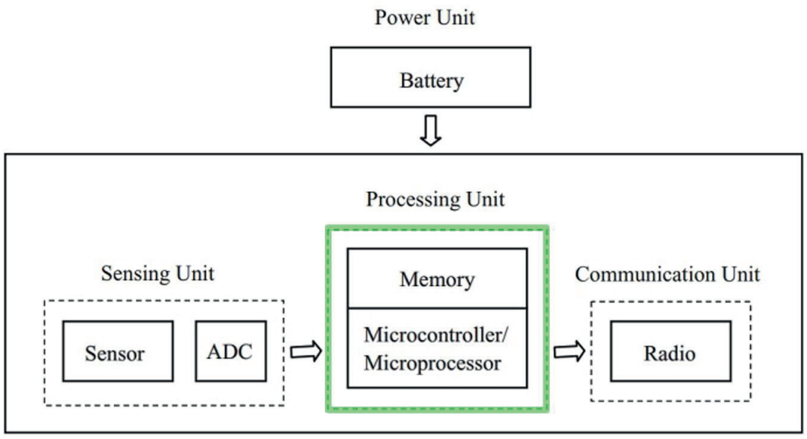


Figure 2.1: Basic components of a wireless sensor node in a WSN

WSNs are infrastructure-less networks[1], often deployed in an *ad hoc* manner, where a large number of wireless sensor nodes collaboratively monitor the physical or environmental conditions of interest. These networks find widespread applications in various fields, including environmental monitoring, agriculture, industrial process control, smart buildings, and healthcare.

In the context of CO_2 sensing, WSN nodes equipped with CO_2 sensors hold significant promise for various applications. They can be deployed in indoor environments to monitor air quality, allowing for timely interventions to improve ventilation or reduce CO_2 levels. In outdoor scenarios, WSN nodes can be used to assess carbon dioxide emissions from industrial facilities, urban traffic, or natural sources, aiding in environmental studies and climate research. Additionally, the ability of WSN nodes to operate in remote and challenging environments makes them suitable for monitoring CO_2 levels in areas with limited accessibility, such as forests or wildlife habitats.

Furthermore, the data collected by WSN nodes can be transmitted to a central Base Station, which acts as the processing unit in the WSN

[1]Ad hoc Networks, or Infrastructure-less Networks, are comprised of independent terminals that communicate with each other by forming a radio network. Wireless networks typically have lower bandwidth compared to wired networks. In such networks, each node acts as a router, and the network connection is distributed among nodes.

system. The Base Station aggregates data from multiple nodes and, in some cases, performs additional processing before forwarding the relevant information to cloud-based servers through the Internet. This architecture allows for scalable data management and real-time access to the monitored parameters.

Related Work on WSN Nodes

There have been several surveys and comparative studies on wireless sensor nodes [70]–[74] published after the inception of the RISC-V project in 2010. These studies provide an overview of numerous commercial and academic wireless sensor nodes, commonly referred to as *sensor motes*, and offer comparisons based on components, technologies, platforms, and other relevant parameters. In examining these surveys, it becomes evident that the processors employed in commercial nodes are predominantly based on RISC architectures, ranging from 8-bit processors like ATMega128 to 32-bit processors such as ARM Cortex M3, which is widely utilized [72].

Interestingly, none of the surveys mentioned above make reference to devices based on RISC-V processors, despite RISC-V being an emerging architecture with substantial potential, particularly in microcontroller design—a domain that currently witnesses significant RISC-V developments. Notwithstanding the ecosystem around RISC-V not being entirely mature, there are notable System-on-Chip (SoC) designs based on RISC-V specifically tailored for low-power IoT edge processing. Many of these designs have emerged after 2018 and are the work of the same research group at the University of Zürich (ETH). Notable examples include GAP-8 [75], Mr.Wolf [76], and Arnold [77]. These innovative RISC-Vbased SoCs represent a progressive trend in the domain of wireless sensor nodes and underscore the increasing relevance of RISC-V in this field.

Energy efficiency stands as the most critical concern for battery-powered Wireless Sensor Network (WSN) nodes. The processors highlighted in the aforementioned surveys are commercial cores whose source codes or schematics remain inaccessible to users. Consequently, modifying the microprocessor or the hardware architecture becomes unfeasible, leading to limited scalability of such sensor nodes. On the other hand, proprietary soft-core processors demand substantial license fees, creating financial barriers for designers.

In contrast, open-source ISAs like RISC-V offer the prospect of creative designs, empowering designers to implement optimal architectures tailored to specific applications. This advantage extends further, enabling straightforward modifications to reuse the same core as the application evolves.

Thus, the realization of an energy-efficient and scalable wireless sensor node utilizing 32-bit RISC-V-based open cores becomes an attainable goal. The inherent flexibility and openness of RISC-V pave the way for innovation and optimized designs, presenting an attractive solution for developing next-generation WSN nodes.

Regarding the hardware platforms employed in WSN nodes, a plethora of systems exist, encompassing microcontrollers, FPGA/SoPC, DSP, ASIC/-SoC, MPSoC, MPPSoC, or hybrid platforms. In this context, Vieira *et al.* were pioneers in their 2003 survey [78], proposing the use of FPGA as a platform to address architectural challenges arising from emerging applications. These challenges encompass computational power, energy consumption, energy sources, communication channels, and sensing capabilities.

For instance, some FPGA-based sensor motes leveraging 32-bit softcore processors are described in [79], where an OpenRISC 1200 processor is employed, and in [80], where a Xilinx MicroBlaze processor is used for vibration analyses in industrial machines utilizing FFT. These FPGA-based solutions offer a versatile and flexible approach, allowing designers to tailor the hardware architecture to the specific demands of the application, thereby enhancing performance, energy efficiency, and adaptability. As WSN technology continues to advance, exploring and innovating with diverse hardware platforms, including FPGA-based implementations, remains a promising avenue for further improvement and optimization.

Therefore, we believe that an FPGA platform for a RISC-V-based WSN node is a highly suitable choice, as it offers fast prototyping, high performance, and exceptional flexibility in both hardware and software implementations. Moreover, new FPGA vendors now provide very low-power FPGAs [81], which have already been successfully implemented in low-power WSN nodes, such as HaLoMote [82]. This combination of features makes FPGA-based solutions well-suited for WSN applications, enabling designers to efficiently explore and optimize their designs while meeting the stringent requirements of energy efficiency and performance demanded by battery-powered wireless sensor nodes.

In summary, advancements in WSN nodes, especially in sensor technology, power efficiency, and data processing, are pivotal for the success of CO_2 sensing applications. These nodes are key to addressing environmental challenges and promoting sustainable development, making research in this area extremely important given the extensive applications of WSNs across various fields.

Chapter 3

Processor Development and Evaluation Methodology

In this chapter, we delve into the methodology adopted to achieve the objectives of this industrial-oriented work. The emphasis here is not merely on academic exploration but also on delivering practical, energy-efficient processor designs that could be deployed in real-world applications.

We provide a thorough review of the processor's design process, touching on various aspects and techniques that contribute to its energy-efficient performance. We will navigate through the design space, meticulously examining different options and considerations to optimize the processor's architecture.

Moreover, we identify the benchmark application that serves as a testbed for assessing the real-world performance of the designed processor. Special attention will be given to algorithmic tweaks and optimizations that bring the performance closer to industrial application requirements. Following this, the chapter discusses how this custom-designed processor is integrated into the PULPino SoC, along with the validation of results obtained from RTL simulations. Additionally, power analysis and energy consumption estimations will be conducted to assess the processor's energy efficiency.

Overall, this chapter offers a comprehensive guide to the design-oriented and practical methodology employed throughout this study, giving detailed insights into the processor's design, operational efficiency, and energy metrics.

3.1 Review of the Design Process for our Custom Low-Power Processor

In this section, we provide a comprehensive review of the custom-designed processor architecture tailored for energy efficiency. The processor was thoroughly crafted to meet the specific requirements of our target application, which involves measuring CO_2 concentrations in the ambient environment.

The design process of our custom low-power processor commenced with a self-contained approach, where we focused on developing a processor and memory-only system. This minimalistic design was aimed at understanding the intricacies of the processor architecture without the complexities introduced by peripheral devices or memory hierarchies. By starting with this simple setup, we could gain deep insights into the processor's behavior and performance characteristics.

Then we proceeded with a meticulous exploration of the RISC-V ISA, aiming to understand its intricacies and develop insights for efficient processor design. Initially, we adopted a single-cycle design approach, which provided a valuable understanding of the instruction set and its associated complexities. A single-cycle processor design, also known as a single-cycle architecture, is a simple and straightforward implementation of a processor where each instruction is executed in a single clock cycle. Basically, all the required steps of instruction fetching, decoding, executing, and memory access are completed within one clock cycle. This approach allowed us to gain familiarity with the RISC-V architecture, enabling us to experiment with different optimizations to enhance energy efficiency.

In addition, in this self-contained design, we bypassed the incorporation of a memory hierarchy, opting for a flat memory model that accessed the single memory module directly, as in a sort of scratchpad memory. The benchmark program, sample data set and lookup tables required to calculate sine and cosine functions are all stored in this predictable and straightforward memory space. This decision allowed us to focus solely on the processor's functionality and performance, disregarding the intricacies associated with memory caching and hierarchy management.

Furthermore, our design targeted a purely embedded application, eliminating the need for an operating system. By designing the processor to operate in a standalone manner, we could eliminate the overhead associated with running an operating system, which is common in more complex computing systems. This allowed us to allocate more resources and attention to optimizing the processor's core functionality and energy efficiency.

As the design evolved, we progressed to a pipelined architecture, recognizing the numerous advantages it offers for low-power processors. Pipelining allows the processor to execute multiple instructions simultaneously, greatly improving performance while keeping power consumption in check. By breaking down the instruction execution into multiple stages, we could achieve higher throughput while still maintaining low power consumption.

The number of pipeline stages plays a crucial role in striking a balance between performance and energy efficiency. Each pipeline stage introduces latency and has its associated trade-offs. In general, more stages can lead to better performance because it allows for more parallelism. However, there is a trade-off between performance and complexity. A larger number of stages can increase the complexity of the processor and introduce more hazards and data forwarding complexities and potentially could lead to higher energy consumption due to increased register file accesses. The optimal number of stages depends on the specific application and design goals.

In our initial design, we chose to employ the classic 5-stage RISC pipeline, which is a fundamental and well-established approach in RISC (Reduced Instruction Set Computer) architectures. The 5-stage pipeline includes the stages of instruction fetch, instruction decode, execution, memory access, and write-back and some examples of RISC processors that use this pipeline are MIPS, SPARC, and Motorola 88000. This design decision was made for several reasons.

Firstly, the classic RISC pipeline offers a straightforward and structured way to organize the execution of instructions, making it an excellent starting point for academic purposes and learning about computer architecture and processor design. Secondly, by using this pipeline, we can efficiently explore and understand the core concepts of RISC architectures, such as instruction-level parallelism and data hazards. This understanding is crucial as we progress to more sophisticated pipeline designs and optimizations.

3.1.1 Initial Support for the Base ISA RV32I

In designing our custom low-power processor, we took a practical and incremental approach. We started with a single-cycle design, initially focusing on simple logic and arithmetic instructions from the RV32I ISA. Then, we proceeded to implement more complex operations such as loads, stores, and conditional jumps. This step-by-step incorporation allowed us to verify the core's functionality in a controlled environment, without the complexity of more advanced features. Once we successfully implemented the complete set of RV32I base instructions, we elevated the design by adding pipeline stages to enhance performance and throughput.

With a stable and functional pipelined architecture in place, we then turned our attention to extending the processor's capabilities. We added support for various ISA extensions by incorporating new hardware elements, such as specialized functional units for integer multiplication and division, and for floating-point operations. Each new feature was integrated carefully, ensuring both compatibility and performance gains at each development stage.

By starting with the RV32I ISA and a single-cycle design, we laid a strong foundation that made it easier to iteratively develop, test, and optimize our processor. This approach, aligned with the philosophy of RISC-V's modular and scalable design, not only simplified verification but also provided a clear pathway for future enhancements and specialization. Additionally, supporting the base ISA RV32I allowed us to evaluate the performance and energy efficiency of our processor in its simplest form. This early assessment served as a valuable baseline to measure improvements achieved through subsequent optimizations and extensions.

While our initial design is based on a conventional RISC pipeline, we acknowledge that contemporary processors often utilize deeper pipeline stages to improve both performance and energy efficiency. Due to time constraints, the evaluation of the benefits of adding more pipeline stages on energy efficiency has been set aside for future work.

3.1.2 The Classic 5-stage RISC Pipeline

The classic 5-stage RISC pipeline is a fundamental architecture widely used in many RISC processors, and is comprehensively described in the seminal book by Patterson and Hennessy [83]. This architecture divides the instruction execution process into five distinct stages, each responsible for a specific task. By doing so, the pipeline enables multiple instructions to be processed concurrently, thereby enhancing the processor's overall performance and efficiency.

The five stages of the classic RISC pipeline are as follows:

1. Instruction Fetch (IF): In this stage, the processor fetches the instruction from memory using the program counter (PC) as the address. The PC is then incremented to point to the next instruction in memory. The instruction fetched from memory is the one that will be executed in the subsequent stages of the pipeline.

2. Instruction Decode (ID): In the ID stage, the fetched instruction is decoded to determine the operation to be performed and the operands involved. The processor identifies the type of instruction (e.g., arithmetic, load/store, branch) and extracts the necessary information from the instruction.

3. Execution (EX): The execution stage is where the actual operation specified by the instruction is performed. This includes arithmetic operations (e.g., addition, subtraction), logical operations (e.g., AND, OR), and other computations. For arithmetic operations, the ALU (Arithmetic Logic Unit) is responsible for carrying out the computations.

4. Memory Access (MEM): In the MEM stage, memory-related operations are performed, such as loading data from memory (load) or storing data to memory (store). If the instruction is a load, the data is fetched from memory and made available for the next stage. If it is a store, the data is written to the specified memory address.

5. Write-Back (WB): In the final stage, the results of the execution are written back to the appropriate register file or memory, depending on the instruction type. For arithmetic operations, the results are typically stored in a register. For load instructions, the loaded data may be written back to a register, while store instructions do not produce any results for write-back.

The classic 5-stage RISC pipeline allows each stage to be dedicated to a specific task, resulting in a simplified and streamlined instruction execution process. It facilitates pipelining, where multiple instructions are in different stages of execution simultaneously. This pipelining enables high instruction throughput and improved performance.

However, the classic pipeline is not without challenges. Data hazards, control hazards, and structural hazards may arise due to dependencies between instructions or due to conflicts in accessing shared resources. These hazards need to be managed effectively to ensure correct and efficient execution of instructions.

Despite its simplicity, the classic 5-stage RISC pipeline serves as the basis for more complex and optimized pipeline designs used in modern processors. It provides a solid foundation for understanding the principles of pipelining, which is crucial for developing energy-efficient and high-performance processors for various applications.

3.1.3 Processor Pipeline Hazards and Optimizations

In a pipelined processor, hazards are situations where the proper execution of instructions is hindered due to conflicts arising from the concurrent nature of the pipeline stages. These hazards can lead to incorrect results and stall the pipeline, reducing the processor's efficiency. The three primary types of pipeline hazards are:

1. Data Hazards: Data hazards occur when an instruction depends on the result of a previous instruction that has not yet completed its execution. This creates a conflict as the subsequent instruction requires the data produced by the preceding one. Data hazards can be addressed through techniques like **data forwarding** (also known as data bypassing) or **stalling** the pipeline until the required data is available. Data forwarding forwards the necessary data from the execution stage to the instruction that needs it, reducing the stalls and improving performance.

2. Control Hazards: Control hazards arise from conditional branches and jumps. When a branch instruction is in the execution stage, the target address of the branch is not known yet. If the branch is taken, the instruction fetch stage needs to be redirected to the new target address, leading to a pipeline flush. Techniques like branch prediction and delayed branching can mitigate control hazards by speculatively predicting the outcome of branches and avoiding pipeline flushes.

3. Structural Hazards: Structural hazards occur when multiple instructions require access to the same hardware resource simultaneously. For example, if two instructions want to access the memory at the same time, a structural hazard arises. This can be resolved by resource duplication or careful scheduling of instructions to avoid resource conflicts.

To tackle data hazards, we implemented data forwarding techniques whenever possible, ensuring that the data needed for instruction execution is available in the subsequent stages without introducing unnecessary stalls. This approach reduced pipeline bubbles (or NOPs - no operation instructions used to introduce delays or fill empty pipeline stages when no useful work is being done) and helped to enhance the overall efficiency of the processor. In other cases, for example, functional units that require several cycles to produce results (e.g., integer division), we chose to stall the pipeline until we obtained the result.

Another improvement in our design involved incorporating a dedicated comparator in the ID stage to handle branch instructions and mitigate control hazards. To enhance branch prediction efficiency, we adopted a "not taken" prediction scheme. With this modification, we succeeded in reducing the branch penalty by one clock cycle when the branch is taken. In conventional designs, the branch decision comparison is usually performed in the ALU on the subsequent stage (EX), resulting in an additional cycle delay. By introducing a specialized comparator, we effectively eliminated this penalty and significantly improved the overall performance of branch instructions.

3.2 Hardware/Software Design Space Exploration of the Processor

The hardware/software design space exploration of the processor involved a careful and iterative process aimed at optimizing energy efficiency and performance while meeting the specific requirements of the target applications. This exploration covered both hardware and software levels and included a wide range of design choices and configurations.

The primary goals of the design space exploration were to identify the most efficient configurations for the processor, considering factors such as the use of floating-point or fixed-point arithmetic, the adoption of specific RISC-V ISA extensions, and the incorporation of hardware accelerators. The exploration aimed to achieve a harmonious balance between performance, energy efficiency, and application-specific requirements.

Hereafter, the comprehensive list of actions undertaken in our design space exploration is presented, broadly categorized into optimizations at the hardware level and optimizations at the software level.

3.2.1 Optimizations at Hardware Level

Exploration of RISC-V Standard Extensions and Specialized Functional Units At the hardware level, we conducted extensive experiments to explore various options for enhancing the processor's capabilities. One such experiment involved incorporating the M standard extension, which enabled dedicated integer multiplication (MUL) and division (DIV) functional units, thereby significantly improving integer arithmetic performance. For the RTL description of the integer division module, we employed the standard long division algorithm [84], which takes one cycle per bit, resulting in 32 cycles to perform an integer division operation.

Furthermore, to facilitate floating-point operations, we integrated the F standard extension, allowing for physical support for floating-point arithmetic. The integration involved incorporating the Floating-Point Unit (FPU) in the EX stage of the processor pipeline. To achieve this, we utilized the *FPnew* [85] open-source IP, developed by the Digital Circuits and Systems Group at ETHZ (PULP Team). *FPnew* is a parametric floating-point unit that supports both standard RISC-V formats and operations, as well as transprecision formats. It is implemented in SystemVerilog, successfully integrated into our low-power processor pipeline, and thoroughly verified.

The addition of these standard extensions yielded significant benefits, as evidenced by the reduction in the number of instructions required by the benchmark algorithm during program compilation. This reduction, in turn, resulted in improved runtime performance and reduced energy consumption, aligning with our objectives of optimizing energy efficiency and achieving enhanced processor capabilities.

Exploration of DSP Slices Utilization To further bolster computational capabilities, we also explored the possibility of leveraging FPGA Digital Signal-Processor (DSP) slices to improve the efficiency of the MUL unit and the fused multiply-add (FMA) operation in the FPU unit. To achieve this, we utilized the DSP48E1 slice available in Xilinx FPGAs [86] for the integer multiplication unit. This operation requires only one clock cycle, and we had the option to pipeline the slice if desired. To guide Vivado in using either the DSP48E1 slices or the FPGA logic in the multipliers, we inserted the directive (* `use_dsp48 = "yes/no"` *) in the Verilog description of the multiplier module. Through our experimentation, we observed that the use of specialized DSP slices contributed to an average 30% reduction in power consumption, as evidenced by the results obtained from the conducted experiments.

Elimination of Logic Related to Unused Instructions As part of our energy optimization efforts, we conducted a thorough analysis of the code to identify and remove unused ISA instructions, thereby eliminating unnecessary circuitry. This analysis was facilitated by a specialized tool developed by D. Castells, further details of which can be found in [87].

Specifically, as a result of this analysis, we targeted shifts and comparisons associated with the `sra` (arithmetic right shift) and `slt[i]` (signed compare [with immediate]) instructions. Additionally, we removed logic related to the management of Control and Status Registers (CSRs), including the `csrrw[i]` (swap values in the CSRs and integer registers), `csrrs[i]` (read and set bits in CSR), and `csrrc[i]` (read and clear bits in CSR) instructions. To maintain a minimal design, the CSR was retained with only two registers: `mcycle`, the machine cycle counter, and `minstret`, the machine instructions-retired counter. These registers are essential for performance measurements and calculating the Instructions per Cycle (IPC) metric of the processor.

Certainly, the list of instructions mentioned above is derived from the analysis conducted on the code generated by the specific compiler utilized in our study. Additionally, it is essential to note that the list of instructions may vary depending on the characteristics and requirements of the application being considered. Different applications may have distinct sets of instructions that are utilized or remain unused, impacting the final list of instructions identified for removal during the design process.

By removing the logic associated with these unused instructions, we achieve a significant reduction in the utilization of FPGA resources, as will be further discussed in Chapter 4, dedicated to the presentation and analysis of the obtained results.

Eliminating Support for Misaligned Memory Access In addition to removing logic associated with unused instructions, we also explored the impact of eliminating support for misaligned memory accesses in the Load-Store Unit (LSU), seeking to reduce resource utilization without compromising performance. Misaligned memory accesses refer to situations where a data item spans across multiple memory locations that are not aligned on natural boundaries (e.g., accessing a 32-bit value that starts at an address not divisible by four). Applications that deal with data structures that are not naturally aligned can benefit from misaligned memory accesses. These applications often involve complex data formats or structures that require data to be accessed across multiple memory locations, with each memory location not aligned to its natural boundary. Examples of such applications include image and multimedia processing, compression and encryption algorithms, and graphics rendering.

While the RISC-V ISA does not impose restrictions on microarchitecture, chip designers have the freedom to choose whether to support misaligned memory accesses or not. Nonetheless, the ISA defines a trap mechanism to emulate these accesses in software, albeit with a time penalty. In our case, since our application does not necessitate access to complex data structures, we chose to remove support for misaligned memory accesses in our LSU. This decision was driven by the significant benefits observed in FPGA resource utilization upon eliminating this logic, making it an efficient trade-off for our specific design requirements. By focusing on aligning the processor's features with the application's needs, we could achieve a highly optimized and specialized processor for our target use case.

Implementation of the Extensions E and Zfinx Furthermore, we carefully considered the adoption of the standard RISC-V ISA extensions E and Zfinx to harness their potential for improved energy efficiency and resource utilization. The E extension, tailored for embedded systems, offers a smaller and more energy-efficient alternative to the standard RISC-V ISA. Similar to the I extension, the E extension features a smaller register file and can be used in conjunction with other RISC-V extensions.

On the other hand, the Zfinx extension provides instructions for floating-point operations in integer registers, which are not available in the base RISC-V ISA. It serves a wide range of applications, including machine learning, scientific computing, and embedded systems. Additionally, adopting the Zfinx extension conferred an extra advantage. Upon inspecting the compiled code, we observed that no integer multiplication or division was executed since these operations were efficiently realized in single-precision floating-point by the FPU. Consequently, this enabled the removal of the M extension and its specific instructions, namely `mul` (signed multiplication, write lower 32-bits), `mulh[u|su]` (signed [unsigned|signed×unsigned], write upper 32-bits), `div[u]` (signed [unsigned] division), and `rem[u]` (signed [unsigned] remainder), further optimizing the processor design for energy efficiency and reducing resource utilization.

Reduction of the Register File The integer register file `x` underwent strategic modifications due to the integration of the E extension, specifically designed for embedded applications, leading to a substantial reduction in its size from 32 to 16 registers.

Additionally, the F extension typically uses a separate set of `f` registers for floating-point computations, primarily aimed at alleviating register pressure in wide superscalar processors that execute multiple instructions per cycle in parallel to achieve higher performance. However, in our simplified RISC-V implementation, adopting the Zfinx extension substantially reduces the implementation cost by eliminating the whole floating-point `f` register file, resulting in optimized resource utilization and improved energy efficiency.

Overall, through these carefully implemented optimizations, the register file size was effectively reduced by a factor of 1/4 compared to a conventional RV32IMF implementation.

3.2.2 Optimizations at Software Level

Floating-point vs. Fixed-point Emulated Support During the software-level optimization phase, we thoroughly examined the advantages of using fixed-point versus floating-point representations, both with emulated support. For the floating-point representation, we utilized the *emFloat* C runtime library developed by SEGGER [88], while the fixed-point support was implemented using a C++ template class created by P. Schregle [89].

In our investigations, we found that when emulating these number representations in the design versions supporting RV32I and RV32IM, the fixed-point representation outperformed the emulated floating-point version by a factor of 4 in terms of execution time and total energy consumption. However, an interesting outcome emerged when we introduced the FPU in the RV32IMF processor version. The hardware-integrated FPU significantly outperformed the software-emulated fixed-point approach, reducing both the execution time and energy consumption by a substantial factor of 2/5.

Consequently, based on these comparative results, we concluded that the use of the fixed-point support in software was not as efficient as incorporating the FPU directly into the pipeline of our custom processor. As a result, we decided to proceed with the exploration of the FPU-integrated design for its superior performance and energy efficiency gains.

Look-Up-Table-Based Trigonometric Functions Another crucial aspect of software optimization involved exploring different strategies for trigonometric computations and comparing the advantages and drawbacks of using trigonometric functions versus employing Look-Up-Tables (LUTs). To achieve this, we created a file containing pre-calculated values of a full sine and cosine wave, which was then included during compilation with the rest of the application files. Since our benchmark application assumes a modulation frequency of 128 Hz and a sampling rate of 16,384 samples per second (128 times 128), the file comprises an array of 128 pre-calculated points for both the sine (in quadrature component) and cosine wave (in-phase component) with a frequency equal to the modulation frequency, i.e., 128 Hz. These arrays are accessed circularly by the application, effectively functioning as a look-up table, and are multiplied point-wise with the sampled signal from the CO_2 sensor to filter out undesired frequency components.

By opting for LUTs instead of making calls to trigonometric functions provided by the standard library, we achieved significant improvements. For instance, in the RV32EM_Zfinx processor version, we observed a reduction in the number of instructions by a factor of 1/20, along with a remarkable 1/25 reduction in energy consumption. The utilization of LUTs as a mechanism for trigonometric computations has proven highly effective, significantly enhancing both performance and energy efficiency in our target application.

3.2.3 Final Processor Design: RisCO2

The culmination of our iterative hardware/software design space exploration efforts yielded the realization of our custom RISC-V processor, which we have named *RisCO2* to reflect that it has been tailored for low-power embedded systems in CO_2 concentration measurements.

The ultimate version of RisCO2 presents a 5-stage, single-issue, in-order processor based on the RV32E_Zfinx instruction set. Our primary focus throughout the design process was to optimize for energy efficiency, making it ideal for integration into NDIR CO_2 sensors that require signal demodulation to infer gas concentration.

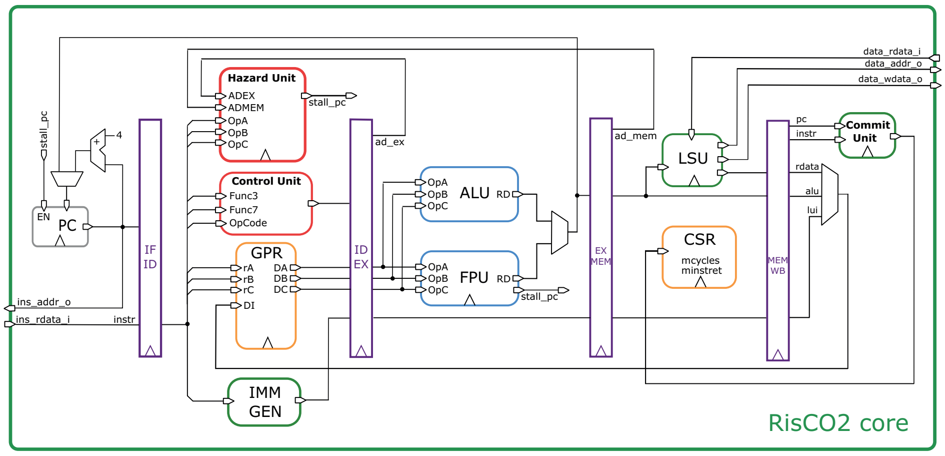


Figure 3.1: Simplified block diagram of the RisCO2 core architecture showing its five pipeline stages and all functional blocks.

The simplified block diagram of the core is depicted in Figure 3.1, illustrating the RisCO2 pipeline composed of five stages through which instructions pass during execution. These stages are Fetch (IF), Decode (ID), Execute (EX), Memory (MEM), and Writeback (WB).

In the IF stage, instructions are fetched from memory and stored in an instruction register. The ID stage then decodes the instruction in the Control Unit, identifies operands, and retrieves them from the General Purpose Register File (GPR). To address potential instruction dependencies, a Hazard Unit is employed, which detects and resolves hazards. This involves inserting pipeline bubbles or forwarding data between stages, ensuring accurate execution order and error-free operation.

A notable enhancement in the Control Unit is the inclusion of a dedicated comparator for branch instructions. This addition allows for earlier detection of branch outcomes and reduces latency in case a branch is taken. In situations where a branch is taken, the processor only needs to flush one instruction upon misprediction. This is achieved by inserting a bubble or no-operation instruction in the EX stage. The presence of the dedicated comparator contributes to improved branch performance and mitigates potential performance penalties associated with branch mispredictions.

Within the EX stage, the Arithmetic-Logic Unit (ALU) performs basic integer arithmetic and logic operations, while the FPU handles floating-point operations such as addition, subtraction, multiplication, division, square root, and FMA on single-precision (32-bit) floating-point numbers. Notably, the FPU exhibits varying latency depending on the operation, often spanning multiple cycles. To maintain correct program execution, the unit includes an output to stall both the program counter (PC) and the pipeline when needed.

In the MEM stage, data is read from or written to memory via the Load Store Unit (LSU). The Control and Status Register Unit (CSR) contains only two registers, namely `mcycle` and `minstret`, utilized for performance measurements.

Lastly, the WB stage handles writing the operation results back to the registers. The Commit Unit verifies when an instruction has reached the final pipeline stage, and its output is used to increment the `minstret` counter in the CSR.

The RisCO2 processor's final iteration showcased significant enhancements in energy efficiency and performance, making it ideally suited for real-world deployment in low-power optical gas sensors designed to measure CO_2 concentrations. These results not only fulfill the objectives of our industry-focused project but also contribute to academic discourse. An initial investigation into our design space exploration, specifically the performance metrics and results achieved during the early design iterations leading up to RisCO2, was published in a conference paper available on IEEE Xplore [90]. Building on that foundational work, our findings and the detailed description of the RisCO2 processor were further elaborated in a scholarly journal publication entitled "RisCO2: A Customized RISC-V Processor for Low-Power Optical CO_2 Sensors." This latter article was featured in *Micromachines* [91], an international, peer-reviewed, open-access journal on the science and technology of small structures, devices, and systems, published monthly online by MDPI.

Summary of Architectural Explorations and Improvements

To sum up, throughout this iterative design exploration process, we examined various alternatives, evaluating the impact of different architectural decisions and software configurations. Notably, we investigated the effects of employing floating-point or fixed-point representations, with emulated or physical support via the FPU unit. Moreover, we examined the influence of augmenting the processor with a MUL/DIV unit, affording the incorporation of the M extension, and explored the prospect of leveraging FPGA DSPs to empower the hardware multiplication functional unit. Additionally, we engaged in an in-depth analysis of the utilization of trigonometric computations versus look-up-table methods, as well as the integration of the E and Zfinx standard RISC-V ISA extensions. Aware of the importance of resource optimization, we evaluated the elimination of logic associated with unused ISA instructions, including those associated with control and status register (CSR) operations and unused integer arithmetic and logical instructions. Concurrently, we sought opportunities for streamlining the processor's Load-Store Unit (LSU) by considering the impact of removing support for misaligned memory accesses. The design space exploration proved instrumental in unveiling an array of possibilities, thereby providing valuable insights into the potential trade-offs between various design alternatives. It enabled us to refine the processor's architecture, optimizing it to proficiently handle the computational tasks intrinsic to our target applications while adhering to the principal objective of attaining a reasonable balance between energy efficiency and performance.

All the results obtained from these experiments can be checked out in Chapter 4, "Results and Discussion", where they will be discussed more thoroughly.

3.3 Concentration Measurement Methodology and Benchmark Application

In the development of our custom RISC-V processor, one of the critical considerations was the execution environment for applications. We adopted a bare-metal execution model, wherein applications run directly on the logic hardware of the processor without the involvement of an intervening operating system. This approach offers several advantages, including reduced overhead, lower power consumption, and enhanced responsiveness, making it well-suited for embedded systems targeting specific applications like CO_2 concentration sensing.

To effectively evaluate the performance of our processor during the design space exploration process, it was imperative to design a benchmark program that closely represents the target application. We focused on an IoT application dedicated to CO_2 concentration sensing. Such sensing applications have far-reaching implications across diverse domains, including industrial process control, safety monitoring, environmental monitoring, and public health.

Our choice to employ low-power optical sensors for this application was rooted in their exceptional suitability for low-power embedded systems. As discussed in previous sections, optical sensors, particularly those utilizing the principles of absorption-based infrared gas sensing, offer high sensitivity, accuracy, and energy efficiency. The energy-efficient nature of these sensors aligns seamlessly with the goals of our processor design, which prioritizes energy efficiency to cater to the power constraints of battery-operated or energy-sensitive applications.

With the rationale behind our choice of application and sensor technology well-established in previous chapters, this section will delve into the concentration measurement methodology and the specific design considerations for the benchmark application.

3.3.1 System Proposal

We propose the utilization of a WSN node as the system model for scripting a benchmark application tailored to our custom processor. Figure 3.2 illustrates the block diagram of the envisioned sensing platform, which takes clear inspiration from the Wireless Sensor Network (WSN) nodes described in Chapter 2. The system represents a wireless sensor node specifically designed for measuring and monitoring CO_2 concentration levels. It integrates several essential components, including an Infrared (IR) (IR) light-emitting diode (Light-Emitting Diode (LED)) serving as the emitter, a photodiode acting as the detector, an analog-to-digital converter (Analog-to-Digital Converter (ADC)), a RISC-V processor responsible for executing the digital demodulation algorithm, a radio transceiver for communication purposes, and a battery to provide power to the entire system.

Below, we provide a concise description of each component:

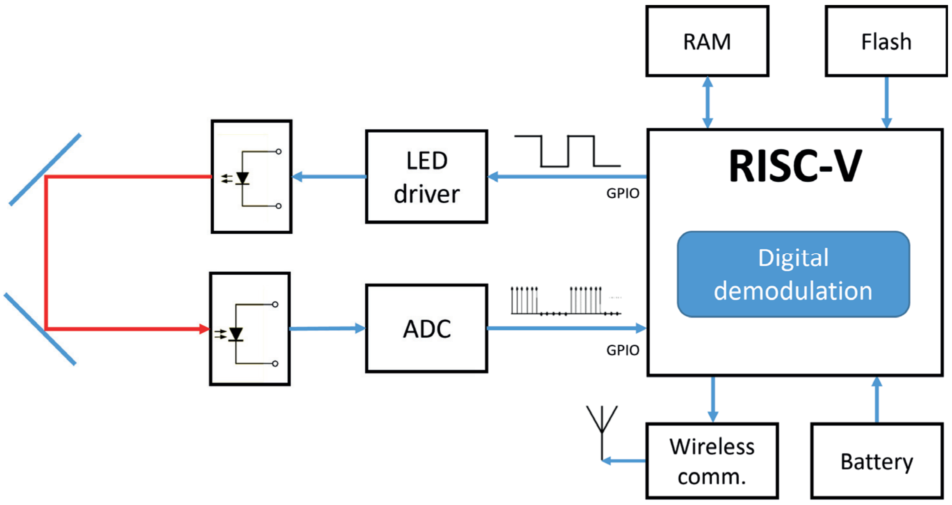


Figure 3.2: Proposed block diagram of the sensing platform

1. IR LED: The IR LED emits light at a specific wavelength (4.25 μm) that corresponds to the absorption spectrum of CO_2. This absorption of light by the gas molecules enables the measurement of the concentration of CO_2 in the air.

2. Photodiode: The photodiode detects the light transmitted through the gas sample and converts it into an electrical signal. It must possess good sensitivity to accurately detect low light levels, as well as high linearity to ensure that the converted electrical signal precisely represents the light intensity.

3. RISC-V processor: The FPGA is utilized to implement the RISC-V processor and the demodulation algorithm. The RISC-V processor controls the system's operation, while the demodulation algorithm extracts the CO_2 concentration information from the electrical signal generated by the photodiode.

4. ADC: The ADC is responsible for converting the analog signal generated by the photodiode into a digital signal that can be processed by the RISC-V processor. To ensure accurate and timely readings, it is important to select an ADC with adequate resolution and sampling frequency. The resolution determines the granularity of the signal and is crucial for accurately capturing the slight changes in light intensity detected by the photodiode. Higher resolution, expressed in bits, allows for a more precise digital representation of the analog signal. On the other hand, the sampling frequency must be high enough to capture the variations in the

analog signal over time, ensuring that no critical information is lost. These two parameters are carefully considered in our model simulation, and their effects on the accuracy of concentration level measurements are examined.

5. Demodulation algorithm: The demodulation algorithm is a software application loaded into the instruction memory of the RISC-V processor and is used to extract the CO_2 concentration information from the electrical signal generated by the photodiode. The algorithm employs digital signal processing techniques for this purpose.

6. Radio Transceiver: The radio transceiver enables wireless communication between the sensing platform and external devices or base stations. It facilitates data transmission and reception, allowing the sensor node to communicate its measurements to other nodes or data processing units.

7. Battery: The battery provides the necessary power to operate the entire sensing platform. It serves as the energy source, ensuring that the system can function autonomously without the need for a continuous external power supply. The choice of an appropriate battery is critical to achieve long-term and reliable operation in battery-powered wireless sensor nodes.

The sensor is based on an IR LED/photodiode optopair, which allows for a much higher modulation frequency compared to other types of detectors that use an IR lamp/thermopile couple [92], [93]. In the latter case, the modulation frequency is very low, typically in the range of 1 to 5 Hz, leading to poor noise filtering capability, and the output signal is highly temperature-dependent. In contrast, the IR LED/photodiode optopair in our sensor enables a higher modulation frequency, which enhances noise filtering and provides improved stability. While Figure 3.2 shows a wireless communication module, our primary focus in this work is on optimizing the processor design and the demodulation algorithm.

The path reflections experienced by the IR light in the drawing are caused by the reflective walls of the waveguide or the housing containing the sensor. These reflections lead to multiple interactions between photons and the target gas molecules [94], [95], effectively increasing the accuracy and sensitivity of the sensor.

In summary, the system depicted in Figure 3.2 serves as a CO_2 concentration sensing platform designed to demonstrate the effectiveness of the processor optimizations implemented during the design phase. The primary objective is to achieve a low-power RISC-V processor optimized for signal processing in wireless CO_2 sensor motes.

3.3.2 Asynchronous Quadrature Demodulation Technique

Asynchronous quadrature demodulation is a fundamental technique employed in optical gas sensors based on light absorption spectroscopy. This technique plays a central role in extracting the gas concentration information from the electrical signal generated by the photodiode, allowing precise and accurate measurements of the target gas.

Digital demodulation offers significant advantages over analog demodulation. It operates effectively with lower signal-to-noise ratios [10], enhancing signal extraction accuracy and minimizing waveform distortion. Moreover, digital demodulation can be efficiently implemented in software, making it adaptable for execution on our application-specific processor, providing the precision and efficiency needed for signal processing. This capability is crucial for achieving reliable and accurate gas concentration measurements in low-power IoT applications with stringent accuracy requirements. By using advanced digital processing capabilities, our processor becomes a powerful tool that can be used in various applications, including environmental monitoring and industrial safety.

Given the significance of the demodulation process in gas concentration sensing, we have selected it as a prime benchmark application to evaluate the energy efficiency and overall performance of our custom processor. By subjecting the processor design to the demands of the demodulation algorithm, we can assess its performance metrics, power consumption, and efficiency in handling signal processing tasks relevant to CO_2 concentration sensing.

Figure 3.3 illustrates the quadrature demodulation process and its underlying components. At the front-end side, we have the gas sensor and the ADC, responsible for converting the electrical signal into digital samples. In the digital domain, the demodulation process occurs on the RISC-V microprocessor, utilizing dedicated demodulation software to extract valuable information from the digital samples.

The symbols employed in the figure to depict the demodulation process are equivalent to those utilized in analog demodulation, symbolizing physical components like the local oscillator (LO), mixers, $90°$ phase shifter, low-pass filters (LPF), and magnitude detector. In the context of digital demodulation, these components translate into specific software processes within the digital domain.

Let us now delve into the quadrature demodulation technique and its constituent components. Quadrature demodulation involves the separation

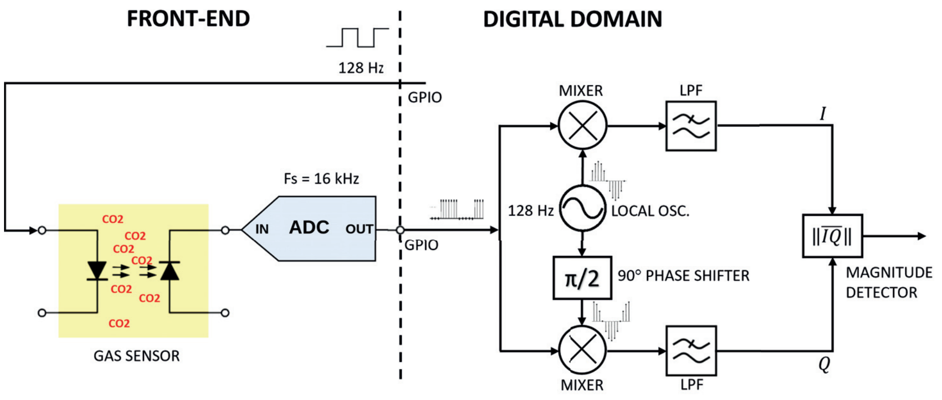


Figure 3.3: Illustration of the quadrature demodulation technique and its components.

of the input signal into its in-phase (I) and quadrature (Q) components, with I representing the signal's amplitude and Q representing its phase. This separation is achieved by employing a reference signal generated by the local oscillator along with its 90° phase-shifted counterpart.

The incoming sampled signal from the photodiode undergoes processing through point-wise multiplication with a reference sinusoid having the same frequency used for light modulation in the emitter (IR LED). In the digital domain, the reference signal can be obtained by computing the sine function at the modulation frequency for each discrete time point. The discrete time, in this case, is determined by the sampling rate used in the ADC. Similarly, the phase shifter generates a 90° phase-shifted version of the reference signal, corresponding to a cosine wave, which can be achieved digitally by computing the cosine function at different discrete time points.

Subsequently, both mixers shown in the diagram perform point-wise multiplication of the sine and cosine signals with the input samples. This operation filters out undesired frequency components, effectively yielding the in-phase (I) and quadrature (Q) components. These components are then subjected to further mathematical operations to extract the relevant data.

Following this, a first-order low-pass filter (LPF) is employed to remove high-frequency components from the I-Q components. The LPF employs a digital integrator, which is essentially a variable that accumulates partial products over a specified integration interval.

Finally, the magnitude detector block computes the modulus of the resulting complex I-Q vector, which is used to accurately infer the gas concentration utilizing the Beer-Lambert law, and taking into account the known characteristics of the sensor, such as the optical path length, gas absorption coefficient, and the reference concentration.

In summary, the asynchronous quadrature demodulation technique, along with the corresponding demodulation application, represents a critical and demanding workload, serving as a fitting assessment tool for evaluating the energy efficiency and performance of our custom processor in low-power IoT applications, especially those involving signal processing for gas concentration sensing.

3.3.3 Simulation Environment and System Modeling

For system modeling, we turned to Jupyter Notebook[11], an interactive computational platform known for its compatibility with Python scripting. Within its framework, Jupyter Notebook offers plenty of libraries tailored for data processing and visualization. Being open-source, it serves as a valuable tool for researchers and data scientists, offering a unified environment that puts together code, text, equations, and visualizations in one cohesive document. The underlying purpose of Jupyter Notebook is to streamline data analysis, numerical simulations, and research, making these processes both interactive and reproducible. While it provides support for various programming languages, our emphasis was on Python, given its rich ecosystem of libraries like NumPy, SciPy, and Pandas. These tools have proven to be indispensable for numerical computations and data visualization. Furthermore, the ability of Jupyter Notebook to execute code cells on-the-fly and instantly display results facilitates a smoother workflow, especially when analyzing data or experimenting with algorithms.

To model our system in Jupyter Notebook, we refer back to the Beer-Lambert law, presented in Equation 2.7 and the concept of transmittance expressed in Equation 2.8. This law establishes that the absorbance A of a gas sample is proportional to its concentration c and the optical path length L through which the light passes. Additionally, the absorbance is related to transmittance, which is defined as the ratio of the transmitted light's intensity I to the incident light's intensity I_0. For the sake of clarity, we revisit the equation presented in a previous chapter:

$$T = \frac{I}{I_0} = e^{-A} = e^{-\alpha \cdot L \cdot c} \tag{3.1}$$

The inverse exponential relationship between transmittance and gas concentration is illustrated in Figure 3.4.

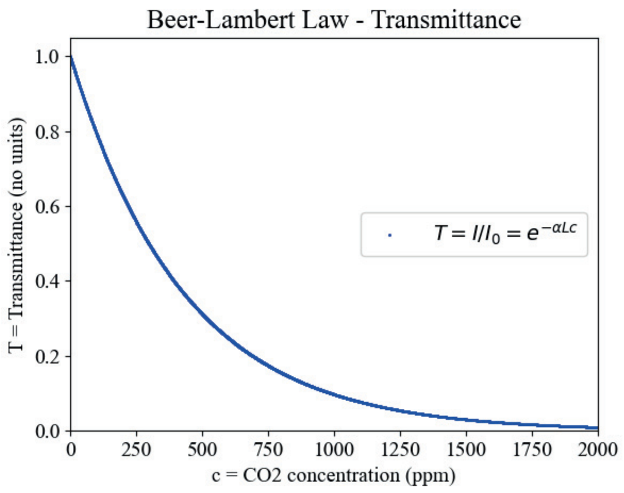


Figure 3.4: The transmittance function follows an inverse exponential curve ($\alpha = 0.023$ cm^{-1} · ppm^{-1} and $L = 10$ cm).

Transmittance is a unitless quantity ranging between 0 and 1. A value of 1 indicates 100% transmission of light. Absorbance, on the other hand, is logarithmically related to transmittance. This relationship can be expressed in terms of either the neperian or decadic logarithm. Consequently, the absorptivity coefficient α must be adjusted in alignment with this relationship.

The value of α is influenced by the wavelength of the incident radiation used in the sensor. To determine this coefficient, we reference spectral absorbance plots generated through simulation software. Specifically, we utilized SpectraPlot [96], a free web tool that simulates spectroscopic data using the HITRAN database [97]. HITRAN, short for high-resolution transmission molecular absorption database, serves as a comprehensive repository housing spectroscopic parameters [98]. These parameters find applications in various computer codes designed for predicting and simulating light emission and transmission within the atmosphere.

Figure 3.5 displays the absorbance spectrum of CO_2 for a specified concentration. The absorbance is not uniform across the spectrum; instead, distinct peaks corresponding to various wavelengths are evident. These peaks arise from the unique vibrational modes associated with the covalent bonds of the molecule.

A typical IR LED operating in the 4.3 μm absorption band of CO_2 has its emission spectrum centered around this wavelength, spreading to include

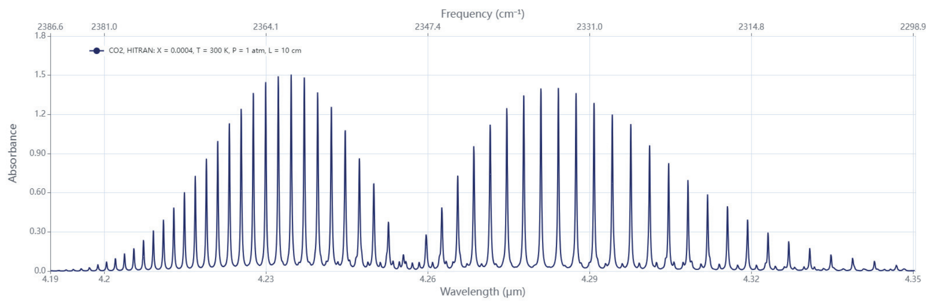


Figure 3.5: CO_2 absorbance spectrum in the range [4.19, 4.35] μm (T = 300 K, P = 1 atm, c = 400 ppm)

all absorbance peaks depicted in the plot. Given this, it is necessary to determine an average absorbance value across the frequency range aligned with the IR LED's emission spectrum. While one approach would be to integrate the curve across the x-axis, the pronounced peaks suggest an alternative strategy: identifying peak values on the plot using a script or spreadsheet tool, summing these values, and then calculating their average. We have successfully employed this latter approach in our study. By deriving the average absorbance, we can deduce the absorptivity coefficient α for a given optical path and concentration. This coefficient is consistently used in our simulations to calculate the CO_2 concentration according to Equation 2.9.

To determine the gas concentration, another essential measurement is the transmitted intensity I of IR radiation through the gas sample. This can be derived from Equation 3.1. The value of I is influenced by both the absorbance at a specific concentration and the initial intensity of the incident radiation I_0:

$$I = I_0 \cdot e^{-\alpha \cdot L \cdot c} \tag{3.2}$$

In Equation 3.2, certain quantities, such as the optical path L and the average absorptivity coefficient α, are constants dictated by the system's design. These constants have been discussed earlier. Additionally, the incident radiation I_0 can also be considered a constant in our system, and its value can be derived mathematically, eliminating the need for external measurement devices. With this in mind, we can rephrase Equation 3.2 as follows:

$$I = k_1 \cdot e^{-k_0 \cdot c} \tag{3.3}$$

Here, k_0 represents the product $\alpha \cdot L$ and k_1 denotes the I_0 of the system. Given varying values of I_0, the transmitted intensity through the gas sample will follow the profile illustrated in Figure 3.6, which resembles an inverse exponential curve. By characterizing the constant k_1 for our system, we can effectively deduce the CO_2 concentration in the sample from the measured transmitted intensity I:

$$c = \frac{ln(I/k_1)}{-k_0} \tag{3.4}$$

It is important to highlight that the LED's gradual aging and degradation, which leads to a reduction in emission intensity, combined with voltage fluctuations from the power source—particularly in battery-operated devices—can adversely influence the stability of radiation incident on the gas sample. Such variations compromise the sensor's accuracy, challenging the assumption that k_1 remains a constant within our system. To address these effects, periodic recalibration of the device is essential to update the value of k_1 and ensure the sensor's optimal operation.

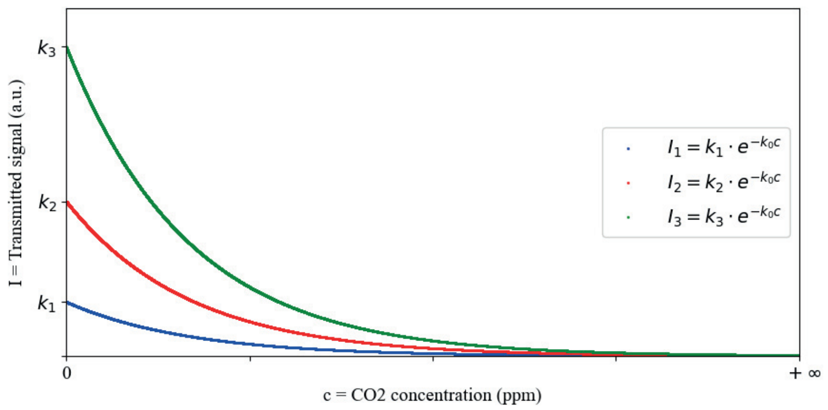


Figure 3.6: Transmitted radiation through a gas sample for different systems with the same optical path length and absorptivity coefficient. Constants k_1, k_2 and k_3 represent different values for the incident radiation I_0.

Sensor Calibration Using Fresh Air To determine k_1, the sensor must be calibrated against a known CO_2 concentration, denoted as c_{ref}. Using

Equation 3.4 we can obtain the value of k_1 as follows:

$$k_1 = \frac{I_{ref}}{e^{-k_0 \cdot c_{ref}}} \tag{3.5}$$

Here, I_{ref} denotes the transmitted intensity that corresponds to the reference concentration c_{ref}. Our method is to capture this value at the output of the software-based quadrature demodulator, in the magnitude detector as illustrated in Figure 3.3. Based on this measurement, the derived value, even though in arbitrary units, is proportional to the infrared radiation transmitted within the sensor cell.

A practical alternative to the laboratory calibration procedure leverages the known fact that the average CO_2 concentration of outdoor air is roughly 400 ppm. Many commercial CO_2 household sensors utilize this assumption for auto-calibration [99], simply by exposing the sensor to fresh outdoor air. While this method is efficient, it may not match the precision achievable in controlled laboratory conditions. Measuring the I_{ref} value in these conditions, and taking c_{ref} as 400 ppm, we can approximately determine the k_1 constant for our system. This then allows us to estimate indoor concentrations using Equation 3.6:

$$c = \frac{ln(\frac{I}{I_{ref}} \cdot e^{-k_0 \cdot c_{ref}})}{-k_0} = c_{ref} + \frac{ln(I_{ref}) - ln(I)}{k_0} \tag{3.6}$$

3.3.4 Analysis of Parameters in the Quadrature Demodulation Process

In parallel to determine necessary equations that will enable our algorithm to calculate the CO_2 concentration of the sample, we also need to put in place a model of the quadrature demodulator. We analyze the different parameters that intervene in the signal demodulation process and their effect on the sensor accuracy. The parameters that have been identified and are worthy of consideration are the measuring range, the sensor's optical path length L, modulation frequency f_m, sampling frequency f_s, system's signal-to-noise ratio (SNR), ADC resolution (bit count), and the LPF's integration interval T.

We employ a Python script to explore these parameters. Structured around a set of nested loops, the innermost loop iterates through the concentration in 1 ppm increments, influencing the light intensity transmitted through the sensor optical path. The exterior loops are designated to traverse the other parameters, specifically the SNR level, sampling frequency, modulation frequency, and quantization bit count.

In every loop iteration, we generate a random white noise signal, integrate it with the transmitted intensity, then proceed to quantize and demodulate the resultant signal, enabling us to estimate the gas concentration. We also evaluate the relative error in every iteration, and upon completing the simulation, we determine the mean error across the whole measuring range. This allows us to contrast the average errors arising from various scenarios that employ different parameter sets. The visualizations we generate via Jupyter Notebook, such as the one depicted in Figure 3.7, include two scenarios with very low SNR levels to assess the immunity of the digital quadrature demodulator against signal noise.

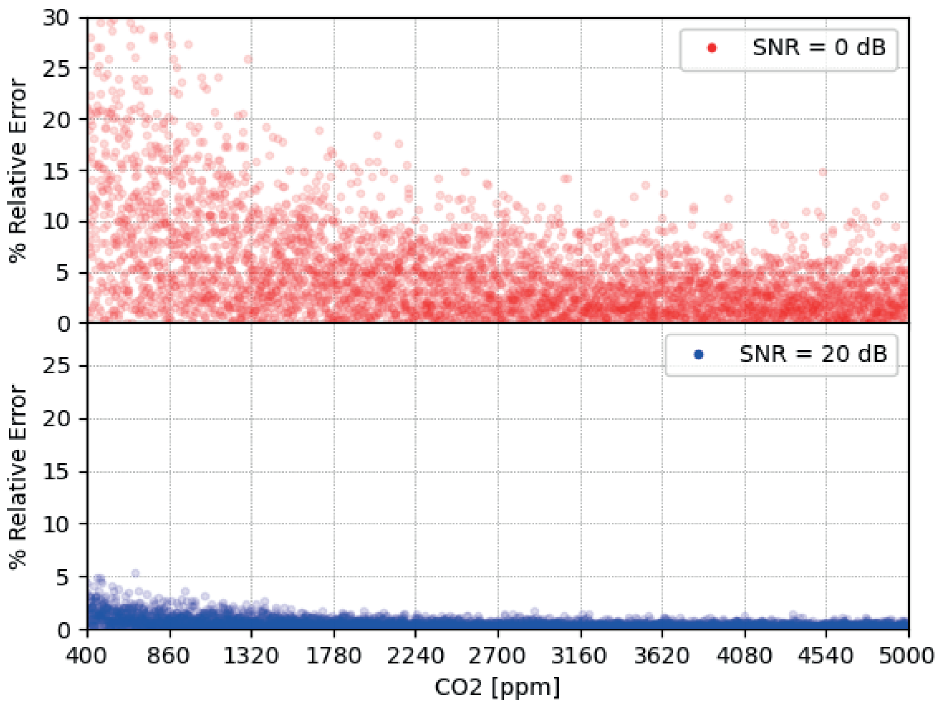


Figure 3.7: Relative error vs. gas concentration obtained during model simulation for a given set of parameters testing two signal-to-noise ratio values: SNR = 0 dB ($\mu = 4.96\%$) and SNR = 20 dB ($\mu = 0.49\%$).

By following, we describe the parameters that we have used in our simulations to better understand the intricacies of the signal demodulation process. The parameter analysis helps to underscore their collective impact on the overall accuracy and performance of the CO_2 concentration measurements. Particularly, we aim to highlight how variations in each parameter might influence the robustness of our system, especially in challenging conditions such as those with low SNR levels. Our ultimate goal is to optimize these parameters to achieve the most reliable and efficient detection of CO_2 concentrations, taking into account both real-world scenarios and potential edge cases.

Measuring range The operational range of a sensor is tailored to its intended application. For indoor air quality monitoring, sensors typically measure up to 0.5% (5,000 ppm) of CO_2. However, atmospheric CO_2 tracking demands sensors with capacities extending up to 1% (10,000 ppm). Distinct sectors necessitate varying sensitivities. Restaurants, breweries, indoor farming, and industrial environments typically deploy a 5% (50,000 ppm) sensor for CO_2 safety oversight. Conversely, specialized domains like modified atmosphere packaging, laboratory environments, cryogenics, and fire suppression systems may require sensors calibrated for 5% to 100% CO_2 concentrations. Building upon our introductory chapter's rationale for selecting indoor air quality monitoring as the target application for our low-power RISC-V processor, our study focuses specifically on this domain, necessitating a measurement range from 400 to 5,000 ppm.

Modulation frequency In our simulation exploration, the values for the modulation frequency f_m were restricted to 2, 8, 16, 128, and 1024 Hz. Our experiments suggest that variations in modulation frequencies do not significantly influence the accuracy of the measurements. Nonetheless, it is important to highlight that our model primarily accounts for white noise with consistent power spectral density, omitting the influence of $1/f$ noise (also known as *pink* noise) that predominates at lower frequencies. The exclusion of this noise from our model was deliberate, given its intricate characterization. However, from a practical standpoint, we recognize the importance of selecting a modulation frequency that is both distant from DC and surpasses the 50/60 Hz interference. As a result, 128 Hz was determined to be an optimal choice for the modulation of the IR LED emission.

Sampling frequency and filter integration time The sampling frequency plays a crucial role in the accuracy of the system. Intuitively, higher frequencies are preferred as they offer more sample points of the signal, enhancing the resolution. However, this is closely tied to the integration time T of the LPF, which we have established as 1 second in our setup.

Should there be a need to reduce this integration time, in order to lessen the measurement latency, a proportional increase in the sampling frequency would be required to ensure the LPF integrates an equivalent number of samples. In our model, we opted for a sampling frequency f_s of 16,384 Hz, a value that is precisely 128 times our chosen modulation frequency. We further realize that extending the integration time, while keeping a constant sampling rate, can enhance the accuracy of low-concentration readings. The trade-off here is that the system benefits from increased samples but at the expense of a longer latency.

ADC resolution Our simulations explored ADC resolutions between 8 and 24 bits, specifically assessing 8, 9, 10, 11, 12, 16, 20, and 24-bit configurations. In our simulation script, floating-point values were converted to signed fixed-point format, designating 5 bits for the integer part (with an additional bit for the sign). This allocation ensures adequate representation, especially when confronted with the amplitude variations introduced by random noise at low SNR values. From our observations, resolutions of 12 bits or above yielded similar accuracy. On the other hand, utilizing fewer bits compromised the results. Consequently, we identified 12 bits as an optimal trade-off and moreover is a resolution frequently found in many cost-effective FPGAs.

System's SNR levels In our simulations, we meticulously varied the Signal-to-Noise Ratio (SNR) to study its impact on system performance. The selected SNR values spanned from -5 to 40 dB, specifically incorporating values of -5, 0, 5, 10, 20, 30, and 40 dB. Our primary objective was to ascertain the immunity of our digital quadrature demodulator in the presence of signal noise. It is crucial to understand that noise can emanate from various sources. Predominantly, thermal noise, originating from the random motion of electrons in a conductor, is ubiquitous in electronic systems. Other potential contributors include pink noise, which dominates at lower frequencies, and electromagnetic interference from external electronic devices. Additionally, imperfect components, manufacturing variabilities, and operational imperfections can introduce noise into the system. Therefore, evaluating our system across a wide range of SNR levels ensures robustness against the myriad noise sources inherent in practical settings.

Optical path length The optical path length of the sensor is a pivotal parameter, directly influencing the sensor's sensitivity. A longer optical path facilitates increased interactions between photons and gas molecules, thereby enhancing measurement accuracy, especially at lower concentration levels. However, selecting the optical path length is not solely about maximizing sensitivity; it is also about adhering to physical dimensions and design considerations. Practical limitations might dictate the feasible length, especially when the goal is to maintain a compact sensor size. Ingenious design strategies, such as the utilization of a reflective waveguide, can help achieve a balance between sensor size and performance. In our simulations, we evaluated various optical path lengths to compare the average error across the operational range under different scenarios. We selected a 7 cm optical path length, as it consistently delivered the best precision for measurements in the 400 to 5,000 ppm range. This choice is further corroborated by the waveguide design study cited in [95], which also identified this length as offering optimal precision within that range.

Summary of the Selected Parameter Values

Table 3.1 provides an overview of the selected parameters for our model simulation following a comprehensive exploration process. These parameters have been optimized to ensure reliable performance when implemented in the benchmark application for our RISC-V processor.

Table 3.1: Model simulation parameters

Parameter	***Value***
Measuring range	400 – 5,000 ppm
Optical path length L	7 cm
Frequency modulation f_m	128 Hz
Sampling frequency f_s	16,384 Hz
Filter integration interval T	1 s
Bit resolution	12 bits
Signal-to-noise ratio (SNR)	20 dB

Additional Observations

Here, it's important to discuss additional findings from various experiments conducted during the system simulation. While our quadrature demodulator employs a sine wave as the reference signal, we explored the alternative of using a square wave to modulate the IR LED emission. The rationale behind this experiment is rooted in the fact that a perfect square wave comprises both a sine wave and an infinite sum of its odd harmonics. The primary frequency, or first harmonic, which accounts for 81% of the total signal power, remains closely correlated with the demodulation signal. As a result, it still facilitates accurate information extraction from the carrier.

It is worth noting that generating a square wave is notably simpler, requiring only the toggling of the digital output that drives the LED on and off, as opposed to the intricate trigonometric calculations necessary for generating a sine wave. Interestingly, our observations revealed that this alteration had an impact on reading accuracy, particularly at lower concentrations. This alteration led to an increase in the relative error of measurements, although it remained within an acceptable range. To counterbalance this effect, we found that doubling the integration time from 1 to 2 seconds helped mitigate the deviation. This trade-off highlights that using a square wave for signal modulation offers implementation simplicity at the expense of sacrificing some degree of accuracy.

Furthermore, it is important to note that our approach did not involve a physical front-end encompassing a gas sensor and an ADC to acquire samples for the digital demodulator. Instead, we opted for a synthetic sample generation method using a Python script. This script simulated the modulation of LED emission, computed the transmitted emission based on the Beer-Lambert law, introduced random white noise, and quantized the resultant signal at the designated sampling rate, mimicking an ADC's behavior.

The pseudo-code in Algorithm 1 describes a procedure that systematically analyzes the impact of different modulation frequencies, sampling frequencies, SNR levels, and CO_2 concentrations on the signal from an IR LED emitter used for CO_2 concentration measurements.

1. Modulation frequency analysis: it starts by looping over a set of predefined modulation frequencies.

2. Sampling frequency analysis: for each modulation frequency, it then loops over a range of sampling frequencies. For a given modulation frequency and sampling frequency, it defines the number of samples based on the sampling frequency and a predefined integration interval. It then calculates the modulated source signal using a square wave and the predefined initial amplitude for the source signal.

3. Demodulation waveform definition and calibration: for demodulation, both sine and cosine waveforms are generated for the current modulation frequency. These waveforms are then used to perform an autocalibration using a reference CO_2 concentration (typically fresh air). This provides a reference measurement value that corresponds

Algorithm 1 Simulation Parameter Analysis

```
 1: start_time = 0, stop_time = 1                              ▷ Units in seconds
 2: interval = stop_time - start_time = 0
 3: for modulation_freq in [2, 8, 16, 128, 1024, 4096] do
 4:     for sampling_frequency in [8192, 16384, 32768, 65536] do
 5:         num_samples ← sampling_frequency × interval
 6:         time_points ← linspace(start_time, stop_time, num_samples)
 7:                                                  ▷ Modulate source LED emission
 8:         chopper ← createSquareWave(modulation_freq, time_points)
 9:         mod_signal ← source_amp × chopper
10:                                                  ▷ Define demodulation waveforms
11:         Sine ← createSineWave(modulation_freq, time_points)
12:         Cosine ← createCosineWave(modulation_freq, time_points)
13:                                          ▷ Compute reference magnitude @400 ppm
14:         ref400PPM ← computeRefVal(mod_signal, Sine, Cosine)
15:                                      ▷ Compute power of modulated source signal
16:         sig_avg_db ← computeSignalPowerInDB(mod_signal)
17:         for target_snr_db in [60, 40, 20, 10, -5] do
18:                                                  ▷ Compute required noise power
19:             noise_avg_db ← sig_avg_db - target_snr_db
20:             noise_power ← convertDBtoWatts(noise_avg_db)
21:             for x from 0 to CO2span with step STEP do
22:                                                  ▷ Compute transmitted intensity
23:                 signal ← applyAbsorption(CO2val[x], mod_signal)
24:                                                  ▷ Noise up the original signal
25:                 noise ← genWhiteNoise(noise_power, len(mod_signal))
26:                 noisy_signal ← signal + noise
27:                                                          ▷ 1st Stage: Mixer
28:                 mixer_I ← Sine × noisy_signal
29:                 mixer_Q ← Cosine × noisy_signal
30:                                                            ▷ 2nd Stage: LPF
31:                 sum_I ← sum(mixer_I)
32:                 sum_Q ← sum(mixwe_Q)
33:                                                      ▷ IQ magnitude detector
34:                 magnitude ← computeMagnitude(sum_I, sum_Q)
35:                                          ▷ Regression of measured magnitude
36:                 CO2ppm ← computeCO2(ref400PPM, magnitude)
37:             end for
38:         end for
39:     end for
40: end for
```

to the I_{ref} in Equation 3.6.

4. Signal power calculation: the average power of the modulated source signal is computed in dB, which is necessary to subsequently generate the noise signal based on the specified SNR level.

5. SNR level analysis: for each combination of modulation and sampling frequencies, the algorithm analyzes a set of signal-to-noise ratios. For each target SNR, it calculates the corresponding average noise in dB.

6. CO_2 Measurement range analysis: for a given SNR, the algorithm simulates the process of measuring CO_2 concentrations over a predefined range. It introduces white noise to the modulated signal to simulate real-world disturbances. This noisy signal is then passed through a mixer, which mixes it with both the sine and cosine demodulation waveforms. The mixed signals are then filtered (through a Low Pass Filter or LPF) to isolate the in-phase and quadrature components. The magnitude of the resulting signal is then computed using the in-phase and quadrature components. Lastly, the algorithm estimates the CO_2 concentration by regressing the measured magnitude against the reference value obtained during autocalibration.

The overall goal of this algorithm is to evaluate the impact of various parameters on the accuracy of CO_2 measurements using an IR LED modulated by different waveforms, in the presence of noise, and across a range of CO_2 concentrations.

3.3.5 Building and Simulation of our RISC-V Application

For the development and testing of our RISC-V application, we utilized SEGGER Embedded Studio (SES) [100], a comprehensive commercial development environment designed specifically for open RISC-V architecture-based devices. SES provided a powerful and streamlined integrated development environment (IDE) that facilitated the building, testing, and debugging of our custom RISC-V applications.

SEGGER Embedded Studio for RISC-V offers a whole C/C++ development system for 32-bit and 64-bit RISC-V microcontrollers and microprocessors. The IDE includes a robust source code editor and a user-friendly build system that enables the seamless building of our applications with a single key press.

Moreover, SES offers a complete RISC-V toolchain for compiling applications tailored to specific ISA extensions. However, for the relatively recent

Zfinx extension, SES does not provide built-in support. Nevertheless, SES allows for the utilization of an external toolchain by specifying the binary's location. In response, we took the initiative to generate our own GNU compiler toolchain for the Zfinx extension, leveraging the publicly available repository [101] maintained by the RISC-V Foundation. Subsequently, we successfully employed this custom toolchain to compile our application, ensuring compatibility and support for the Zfinx ISA extension.

Additionally, one of the most valuable features of SES is its core simulator and debugger, which offers a PC-based, fully functional simulation of the target RISC-V microprocessor. This simulator allowed us to execute and debug parts of our RISC-V application without the need for hardware, significantly speeding up the development process.

Throughout the processor optimization process, we created a new project for each different processor design, with each project targeting distinct RISC-V ISA extension combinations that we wanted to explore. The objective was to simulate the RISC-V application, specifically compiled for the ISA that we had implemented in our processor design.

The task involved translating the Python scripts used in Jupyter Notebook to model and simulate the quadrature demodulation process into C/C++ files. Additionally, we incorporated the sample datasets and the discrete sine and cosine waves used as a reference in the quadrature demodulator, all of which were also generated using Python scripts in Jupyter Notebook.

Our primary objective was to simulate the RISC-V application targeting the specific RISC-V ISA that we implemented in our design. During the simulation, we thoroughly validated the computation of the CO_2 concentration, ensuring that it corresponded with the expected value and fell within the acceptable error margin.

By leveraging the capabilities of SEGGER Embedded Studio, we successfully built, tested, and simulated our RISC-V application. This comprehensive development environment proved to be a valuable tool in verifying the correctness and accuracy of our implementation, ultimately ensuring the reliable performance of our custom RISC-V processor for CO_2 concentration sensing applications.

3.4 Integration of RisCO2 in PULPino: A Comparative Analysis

In this section, we detail the integration process of RisCO2, our custom soft-core RISC-V processor, into the PULPino reference platform. PULPino is an open-source single-core RISC-V SoC developed by the PULP team, designed to be a flexible and efficient platform for various applications. The primary goal of integrating RisCO2 into PULPino was to assess its performance, resource utilization, and power consumption against other well-established RISC-V processors.

3.4.1 Overview of PULPino SoC

PULPino is designed as a lightweight, low-power platform for embedded systems with modularity and versatility in mind. It features a compact yet powerful architecture with several key components:

- **RISC-V Cores:** PULPino supports different RISC-V cores, including the Ri5cy and Zero-riscy cores, each optimized for specific use cases and performance requirements.
- **Memory System:** The SoC includes separate single-port data and instruction RAMs, each with a size of 32 kB. These RAMs are non-contiguous in the address space, providing flexibility for different memory accesses.
- **Boot ROM:** The platform incorporates a boot ROM with a boot loader capable of loading a program via Serial Peripheral Interface (SPI) from an external flash device. This feature simplifies the process of booting the system and loading initial programs.
- **Peripherals:** PULPino includes various peripherals such as UART, GPIO, and SPI Master.
- **Advanced Debug Unit:** The platform supports JTAG and SWD interfaces through its advanced debug unit.
- **SoC Control:** A small and simple APB peripheral provides platform information and allows pad muxing on the ASIC.

The integration of RisCO2 into PULPino aimed to leverage the platform's rich set of features and facilitate a comprehensive comparison with other RISC-V processors.

3.4.2 Comparison with Reference Processors

To conduct a fair and accurate assessment of RisCO2's performance, we followed a systematic approach. We created separate projects for each of the reference processors supported by PULPino, including Zero-riscy, Micro-riscy, Ri5cy, and CV32E40P. This approach allows us to test and evaluate each integration independently, enabling a fair and detailed comparison of the resulting outcomes.

The chosen reference processors present a diverse set of characteristics. On one hand, Zero-riscy and Micro-riscy are well-known processors renowned for their small size and energy efficiency. They are commonly employed in low-power embedded systems, IoT devices, and wearable technology. On the other hand, Ri5cy and CV32E40P are more complex RISC-V cores, designed to cater to more powerful embedded systems with higher performance requirements.

By integrating RisCO2 with these processors, we sought to assess RisCO2's scalability and competence in meeting the demands of complex applications.

3.4.3 Key Considerations in the Integration

The integration process involved important considerations to optimize the PULPino SoC platform for our testing purposes. Firstly, we merged the two separate 32 kB single-port data and instruction RAMs into a single 256 kB BRAM (Block RAM) with true dual-port memory. This unification provides a streamlined and efficient memory structure, enabling seamless access to both data and instruction programs within the unified address space. Secondly, the boot ROM was eliminated from the design as part of our optimization process. By preloading the BRAM with the benchmark program and the sample dataset offline, the boot ROM became unnecessary. This ensured that the necessary code and data were readily available during the processor's initialization, streamlining the system's startup and improving overall efficiency.

Figure 3.8 illustrates a block diagram of the customized PULPino platform, showcasing the customized modifications made for the integration of RisCO2. This adapted configuration allowed us to efficiently compare the performance of RisCO2 with other RISC-V processors within a well-established, versatile SoC framework.

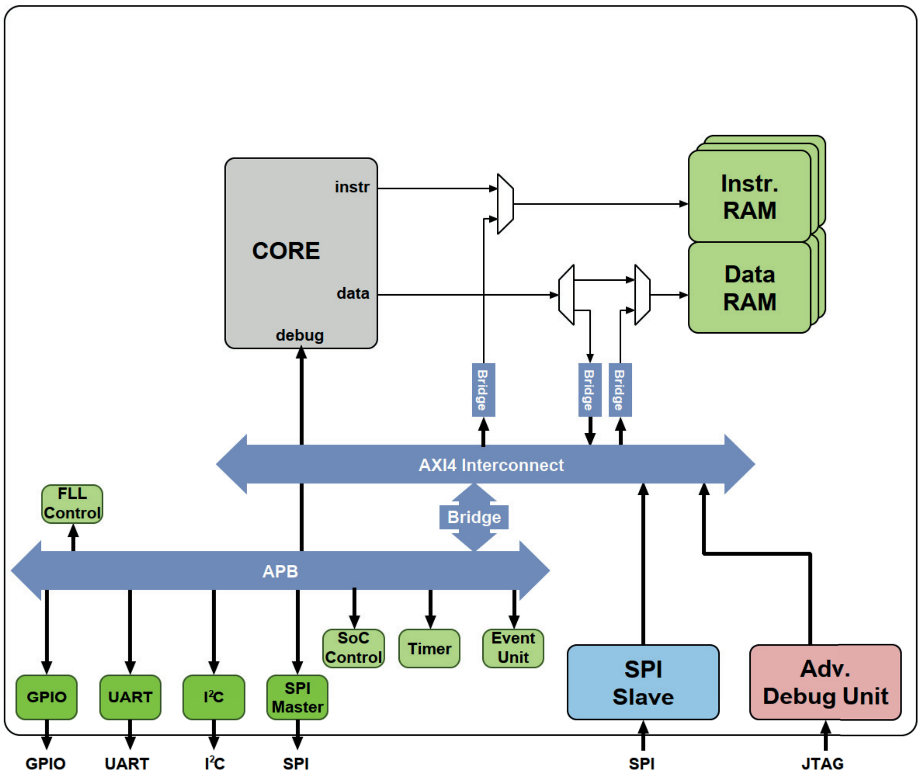


Figure 3.8: Block diagram of the customized PULPino platform used to test the different cores.

3.4.4 Analyzing Performance Metrics

Throughout the integration process, we conducted an in-depth analysis of several performance metrics to comprehensively assess the capabilities of the RisCO2 soft-core processor. Key metrics closely examined include:

- **Resource Utilization:** Thorough evaluations were conducted to gauge RisCO2's efficiency in utilizing FPGA hardware resources, including DSP blocks, memory Block RAM (BRAM), Look-Up Tables (LUTs), and Flip-Flops (FFs). By comparing its resource utilization with other cores, we gained insights into how effectively RisCO2 leverages the available FPGA resources.

- **Performance Analysis:** To assess the overall performance of RisCO2, we designed and implemented a customized application tailored to the quadrature demodulation process. This application performed the extraction of CO_2 concentration levels from the sample dataset that we generated. By employing this application, we were able to measure a series of standard performance metrics and evaluate RisCO2's computational capabilities effectively. The comparisons with other RISC-V cores supported by PULPino offered valuable insights into the relative performance of RisCO2 in practical scenarios.

- **Power Consumption Estimation:** Power efficiency is of paramount importance in many embedded applications. We estimated average energy consumption during the application's runtime to assess RisCO2's energy efficiency. Such comprehensive power estimation enabled us to make informed comparisons with other processors and understand RisCO2's energy efficiency in practical operating conditions.

By considering these crucial factors and conducting a comprehensive evaluation, we gained valuable insights into RisCO2's strengths and areas for further optimization. The integration of RisCO2 into the PULPino platform allowed us to perform a fair and insightful comparison against well-established RISC-V processors. The resulting comprehensive analysis of resource utilization, power consumption, and performance metrics provided valuable data to guide future optimizations and highlighted RisCO2's potential as a viable candidate for a wide range of embedded systems applications, including energy-efficient wireless sensor nodes, thereby contributing to the development of cutting-edge technologies for environmental monitoring and scientific research.

3.5 RTL Simulation and Validation

Before proceeding with the design synthesis, it is essential to verify and validate its functionality and correctness through RTL simulations. RTL simulations are used to model and simulate the digital circuit at the register-transfer level, representing how data is transferred between registers in the hardware.

In this work, we performed RTL simulations using the Vivado IDE from Xilinx, Inc., a popular development environment for FPGA and SoC designs. The RTL simulations were a crucial step in ensuring that our custom RISC-V processor design behaved as intended and adhered to the specifications of the RISC-V ISA that we implemented.

To ensure the correctness and accuracy of our custom RISC-V processor design, we conducted thorough comparisons between the ISA simulation in

SEGGER Embedded Studio and the RTL simulation in Vivado IDE. The purpose of these comparisons was to verify that the processor executed instructions correctly and consistently in both simulations. Any discrepancies or deviations from the expected behavior would be carefully analyzed and addressed, ensuring that the processor design was robust and reliable.

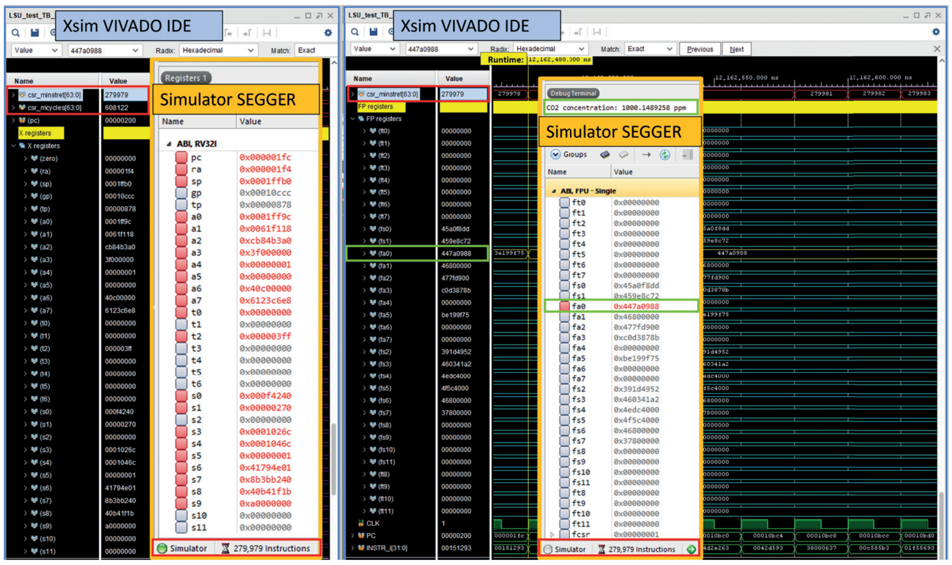
During the comparisons, we inserted breakpoints at specific locations in the program and checked various critical aspects. First, we verified that both simulations matched the program counter (PC) and the number of retired instructions. Additionally, we carefully inspected the content of both the integer `x` and floating-point `f` register files in both simulations. The registers should hold identical values in both simulations, proving that the program's execution in the RTL simulation is correct, at least at the register file level. Another important check involved comparing the content of the RAM memory in the RTL simulation after executing store instructions. This comparison ensured that the stored content was accurate and consistent with the debugger RAM. These validations ensured that the processor progressed through the program as expected, without any discrepancies.

To exemplify this verification process, we present Figure 3.9, which consists of a composite image created by overlaying screenshots from both the Vivado simulator and the SES simulator. This simulation corresponds to the RV32IMF implementation, enabling a thorough comparison and validation of the RTL operation. In the SEGGER simulator, a total of 279,979 instructions were executed, perfectly aligning with the core's CSR counter of retired instructions `csr_minstret`. Furthermore, the values stored in the integer registers (depicted on the left side of the picture) and floating-point registers (on the picture's right side) exhibit precise correspondence between both simulations.

The debug terminal in SES provides the expected output, including the calculated CO_2 value of 1000.16 ppm, which is loaded in the `fa0` register in IEEE754 format (value 0x447a0988). The simulation runtime in Vivado was 12.16 ms, utilizing a frequency of 50 MHz. To obtain the instructions per cycle (IPC), the `csr_minstret` value is divided by `csr_mcycles`. By comparing the data obtained from both simulations, we validate the precise functioning of the RTL design.

In conclusion, besides ensuring the correct behavior of the processor, the important metrics that we extract from the RTL simulations for comparison with other reference processors include:

1. **Number of cycles** and **retired instructions**: These metrics are obtained from the `mcycles` and `minstret` CSRs. By analyzing these



```
Name                   Value
csr_minstret[63:0]     279979
csr_mcycles[63:0]      608122
(pc)                   00000200
X registers
  (zero)               00000000
  (ra)                 000001f4
  (sp)                 0001ffb0
  (gp)                 00010ccc
  (tp)                 00000878
  (a0)                 0001ff9c
  (a1)                 0061f118
  (a2)                 cb84b3a0
  (a3)                 3f000000
  (a4)                 00000001
  (a5)                 00000000
  (a6)                 40c00000
  (a7)                 6123c6e8
  (t0)                 00000000
  (t1)                 00000000
  (t2)                 000003ff
  (t3)                 00000000
  (t4)                 00000000
  (t5)                 00000000
  (t6)                 00000000
  (s0)                 000f4240
  (s1)                 00000270
  (s2)                 00000000
  (s3)                 0001026c
  (s4)                 0001046c
  (s5)                 00000001
  (s6)                 41794e01
  (s7)                 8b3bb240
  (s8)                 40b41f1b
  (s9)                 a0000000
  (s10)                00000000
  (s11)                00000000
```

```
Registers 1
Name     Value
ABI, RV32I
  pc     0x000001fc
  ra     0x000001f4
  sp     0x0001ffb0
  gp     0x00010ccc
  tp     0x00000878
  a0     0x0001ff9c
  a1     0x0061f118
  a2     0xcb84b3a0
  a3     0x3f000000
  a4     0x00000001
  a5     0x00000000
  a6     0x40c00000
  a7     0x6123c6e8
  t0     0x00000000
  t1     0x00000000
  t2     0x000003ff
  t3     0x00000000
  t4     0x00000000
  t5     0x00000000
  t6     0x00000000
  s0     0x000f4240
  s1     0x00000270
  s2     0x00000000
  s3     0x0001026c
  s4     0x0001046c
  s5     0x00000001
  s6     0x41794e01
  s7     0x8b3bb240
  s8     0x40b41f1b
  s9     0xa0000000
  s10    0x00000000
  s11    0x00000000
Simulator   279,979 Instructions
```

```
Name                   Value
csr_minstret[63:0]     279979
FP registers
  (ft0)                00000000
  (ft1)                00000000
  (ft2)                00000000
  (ft3)                00000000
  (ft4)                00000000
  (ft5)                00000000
  (ft6)                00000000
  (ft7)                00000000
  (fs0)                45a0f8dd
  (fs1)                459e8c72
  (fa0)                447a0988
  (fa1)                46800000
  (fa2)                477fd900
  (fa3)                c0d3878b
  (fa4)                00000000
  (fa5)                be199f75
  (fa6)                00000000
  (fa7)                00000000
  (fs2)                391d4952
  (fs3)                460341a2
  (fs4)                4edc4000
  (fs5)                4f5c4000
  (fs6)                46800000
  (fs7)                37800000
  (fs8)                00000000
  (fs9)                00000000
  (fs10)               00000000
  (fs11)               00000000
  (ft8)                00000000
  (ft9)                00000000
  (ft10)               00000000
  (ft11)               00000000
CLK                    1
PC                     00000200
INSTR_i[31:0]          00151293
```

```
Debug Terminal
CO2 concentration: 1000.1489258 ppm
```

```
Name     Value
ABI, FPU - Single
  ft0    0x00000000
  ft1    0x00000000
  ft2    0x00000000
  ft3    0x00000000
  ft4    0x00000000
  ft5    0x00000000
  ft6    0x00000000
  ft7    0x00000000
  fs0    0x45a0f8dd
  fs1    0x459e8c72
  fa0    0x447a0988
  fa1    0x46800000
  fa2    0x477fd900
  fa3    0xc0d3878b
  fa4    0x00000000
  fa5    0xbe199f75
  fa6    0x00000000
  fa7    0x00000000
  fs2    0x391d4952
  fs3    0x460341a2
  fs4    0x4edc4000
  fs5    0x4f5c4000
  fs6    0x46800000
  fs7    0x37800000
  fs8    0x00000000
  fs9    0x00000000
  fs10   0x00000000
  fs11   0x00000000
  ft8    0x00000000
  ft9    0x00000000
  ft10   0x00000000
  ft11   0x00000000
  fcsr   0x00000001
Simulator   279,979 Instructions
```

Figure 3.9: Image composite made with screenshots of the simulations performed with Vivado to test the RTL logic and SEGGER Embedded Studio to simulate the compiled RISC-V application.

counters, we can infer the number of instructions per cycle (IPC), a crucial metric for performance comparison.

2. **Application execution time**: This metric allows us to estimate the energy consumption during the application runtime. By multiplying the estimated power by the execution time, we gain valuable insights into the processor's energy efficiency.

Once the RTL simulation aligns with the RISC-V application simulation, we can confidently proceed to synthesis and implementation, and annotate FPGA resource utilization for each design version. This verification process, bolstered by the comparison of key metrics, assures the reliability and effectiveness of our custom RISC-V processor for CO_2 concentration sensing applications.

3.6 Synthesis and Implementation

During the synthesis phase, we followed Vivado's default strategy to convert our RTL (Register-Transfer Level) design into a gate-level netlist. For implementation, we chose the `Performance_ExplorePostRoutePhysOpt` strategy. This strategic option integrates physical optimization techniques and employs diverse algorithms for optimizing placement and routing. The objective was to enhance the final implementation's performance and ensure it meets the required timing constraints.

For a clock frequency setting of 25 MHz, all the different designs passed the timing analysis, validating their functionality and compliance with timing specifications.

The critical metrics we gathered after implementation included FPGA resource utilization, such as LUTs, FFs, DSPs, and BRAM. These metrics provide valuable insights into the efficient utilization of FPGA resources.

Additionally, we performed power estimation using Vivado's power analysis tool. However, it is essential to note that the power estimation at this stage lacks statistical information regarding the switching activity of transistors. Consequently, it offers a medium level of confidence. To achieve a more precise power analysis, we must run a post-implementation timing simulation to generate a Switching Activity Interchange Format (SAIF) file. This file contains essential switching activity information, which we can then utilize for a more accurate and reliable power consumption estimation.

3.7 Post-Implementation Timing Simulation and Power Analysis

After completing the implementation, we conducted timing simulations to obtain precise power consumption estimations based on the Switching Activity Interchange Format (SAIF) information.

As explained in Section 2, the dynamic power consumption P_{dyn} in CMOS circuits is typically represented by the Equation 2.1. In this equation, α stands for the probabilistic switching activity factor, C_T denotes the total capacitance of the circuit, f_{clk} is the switching frequency, and V represents the supply voltage. Our primary focus is on measuring the total energy consumption during the application's runtime, as defined by Equation 3.7, where P_{sta} represents the static power consumption.

$$E = \int_t P_{dyn} + P_{sta} \tag{3.7}$$

To obtain a detailed estimation of P_{dyn} and E, we need to capture the switching activity of all gates in our circuit while executing our application. Vivado IDE allows capturing this activity during post-implementation timing simulation, generating SAIF files used to provide detailed power estimates for different regions of the FPGA fabric. For certain parts of the design, Vivado can acquire accurate switching activity, while for others like memories, it resorts to a probabilistic approach to estimate power consumption.

To perform the analysis efficiently, capturing the switching activity for the entire demodulation algorithm's runtime is unnecessary. Instead, we set the simulation interval to 1 ms, allowing us to capture multiple iterations of the main loop within the demodulation algorithm. This time setting yields a fair average power value that can be extrapolated to the entire program execution, as over 95% of the program runtime occurs within the demodulation loop.

Furthermore, the power simulation tool annotated over 92% of the nets in all different SoC implementations accurately, using probabilistic computations for the remaining nets. This methodology allowed us to obtain precise power estimates for each of the five RISC-V processors implemented in the PULPino SoC, providing valuable insights into their respective energy efficiency and resource utilization.

Chapter 4

Experimental Results and Discussion

In this chapter, we delve into the comprehensive experimental evaluation that led to the development of RisCO2, a processor designed with a focus on energy efficiency. The first stage focuses on selecting standard ISA extensions to optimize the RISC-V architecture. The second stage involves more granular adjustments, including specialized ISA extensions and the removal of unnecessary logic to further enhance energy efficiency. Following these design phases, we incorporate RisCO2 into the reference SoC platform, PULPino, to compare its performance against other processors under the same testing environment.

Through a process of careful experimentation and iterative design, this chapter aims to shed light on how RisCO2 compares favorably in terms of energy efficiency with other processors in similar benchmarking conditions.

4.1 Stage One: Initial Design and ISA Exploration

In the initial stage of our design exploration, we developed a simplified, self-contained system that included only the processor and memory. This approach provided a controlled testbed for evaluating the impact of different architectural variations on energy efficiency, each integrating various standard extensions of the baseline ISA.

It is important to note that the results and considerations presented here apply specifically to the 5-stage pipelined processor design, as detailed in Section 3.1.2. This processor version evolved from our initial single-cycle design and serves as the foundation for the ISA exploration conducted in this first stage. The findings offer critical insights into the performance metrics and energy efficiencies of this more complex pipeline architecture, setting the stage for further design enhancements and optimizations.

This stage also served as an experimental arena for testing different software adaptations of the target application. This included using software emulation for fixed-point and floating-point arithmetic to acquire initial insights into which architectural features and software adjustments would best serve our application.

At the hardware level, we examined giving support for specific operations, such as integer multiplication, division, and floating-point calculations. We also assessed the potential advantages of integrating DSPs available in the FPGA to improve energy-efficient performance. This initial stage laid the groundwork for the more detailed design explorations that followed.

The performance outcomes of our design exploration are comprehensively detailed in Tables 4.1 and 4.2. The first column in each table provides each unique design with a version number, making it easier for subsequent visual analysis which will be presented in a graph later. Following this, in Table 4.1 the next four columns specify key design choices that characterize each experimental setup. These columns include: the implemented RISC-V architecture variant, either RV32I, RV32IM, or RV32IMF; the numerical format used by the algorithm, which is either single-precision floating-point or fixed-point; the kind of support allocated for arithmetic instructions, whether they are emulated in software or executed through dedicated hardware support; and lastly, the computational strategy employed for trigonometric functions, which is either based on C-Runtime libraries or leverages lookup tables. The remaining columns in the table present the measured FPGA resource utilization metrics, including Lookup Tables (LUT), Flip-Flops (FF), and Digital Signal Processors (DSP).

In Table 4.2, the columns are organized to present performance metrics derived from the combination of the design variables previously outlined. The metrics include: the total number of instructions, denoted as $N.Instr. \times 10^6$, and the clock cycles required by the benchmark application to both demodulate the signal and calculate the CO_2 concentration; the Instructions Per Cycle (IPC), an essential metric for assessing computational efficiency; the total execution time, termed as *Exec. Time*, which indicates the total time in seconds required for the benchmark program to complete its tasks; the processor's dynamic power and energy consumption, as estimated by the simulation's power analysis tool, offers quantitative insights into its power efficiency. The energy consumption is computed by multiplying the dynamic power by the application's execution time. The power analysis tool also provides the overall non-disaggregated static power consumption P_{sta}, which accounts for the total static power consumption of the FPGA chip, measured at 0.099 W; and finally, the greenness factor represents the computational performance per unit of power consumed, measured in GOPS/W (Giga Operations Per Second per Watt). It serves as an index of how energy-efficient a processor is while performing its tasks.

Version #	**ISA**	**Variable Type**	**Support FloatP/FixP**	**Trigo. Funct.**	**LUT**	**FF**	**DSP**
1	RV32I	SP float	SW Emul.	C-Rnt.	2661	2158	2
2	RV32I	SP float	SW Emul.	C-Rnt.	2695	2158	0
3	RV32I	SP float	SW Emul.	LUT	2661	2158	2
4	RV32I	SP float	SW Emul.	LUT	2695	2158	0
5	RV32I	Fixed-p.	SW Emul.	C-Rnt.	2661	2158	2
6	RV32I	Fixed-p.	SW Emul.	C-Rnt.	2695	2158	0
7	RV32I	Fixed-p.	SW Emul.	LUT	2661	2158	2
8	RV32I	Fixed-p.	SW Emul.	LUT	2695	2158	0
9	RV32IM	SP float	SW Emul.	C-Rnt.	3010	2290	10
10	RV32IM	SP float	SW Emul.	C-Rnt.	4637	2290	0
11	RV32IM	SP float	SW Emul.	LUT	3010	2290	10
12	RV32IM	SP float	SW Emul.	LUT	4637	2290	0
13	RV32IM	Fixed-p.	SW Emul.	C-Rnt.	3010	2290	10
14	RV32IM	Fixed-p.	SW Emul.	C-Rnt.	4637	2290	0
15	RV32IM	Fixed-p.	SW Emul.	LUT	3010	2290	10
16	RV32IM	Fixed-p.	SW Emul.	LUT	4637	2290	0
17	RV32IMF	SP float	HW FPU	C-Rnt.	7085	4188	12
18	RV32IMF	SP float	HW FPU	C-Rnt.	9057	4193	0
19	RV32IMF	SP float	HW FPU	LUT	7085	4188	12
20	RV32IMF	SP float	HW FPU	LUT	9057	4193	0

Table 4.1: ISA support, number format, trigonometric function strategy, and FPGA resource utilization for different processor design variations.

In a close examination of the data provided in Tables 4.1 and 4.2, a number of pivotal observations come to light. First and foremost, configurations utilizing the RV32IM architecture consistently outperform their

RV32I-based counterparts. The key to this improved performance lies in the additional M extension in the RV32IM architecture, which provides hardware-level support for integer multiplication and division. This eliminates the need for software emulation of these operations, thereby reducing the total instruction count and, consequently, lowering both execution time and energy consumption.

Secondly, within the context of RV32I and RV32IM processors—which do not natively support hardware-accelerated floating-point arithmetic—we can see that fixed-point emulation consistently surpasses emulated single-

Ver. #	**N. Instr. $\times 10^6$**	**Clock Cyc.$\times 10^6$**	**IPC**	**Exec. Time [s]**	**P$_{dyn}$ [W]**	**Energy [mJ]**	G **[GOPS/W]**
1	117.7	144.0	0.82	2.88	0.036	104	1.1
2	117.7	144.0	0.82	2.88	0.029	83.5	1.4
3	10.2	13.2	0.77	0.263	0.038	10.0	1.0
4	10.2	13.2	0.77	0.263	0.037	9.74	1.0
5	39.9	46.8	0.85	0.936	0.042	39.3	1.0
6	39.9	46.8	0.85	0.936	0.043	40.3	1.0
7	6.62	7.53	0.88	0.151	0.037	5.57	1.2
8	6.62	7.53	0.88	0.151	0.036	5.42	1.2
9	38.6	49.7	0.78	0.994	0.062	61.6	0.6
10	38.6	49.7	0.78	0.994	0.177	176	0.2
11	4.59	6.3	0.72	0.127	0.051	6.46	0.7
12	4.59	6.3	0.72	0.127	0.094	11.9	0.4
13	1.99	2.03	0.98	0.041	0.074	3.82	0.7
14	1.99	2.03	0.98	0.041	0.221	11.4	0.2
15	1.38	1.45	0.95	0.029	0.043	1.25	1.1
16	1.38	1.45	0.95	0.029	0.110	3.19	0.4
17	2.80	7.00	0.40	0.140	0.058	8.12	0.3
18	2.80	7.00	0.40	0.140	0.127	17.8	0.2
19	0.28	0.61	0.46	0.012	0.037	0.45	0.6
20	0.28	0.61	0.46	0.012	0.056	0.68	0.4

Table 4.2: Performance metrics, dynamic power, energy consumption, and greenness factor of various processor design variations, all evaluated at a clock frequency of 50 MHz. The total static power consumption for the FPGA chip is a constant 99 mW across all design variations.

precision floating-point calculations in both speed and energy efficiency. This is predominantly due to the dramatic decrease in the total number of required instructions. For instance, a comparison between version #9, which utilizes floating-point emulation and consumes 61.6 mJ, and version #13, which employs fixed-point emulation and consumes a mere 3.82 mJ, illustrates an impressive energy saving—reducing consumption by a factor of 1/16. Both versions rely on C-runtime libraries for trigonometric operations. Even more compelling is that further reductions in execution time and energy consumption can be achieved by replacing runtime trigonometric calculations with a lookup table. This results in a notable algorithmic speed-up, primarily due to the substantial reduction in the number of code instructions. Importantly, this lookup table is stored in the data memory, explaining why the number of LUTs remains constant in Table 4.1 across those specific design variations.

Lastly, the utilization of DSP48E1 slices in the FPGA for the integer multiplication module—in both the RV32IM and RV32IMF architectures—leads to an observable reduction in power consumption. This is an expected outcome given that DSP48E1 slices are highly specialized and efficient hardware modules for integer multiplication tasks. Combining all these improvements, we succeeded in developing an RV32IMF-based system that boasts a remarkably low energy consumption of just 1.25 mJ in version #15, as opposed to the 104 mJ observed in version #1.

In our pursuit of minimizing energy consumption further, we explored the impact of integrating an FPU directly into the execution stage of the processor pipeline. These configurations correspond to the RV32IMF versions listed in the table. Our analysis reveals that configuration #19 stands out with the lowest energy consumption of 0.45 mJ. This represents a remarkable 64% reduction in energy consumption when compared to version #15, which already had a notably low energy consumption.

However, it is crucial to acknowledge the trade-offs involved. The integration of the FPU into the processor's execution stage results in a substantial increase in FPGA resource utilization—specifically, a 135% increase in the number of look-up tables (LUTs) and an 83% rise in flip-flops (FFs). Therefore, while the inclusion of the FPU significantly boosts energy efficiency, it does so at the cost of increased hardware resource consumption.

Understanding the relationship between hardware design choices and the greenness factor G (Equation 2.6) can shed light on how to optimize energy consumption. For instance, designs utilizing DSPs (those versions with odd numbering) require fewer LUTs compared to their non-DSP counterparts, thereby reducing the overall capacitance C_T. This decrease in C_T leads to a reduction in dynamic power consumption, subsequently improving the greenness factor. Conversely, incorporating an FPU increases C_T and thus the dynamic power. This design choice worsens G due to an increase in power consumption coupled with a decrease in IPC, attributed to the longer latency associated with floating-point operations. However, it is worth noting that this lower IPC is offset by the requirement for fewer instructions to complete the same tasks, thus completing computations more quickly.

When dynamic power, energy, and greenness are considered in tandem, they offer a comprehensive view of a processor's efficiency. For instance, a processor may have low dynamic power consumption but also offer reduced performance, leading to a mediocre greenness factor. Alternatively, a processor with higher dynamic power could significantly outperform in computational tasks, resulting in an elevated greenness factor.

In summary, version #19 demonstrates that integrating an FPU into the processor pipeline can achieve exceptional energy savings, albeit with increased demands on FPGA resources and a worse greenness factor compared to a more simple processor that only supports the RV32I ISA. This finding presents an interesting design choice for subsequent design explorations, where the benefits in energy savings need to be carefully weighed against the increased hardware footprint.

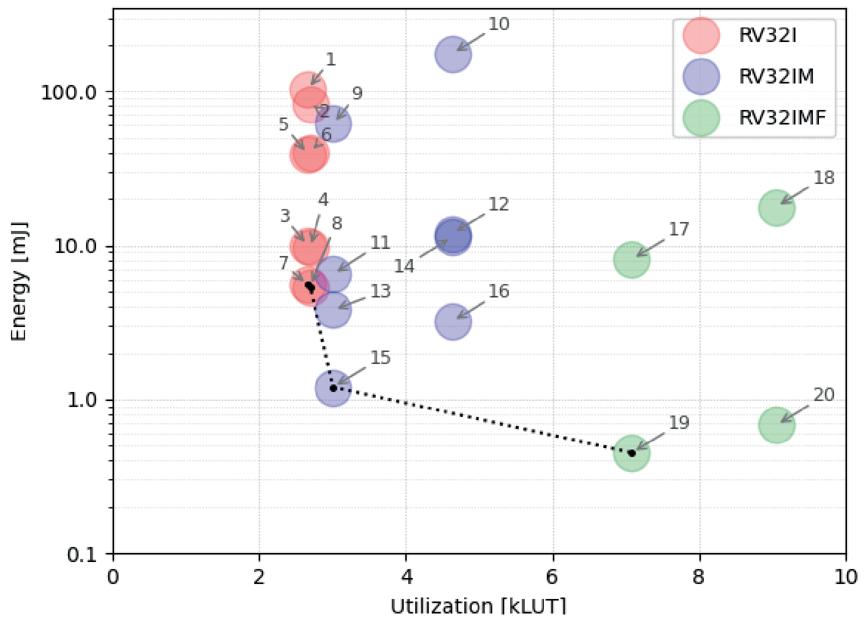


Figure 4.1: Processor's resource utilization versus energy consumption and Pareto frontier line.

In order to facilitate a more comprehensive understanding of the key data listed in Tables 4.1 and 4.2, we have created Figure 4.1, which plots energy consumption against FPGA resource utilization for each processor design variation. The Pareto frontier line has been superimposed on the graph to identify the configurations that represent the most optimal trade-offs between energy efficiency and resource utilization. This Pareto analysis serves as a valuable guide for making design choices, revealing the point(s) where incremental gains in energy efficiency start to demand disproportionately high resource investment.

It is worth mentioning that the design versions with odd numbering make use of DSP blocks, leading to a decrease in LUT utilization compared to their even-numbered counterparts that do not utilize DSPs. An interesting observation can be made regarding the RV32IMF architecture when integrated with a hardware FPU. While it shows a dramatic reduction in the processor's energy consumption (down to 0.45 mJ with version #19), it also exhibits a considerable increase in FPGA resource utilization (LUTs increased to 7085 and FFs to 4188 plus 12 DSPs). This scenario highlights the classic engineering dilemma of a trade-off, where achieving higher energy efficiency comes at the cost of increased resource requirements. As such, the decision to integrate an FPU should be carefully weighed based on specific application requirements and resource constraints.

4.2 Stage Two: Fine-Grained Design Optimization

After successfully identifying a top-performing design during stage one, stage two shifts towards a fine optimization of this chosen architecture. The aim is to optimize both specialized application requirements and energy efficiency. For this fine-tuning stage, we selected the RV32IMF version #19 design with an energy consumption of 0.45 mJ.

The specific optimizations undertaken are comprehensively detailed in the Methodology section, which can be found under Section 3.2.1. For the sake of clarity and to facilitate a better understanding of the results yielded by these particular actions, a summarized review of these optimizations is provided below:

1. E Extension Support: This reduces the integer register count from 32 to 16, effectively halving the storage requirement for integer variables.

2. Zfinx Extension Support: The introduction of this extension eliminates

the necessity for a dedicated floating-point register file by permitting the shared use of the integer register file for both integer and floating-point data. Overall, these enhancements reduce the register file size by a quarter when compared to an RV32IMF setup.

3. Removal of Unused Arithmetic Instructions: Through a targeted analysis, we identified integer arithmetic instructions from the ISA that were not utilized in our specific application. These included integer multiplication, division, and remainder instructions like `mul`, `mulh[u|su]`, `div[u]`, and `rem[u]`. By removing the hardware logic associated with these unused instructions, we were able to reduce the FPGA resource utilization further. This action also resulted in the effective elimination of the M extension, bringing the design into tighter alignment with the application-specific requirements of our project.
4. Control and Status Register Logic Removal: Unnecessary control and status register management instructions (`csrrw[i]`, `csrrs[i]`, and `csrrc[i]`), along with unused shift and comparison instructions (`sra`, `slti`, and `slt`), were pruned from the architecture.
5. Removal of Support for Misaligned Memory Access: To further streamline resource utilization, we eliminated all logic designed to handle misaligned memory accesses. This simplification of the load-store unit is based on the assumption that the compiler will generate only 4-byte aligned memory addresses.

Step	**ISA**	**LUT(%^{1})**	**FF(%^{1})**	**DSP(%^{1})**	**Time (%^{1}) [ms]**	**Energy (%^{1}) [mJ]**
Initial	RV32IMF	7,085	4,188	12	12.2	0.45
1, 2	RV32EM_Zfinx	6,518 (0.92)	2,693 (0.64)	12 (1.0)	10.3 (0.84)	0.42 (0.93)
3	RV32E_Zfinx	5,307 (0.75)	2,545 (0.61)	2 (0.17)	10.3 (0.84)	0.32 (0.71)
5	RV32E_Zfinx	5,126 (0.72)	2,478 (0.59)	2 (0.17)	10.3 (0.84)	0.31 (0.69)
4 (RisCO2)	RV32E_Zfinx	4,692 (0.66)	2,293 (0.55)	2 (0.17)	10.3 (0.84)	0.28 (0.62)

^{1} % are calculated as the factor between the current result and the initial one.

Table 4.3: Comparative summary of design improvement steps: evaluating the impact on FPGA resource utilization, demodulation algorithm runtime, and processor's energy consumption at a clock frequency of 50 MHz.

Through the careful application of these optimizations, we achieved significant improvements in our initial RISC-V RV32IMF processor. This led to the development of RisCO2, a 5-stage, single-issue, in-order processor specifically tailored for energy-efficient operations. This finalized version is based on the RV32E_Zfinx instruction set and is intended for deployment in NDIR CO_2 sensors that require signal demodulation for gas concentration detection. The benefits of these design adjustments are manifest in reduced energy consumption and resource utilization, compared to our original processor.

Table 4.3 provides a comprehensive step-by-step breakdown of the various improvements made. It highlights changes in FPGA resource utilization, the algorithm's execution time, and the processor's energy consumption. The results particularly underline the remarkable reduction in energy consumption from the initial design to the final RisCO2 configuration, highlighted by a drop from 0.45 mJ to 0.28 mJ. The table also indicates a consistent decrease in LUT and FF utilization across the various design iterations, culminating in an optimized version with improved energy efficiency. It is noteworthy that the time required for the demodulation algorithm remains largely unchanged, indicating that the optimizations have not compromised the processor's runtime performance. This indicates that the removed logic was indeed superfluous, serving only to waste energy rather than contribute to system functionality. Interestingly, in contrast to our initial design exploration results shown in Tables 4.1 and 4.2, reduced resource utilization in this case did not lead to longer execution times.

The graphical representation presented in Figure 4.2 provides a visual summary of the cumulative performance gains achieved through the sequence of design iterations delineated in Table 4.3. The plot reveals a nearly directly proportional relationship between reductions in energy consumption and decreases in LUT utilization. This suggests that optimizing the hardware utilization effectively contributes to lowering the processor's energy demands.

4.3 Integration and Comparative Analysis with PULPino SoC

Once we finished our design exploration process, the primary objective was to evaluate the real-world performance and efficiency of our optimized RisCO2 processor. To achieve this, RisCO2 was integrated into an existing SoC framework, specifically the PULPino platform. PULPino is a well-known SoC platform that is often used as a benchmark in processor design and performance studies. This integration served a dual purpose: first, it provided a standard environment for experimental validation of RisCO2;

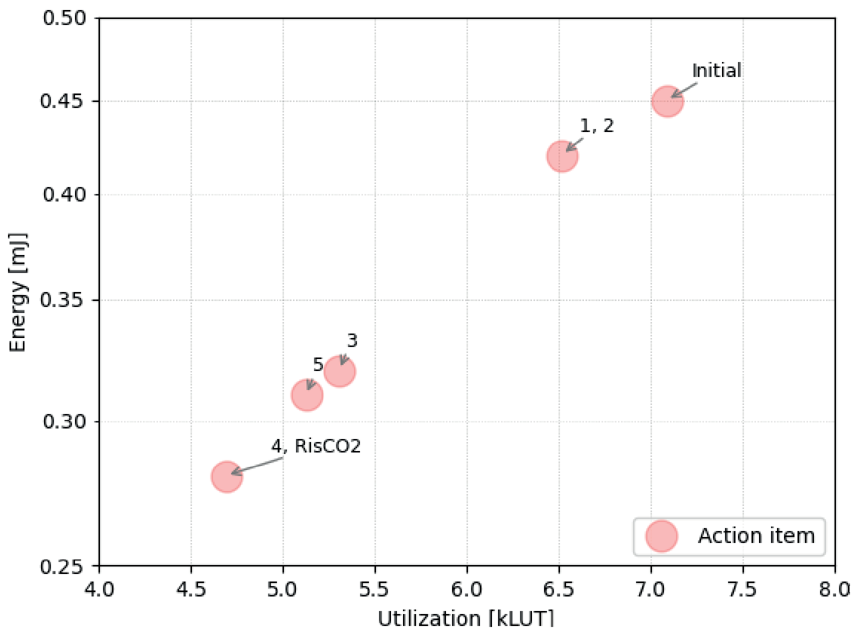


Figure 4.2: Correlation between LUT utilization and the energy consumption of the processor core, following the sequential implementation of optimization actions listed above.

second, it allowed for direct comparison with other reference processors that have been implemented on the PULPino platform.

The integration process involved replacing the existing CPU core in the PULPino SoC with the RisCO2 processor. To facilitate seamless integration, we made additional adjustments to achieve pinout compatibility within the core region of the PULPino system. Specifically, these adjustments involved adapting the memory bus to align with the three-level handshake interface (request, grant, valid) used by PULPino. This allowed RisCO2 to operate under the same conditions as the reference processors, ensuring that any performance differences were due to the processor design itself, rather than variations in the test environment.

After integration, RisCO2 underwent testing using the benchmark application outlined in Chapter 3 (Methodology). This application, specifically designed to demodulate a set of data samples and determine the CO2 concentration, offers an assessment of the processor's performance under its intended operational conditions. For a fair and direct comparison, the same benchmark was applied to all other reference processors integrated into the PULPino SoC. Special attention was focused on estimating power and energy consumption metrics from the post-implementation timing simulations, given their critical importance for RisCO2's intended application.

Core	ISA	Pipeline stages	LUT	FF	DSP	Variable Type
Micro-riscy	RV32E	2	2,225	1,276	0	Fixed-point
Zero-riscy	RV32IM	2	3,171	1,928	1	Fixed-point
Ri5cy	RV32IMF	4	11,912	4,249	8	SP Floating
CV32E40P	RV32IM_Zfinx	4	9,072	2,553	7	SP Floating
RisCO2	RV32E_Zfinx	5	4,889	2,354	2	SP Floating
RisCO2$^{(*)}$	RV32E_Zfinx	5	4,795	2,031	2	SP Floating

$^{(*)}$ Final version of RisCO2 after optimizing the FPU pipeline.

Table 4.4: Comparison of RisCO2 with other RISC-V reference processors in terms of supported ISA, pipeline stages, and FPGA resource utilization.

Tables 4.4 and 4.5 provide a comparative performance analysis of five RISC-V processors: RisCO2, Zero-riscy, Micro-Riscy, Ri5cy, and CV32E40P. These processors are evaluated based on several metrics, including FPGA resource utilization (i.e., LUT, FF, DSP), number representation in the application algorithm (either fixed-point or single-precision floating-point), and computational performance metrics such as the number of instructions and clock cycles required for signal demodulation and CO_2 concentration calculation. Additionally, the table reports the IPC, execution time for the demodulation algorithm, as well as the dynamic power and energy consumption of each processor. All measurements are taken at a clock frequency of 25 MHz.

Core	N.Instr. $\times 10^6$	Clock Cyc.$\times 10^6$	IPC	Exec. Time [ms]	P_{dyn} [mW]	Energy [mJ]	G [GOPS/W]
Micro-riscy	7.244	9.328	0.78	373.13	15	5.60	1.29
Zero-riscy	1.380	1.693	0.82	67.71	20	1.35	1.02
Ri5cy	0.280	0.379	0.74	15.15	52	0.79	0.36
CV32E40P	0.251	0.318	0.79	12.73	49	0.62	0.40
RisCO2	0.252	0.518	0.49	20.71	14	0.29	0.87
RisCO2$^{(*)}$	0.252	0.387	0.65	15.47	5	0.08	3.35

$^{(*)}$ Final version of RisCO2 after optimizing the FPU pipeline.

Table 4.5: Comparison of RisCO2 with other RISC-V reference processors in terms of performance, dynamic power, energy consumed by the processor, and greenness G for a clock frequency of 25 MHz.

The tables include an additional set of results for RisCO2. This is a noteworthy inclusion driven after its direct comparison with CV32E40P, which is an architecture closely resembling RisCO2 and also uses the FPnew FPU IP. While RisCO2 excelled in terms of energy efficiency, it lagged in IPC, with a score of 0.49 compared to CV32E40P's 0.79. These findings led us to undertake a targeted design revision aimed at IPC improvement. The initial lower IPC in RisCO2 can be traced back to the use of superfluous pipeline registers within the FPU. Initially, these were included to meet timing requirements when operating at a clock frequency of 50 MHz. However, we reduced the clock frequency to 25 MHz because we could not achieve timing closure with Ri5cy for performance comparison. At this lower frequency, we found the additional registers to be not just unnecessary, but actively detrimental to performance. After their removal, both the application runtime and dynamic power consumption saw significant reductions—from 19 mW to 5 mW. Given the extent of these improvements, we felt compelled to include these last-minute results in the current work.

From the data presented in Table 4.5, it is evident that RisCO2 exhibits superior energy efficiency compared to its counterparts. Specifically, it demonstrates a 90% and 87% reduction in energy consumption relative to the next best-performing processors, Ri5cy and CV32E40P, respectively. This significant energy savings is particularly noteworthy given that RisCO2 also exhibits lower FPGA resource utilization when compared to these two processors, making it a more economical choice for energy-sensitive applications.

In contrast, Micro-riscy, despite its low resource utilization and power consumption comparable to that of RisCO2, consumes over 70 times more energy. This discrepancy is primarily due to its considerably longer execution time, which can be attributed to its lack of specialized functional units capable of performing complex arithmetic operations beyond basic addition and subtraction. This limitation hampers its computational efficiency and results in inflated energy consumption figures. Conversely, Ri5cy, which has the highest resource utilization among the five processors, consumes nearly 10 times more energy than RisCO2. It is worth noting, however, that Ri5cy and CV32E40P do offer better instruction throughput or IPC rate. While this can indicate more performant cores in terms of computational speed, it comes at the cost of significantly increased energy consumption and resource utilization, making them less suitable for this energy-constrained specific application.

Notably, based on the table, the *greenness* factor (G) for RisCO2 is 3.35 GOPS/W, which is significantly higher than the other listed RISC-V reference processors. To put this into perspective, the next highest *G* factor is for Micro-riscy at 1.29 GOPS/W, which is more than 2.5 times less efficient in terms of greenness compared to RisCO2.

The *G* factor represents a form of energy efficiency, quantifying the number of Giga Operations Per Second per Watt (GOPS/W) that a processor can achieve. A higher *G* factor means that the processor can perform more computations for a given amount of power, making it more energy-efficient. Certainly, the high greenness factor of RisCO2 has several interconnected implications that make it a compelling choice across a range of applications. Its energy efficiency not only makes it suitable for battery-powered or energy-sensitive devices but also reduces its environmental impact through lower heat generation and electricity consumption, contributing to a reduced carbon footprint. A higher *G* factor also implies a more "green" or environmentally friendly processor, which could be an appealing factor in markets or sectors that prioritize sustainability. Finally, its high level of energy efficiency makes it particularly well-suited for IoT and edge computing applications, which often operate under constrained power conditions.

Figure 4.3 extends the analysis of Figure 4.2 by incorporating the reference processors under scrutiny in this study. RisCO2 occupies an advantageous position within this multi-dimensional design space, combining beneficial features from different architectures.

On one front, RisCO2 aligns itself with the design philosophies of Micro-riscy and Zero-riscy, both processors that are engineered for minimal resource utilization and low power consumption. This characteristic positions RisCO2 as a resource-efficient choice, particularly beneficial for applications where conserving FPGA resources is paramount.

Conversely, RisCO2 also embodies some of the performance-oriented attributes of Ri5cy and CV32E40P, processors that are geared towards extracting the highest possible computational performance within a fixed power envelope. This multifaceted design approach makes RisCO2 versatile, and capable of balancing both power efficiency and computational performance.

It is important to clarify a particular point highlighted in the graph. The initial version of RisCO2 already outperforms both Ri5cy and CV32E40P in terms of efficiency despite being an early-stage design. One contributing factor is its greater usage of DSP blocks—12 in total—compared to the other reference processors. This architectural choice not only minimizes the use of LUTs and FFs but also capitalizes on the specialized DSP48 blocks available in Xilinx FPGAs. These DSP48 blocks are inherently more

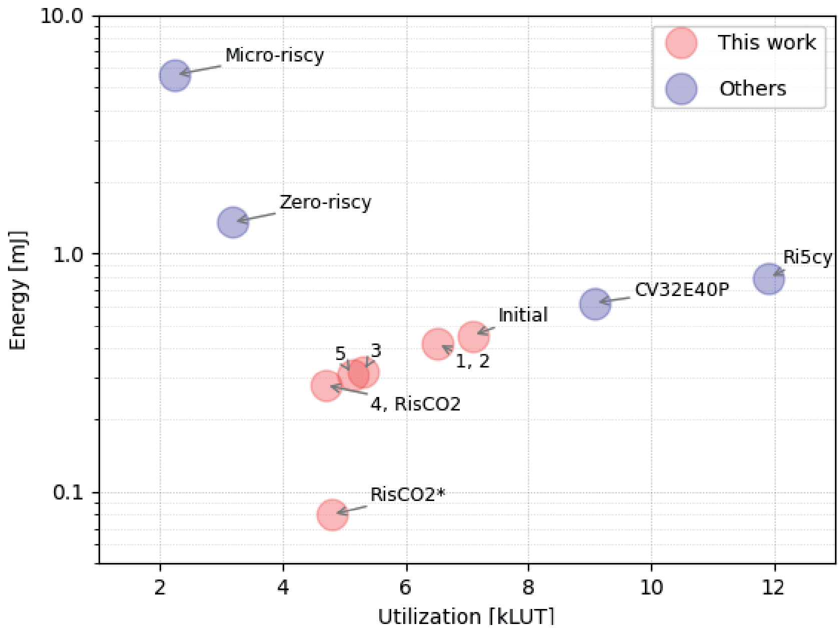


Figure 4.3: LUT utilization and energy consumption of the different cores tested in this study.

power-efficient, thereby enhancing the overall energy efficiency of RisCO2.

The power analysis report generated by Vivado serves as an insightful resource for understanding the power dynamics within our system. It not only estimates the total power consumption for the complete SoC but also offers a granular breakdown of power usage across the various RTL modules comprising our design. Figure 4.4 visually captures this data in the form of a segmented pie chart. This chart outlines the dynamic power distribution among the main components of the PULPino SoC when it is augmented with the RisCO2 processor and is running the demodulation test program.

According to Vivado's analysis, the total dynamic power consumption for the PULPino platform, under these conditions, amounts to 19 mW at a clock frequency of 25 MHz. The pie chart presents this total as a set of relative percentages, attributing portions of the overall consumption to each component within the SoC. Additionally, the total static power consumption is noted to be 94 mW.

As depicted in the pie chart, the core region is the largest consumer of power, accounting for 47.4% of the total. This region encompasses the components situated above the AXI interconnect as depicted in Figure 3.8.

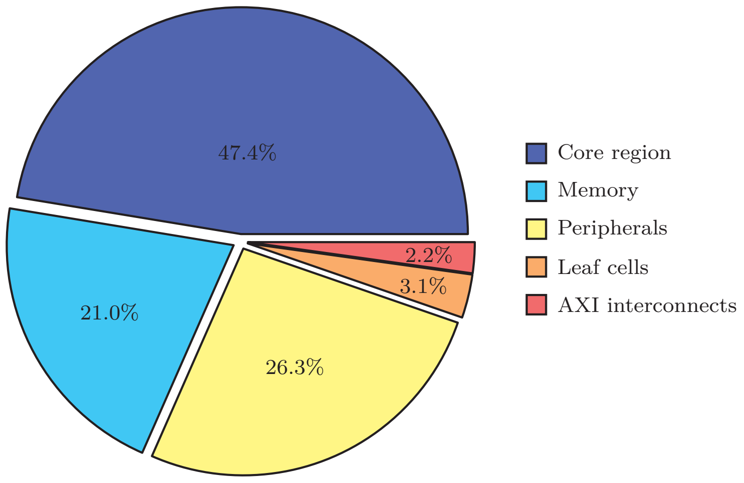


Figure 4.4: Dynamic power distribution among the main components of the PULPino platform when integrating a RisCO2 processor.

This includes the processor core and the multiplexors controlling the data flow for both the data and instruction memories. The instruction and data memory, which are represented separately in the pie chart, consume an additional 16% of the total power. Cumulatively, the processor and memory components alone are responsible for 68.4% of the system's total power consumption.

The peripherals on the PULPino platform, although not utilized by the test program, still consume 26.3% of the power. These peripherals enhance the system's functionality and enable connectivity, facilitating seamless communication with a range of external devices like sensors, actuators, and various communication interfaces.

AXI interconnects, serving as the communication backbone that links the processor, peripherals, and other key components, account for 2.2% of the total power consumption. While these interconnects play a critical role in ensuring efficient data transfer within the system, their power consumption is relatively low in this specific case due to the application's lack of communication with peripherals. Additionally, another 3.1% of the power consumption is attributed to leaf cells. These cells are crucial for interfacing the FPGA with external hardware and for maintaining optimal signal integrity.

The pie chart shown in Figure 4.5 provides a more granular breakdown of the dynamic power consumption during the execution of the demodulation test program, focusing specifically on the power consumption within the various RTL modules of the RisCO2 processor. The chart delineates how the 5 mW of dynamic power consumption, as indicated in Table 4.5, is distributed among these individual components. This level of power breakdown analysis provides us with invaluable insights into the inner workings and efficiency of the RisCO2 processor. It helps in pinpointing not only the most energy-consuming modules but also offers clues for potential optimization strategies.

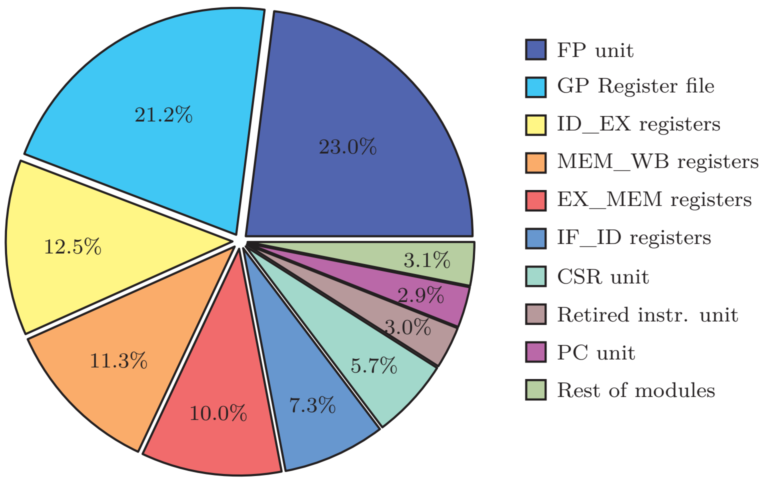


Figure 4.5: Dynamic power distribution among the various modules of the RisCO2 processor.

The dynamic power breakdown analysis for the RisCO2 processor offers significant insights into its power distribution. Most notably, the Floating-Point Unit and the General-Purpose (GP) Register file emerge as the most power-intensive components, accounting for 23.0% and 21.2% of total power consumption, respectively. The FPU's high consumption is understandable in the context of the demodulation test program. The algorithm relies heavily on floating-point calculations at each loop iteration, which underscores the necessity for a high-performance FPU in the system. Similarly, the GP Register file is a key repository for the processor's general-purpose registers, essential for temporarily storing data during program execution.

The FP unit and the GP Register file together consume nearly 44.2% of the power, suggesting that any efforts to optimize these specific components could have a substantial impact on overall energy efficiency. However, these are also some of the most functionally critical parts of the processor, so any optimizations would need to be carefully balanced to avoid sacrificing performance.

Pipeline stage registers, with ID_EX, MEM_WB, EX_MEM, and

IF_ID registers each account for 12.5%, 11.3%, 10.0%, and 7.3% of the power consumption, respectively. Cumulatively, they contribute to 41.1% of the overall power usage. These registers are instrumental in orchestrating the instruction execution pipeline, facilitating the flow of data and control signals between different stages. Their relatively high power consumption can be attributed to the need for fast and efficient data transfer within the pipeline. Interestingly, even though the pipeline stage registers collectively consume a significant chunk of the power, the power distribution among them is not uniform. Specifically, the ID_EX and EX_MEM registers consume twice as much power as the MEM_WB and IF_ID registers. This could imply a heavier data transfer load or more complex control logic at these particular stages, warranting a closer examination for possible optimizations.

The pipeline stage registers also contribute significantly, with the ID_EX, MEM_WB, EX_MEM, and IF_ID registers collectively accounting for 41.1% of the total power consumption, making the pipeline the most significant consumer as a group. This could be an area to explore more granular power-saving techniques, perhaps by optimizing the way instructions are queued or data is transferred. These registers are vital for ensuring seamless data and control signal flow between different pipeline stages, hence their substantial contribution to power usage.

Further down the list, the Control Status Register (CSR) unit, responsible for a variety of control and status functions, accounts for 5.7% of the power consumption. Following that, the Retired Instruction Unit, which generates a pulse every time an instruction is completed or retired (and serves to increment the `minstret` performance counter in the CSR), takes up 3.0% of the total power. Lastly, the Program Counter (PC) unit, which manages the sequence of instructions being executed, consumes 2.9% of the power. The CSR, PC, and Retired Instruction units combined take up 11.6% of the power. Although these units perform essential control and management tasks, their comparatively lower consumption might indicate already efficient design or fewer opportunities for optimization without functional compromises.

Finally, a catch-all category labeled as 'Rest of modules' constitutes the remaining 3.1%. This category includes various auxiliary circuits, control logic, and other components necessary for the full functionality of the processor that individually does not consume much power. Exploring this category further might uncover low-hanging fruit for easy power-saving optimizations.

All these values are observed under the execution of a specific test program (the demodulation test program). The power consumption characteristics might vary with different types of workload and understanding the dynamic power breakdown for different types of tasks helps identify areas of the processor design that are particularly power-hungry. This type of analysis provides the foundation for targeted optimizations, such as implementing power-saving techniques in the FPU when not in use, optimizing data flow in the pipeline registers, or exploring alternative register file architectures to further reduce power consumption.

4.4 Real Power Consumption Measurements on FPGA Chip

The objective of this section is to outline the methodology employed for measuring real-time power consumption on the Xilinx Alveo U200 FPGA board. The focus of this analysis is the implementation of the PULPino SoC, with particular attention to the reference cores included in our study. We have limited our measurements to the three most energy-efficient cores, as listed in Table 4.5: Ri5cy, CV32E40P, and RisCO2.

The Alveo U200 [102] is a high-end FPGA board designed by Xilinx, primarily for accelerating data center tasks such as machine learning, video processing, and other data-intensive activities. We chose to transition from the Nexys4 board to the Alveo U200 due to the advanced features it offers, which are particularly useful for real-time power consumption measurements. These features include a built-in satellite controller that can communicate directly with a power regulator. This eliminates the need for external measuring instruments, thereby streamlining the process of assessing power usage while the PULPino SoC runs our benchmark application on one of the selected cores. Additionally, the Alveo U200 board is fully compatible with Xilinx's Vitis unified software platform, simplifying design deployment. It also supports kernels developed in OpenCL, offering a more straightforward transition from software to hardware implementations. The Vitis unified software platform [103] is more geared towards data center acceleration tasks and typically targets Xilinx's more advanced hardware platforms, such as the Alveo series of data center accelerator cards and the Zynq UltraScale+ series of SoCs.

The methodology that we followed to get the power measurements is outlined in the following steps:

1. RTL design of a self-contained PULPino SoC system: The first step involves designing a self-contained PULPino system for each processor we

intend to evaluate. This design integrates a specific version of the core under examination, ROM memory containing the test program, RAM memory for general storage during the test, and I/O pins for external communication.

2. Integration into Vitis RTL kernel: The next phase involves the encapsulation of the RTL design into a Vitis RTL Kernel. This kernel allows for the execution of our tests on the Alveo U200 board, offering direct access to the hardware features that facilitate precise power measurement functionalities.

3. Output configuration: To complete the design process, one of the I/O pins is configured as an output used by the kernel. This is accomplished by performing a logical OR with the 'ap_done' signal. This serves as a flag to indicate to the host system that the kernel has completed its execution.

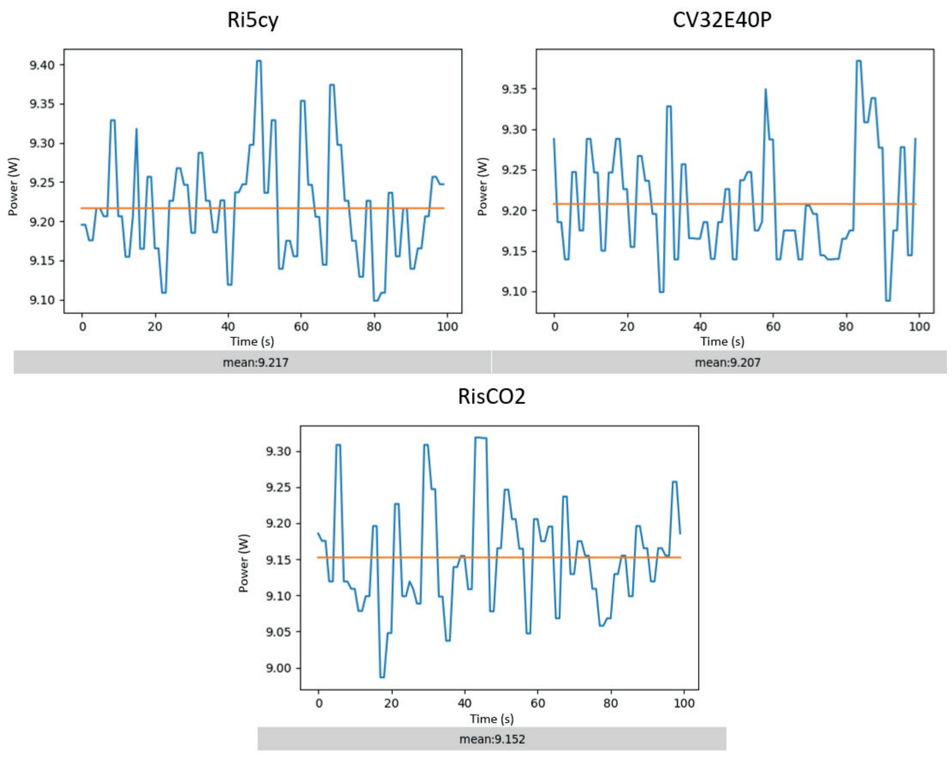


Figure 4.6: Obtained power readings, along with the averaged power consumption for each tested processor core.

4. Host application development: We developed a custom host application, running on an x86 architecture, to boot the FPGA kernel. This

step is crucial because RTL kernels in Vitis include a "satellite controller," an embedded system that communicates with an onboard power regulator. This regulator has a built-in power meter whose real-time data can be extracted using the command `xbutil -report electrical`.

5. Python program for data collection: Finally, to automate the data collection process, we implemented a Python program that invokes the `xbutil` command at regular intervals. This script calculates the average power consumption based on these collected readings, in order to compensate for the variability in instantaneous power values. We have modified the test program so that the algorithm runs in an infinite loop, but we limit the test to run for a duration of 100 seconds. During this interval, periodic power measurements are taken for the FPGA chip, not the entire board. These measurements are then averaged to smooth out the variability in the readings. The obtained power readings, along with the averaged power consumption for each tested processor core, are presented in Figure 4.6.

By following this methodology, we ensure that we obtain accurate, consistent, and reliable data on power consumption for different processor cores when implemented on the FPGA platform. This data then serves as a basis for comparative analysis aimed at optimizing power efficiency, which is a core objective of this thesis.

Core	**FPGA Power [W]**	**Estimated Core Power [mW]**
Ri5cy2	9.217	70
CV32E40P	9.207	60
RisCO2	9.152	5

Table 4.6: Measurement of the total power consumption of the Alveo U200 FPGA chip, including the PULPino SoC, for different processors and using a clock setting of 25 MHz.

Table 4.6 presents the total power consumption values for the Alveo U200 FPGA as measured by the Vitis software platform. Each entry in the table corresponds to a distinct project setup, in which the FPGA incorporates both the PULPino SoC and the targeted processor under evaluation. All the projects operate at a clock frequency of 25 MHz. The data from Table 4.6 is visually represented in Figure 4.7.

The real power measurements from the FPGA can be conceptualized as comprising two components: one component that remains relatively constant irrespective of the processor design, and another that varies depending on

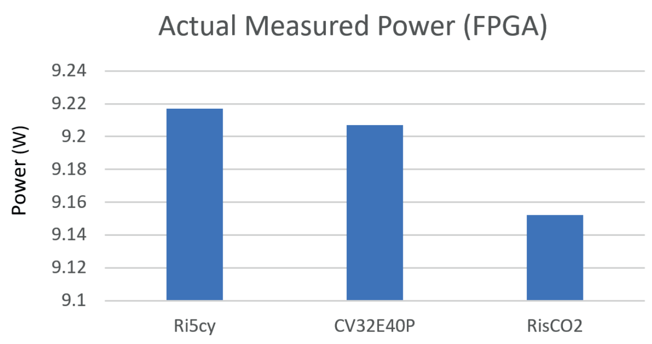


Figure 4.7: Total power measurement of the FPGA chip including PULPino SoC featuring different processors.

the specific processor in use. For instance, in the case of the RisCO2 processor with a dynamic power consumption of 5 mW, we can estimate the constant part of the power to be $9.152W - 0.005W = 9.147W$. Subtracting this constant value from the measured power allows us to isolate the power consumption attributable to the processor, as demonstrated in the third column of Table 4.6.

Upon comparing these adjusted power values with the simulation data in Table 4.5, we find a reasonable agreement between the two datasets. This concurrence is visually represented in Figure 4.8.

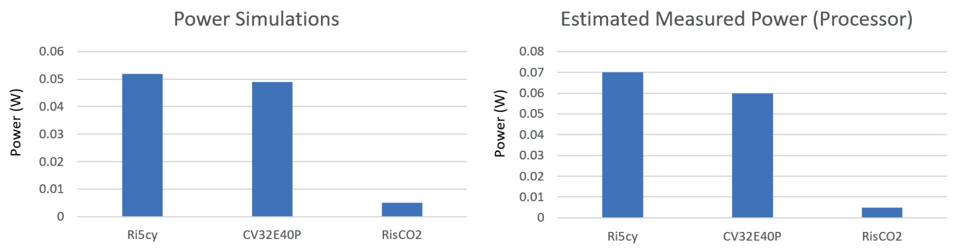


Figure 4.8: Comparison of simulated processor power consumption from Vivado with estimated power measurements inferred from Vitis.

The alignment between the real and simulated power values serves to validate the simulation results, which in turn have been instrumental in guiding the design optimization of the RisCO2 processor.

4.5 ASIC Synthesis Results

In a final push to gather comprehensive data, we succeeded in acquiring synthesis results for both the RisCO2 and Zero-riscy processors when integrated into the PULPino SoC. The synthesis process was carried out using Cadence's Genus synthesis solution [104], targeting TSMC's 65nm technology.

Module	**Cell Count**	**Cell Area [μm^2]**	**Net Area [μm^2]**	**Total Area [μm^2]**
pulpino_top	28379	332879.5	56495.7	389375.2
core_region	19591	285399.1	39190.2	324589.3
RisCO2	14407	66500.4	29825.6	96326.0
FP_unit	7358	31542.0	14969.7	46511.7
reg_file	525	5832.0	211.8	6043.8
rf_mux	1068	2882.4	1796.9	4679.3
ID_EX	242	2569.2	137.1	2706.3
EX_MEM	174	1839.6	100.5	1940.1
MEM_WB	158	1701.6	86.8	1788.4
FPU_ctrl	181	563.6	153.4	717.0
hazard_unit	152	457.2	238.5	695.7
decoder	100	235.2	139.4	374.6
forward_unit	68	208.0	79.2	287.2
rf_decoder	26	60.0	24.7	84.7
instr_mem	7	94545.0	21.0	94566.0
data_mem	7	94545.0	21.0	94566.0
instr_mem_axi	1210	8129.6	2024.2	10153.8
data_mem_axi	1189	7923.2	1995.0	9918.2
peripherals	6885	40163.2	12809.6	52972.8
axi_intercnct	1898	7273.2	3135.5	10408.7
clk_rst_gen	5	44.0	4.3	48.3

Table 4.7: Synthesis area results of RisCO2 when integrated into the PULPino SoC, using TSMC 65nm technology.

The primary objective of this exercise is to offer a side-by-side comparison of the two cores under conditions that simulate their implementation in an actual silicon chip. By doing so, we aim to provide a clearer understanding of how each processor performs in terms of key metrics such as resource utilization when deployed in real-world applications.

Module	**Cell Count**	**Cell Area [μm^2]**	**Net Area [μm^2]**	**Total Area [μm^2]**
pulpino_top	20253	297100.7	39562.7	336663.4
core_region	11277	248797.9	21703.4	270501.3
Zero-riscy	6048	29662.4	12084.1	41746.5
reg_file	2755	16302.4	5337.9	21640.3
if_stage	1028	4819.2	1881.8	6701.0
cs_regs	467	1988.4	790.8	2779.2
ex_alu	538	1774.0	901.8	2675.8
load_store	435	1821.6	703.3	2524.9
debug_unit	232	1230.4	293.5	1523.9
hazard_unit	152	457.2	238.5	695.7
id_decoder	159	366.4	221.4	587.8
id_controller	68	208.0	79.2	287.2
id_int_ctrl	13	90.8	12.2	103.0
instr_mem	7	94545.0	21.0	94566.0
data_mem	7	94545.0	21.0	94566.0
instr_mem_axi	1223	8161.6	2050.6	10212.2
data_mem_axi	1189	7919.6	1995.0	9914.6
peripherals	7035	40912.8	13041.4	53954.2
axi_intercnct	1936	7346.0	3196.0	10542.0
clk_rst_gen	5	44.0	4.3	48.3

Table 4.8: Synthesis area results of Zero-riscy when integrated into the PULPino SoC, using TSMC 65nm technology.

4.5.1 Synthesis Area Results

Tables 4.7 and 4.8 summarize the synthesis results obtained for each core. The overall area for the PULPino with RisCO2 is about 15.7% larger than the PULPino with Zero-riscy. Within these configurations, the core region of RisCO2 exceeds that of Zero-riscy by about 20%. Remarkably, the RisCO2 processor core itself is more than twice as large as the core of Zero-riscy. This substantial size is due in part to RisCO2's Floating-Point Unit, which alone occupies 46,511.7 μm^2. This accounts for nearly 48.3% of RisCO2's total core area, corroborating the resource utilization observed in FPGA implementations. However, this large allocation for the FPU does limit the area available for other components, such as the register file.

In contrast, Zero-riscy lacks an FPU, which is the primary reason for its smaller overall area. However, Zero-riscy compensates by dedicating a significantly larger area to its register file—21,640.3 μm^2, which is approximately 3.6 times larger than RisCO2's register file of 6,043.8 μm^2. This confirms a different architectural approach to register management and may lead to reduced memory access times. The larger register file could offer performance gains, although potentially at the cost of increased power consumption.

To summarize, RisCO2's larger footprint is primarily due to its added complexity, including an FPU and a deeper pipeline. Zero-riscy, despite being smaller overall, allocates more area to its register file, highlighting how the two are optimized for different types of workloads and thus exhibit different performance characteristics as a result.

4.5.2 Power Estimation Results

The power estimations presented below were generated by the synthesis tool, using a clock setting of 100 MHz:

Core	**Leakage [μW]**	**Internal [mW]**	**Switching [mW]**	**Total [mW]**
RisCO2	14.90	15.00	4.64	19.66

Table 4.9: Power estimations using a clock setting of 100 MHz.

The term 'leakage' power is essentially synonymous with 'static' power, as previously detailed in Chapter 2. Both terms describe the power consumed by a digital circuit when it is not actively switching but remains powered on. The terms 'internal' power and 'switching' power refer to two distinct components that make up the total dynamic power consumption of a digital circuit.

'Internal' power refers to the power consumed due to internal node capacitances in the logic gates as they charge and discharge, typically during transitions between logic states. This is also sometimes termed 'short-circuit power,' as it accounts for the power consumed when the PMOS and NMOS transistors in a CMOS gate are both momentarily conducting, creating a brief short-circuit current path from VDD to ground.

'Switching' power refers to the power consumed while driving the outputs, which entails charging and discharging the load capacitances—often input capacitances of subsequent gates or interconnect capacitances. Switching power is directly proportional to both the switching frequency and the square of the supply voltage.

4.6 ASIC Layout

Figure 4.9 provides a schematic or conceptual representation of the ASIC generated as a result of the synthesis process using TSMC's 65 nm technology. This illustrative layout is aimed at providing the spatial distribution and area allocation of each of the main constituents within the PULPino SoC; this specific example features the RisCO2 processor core.

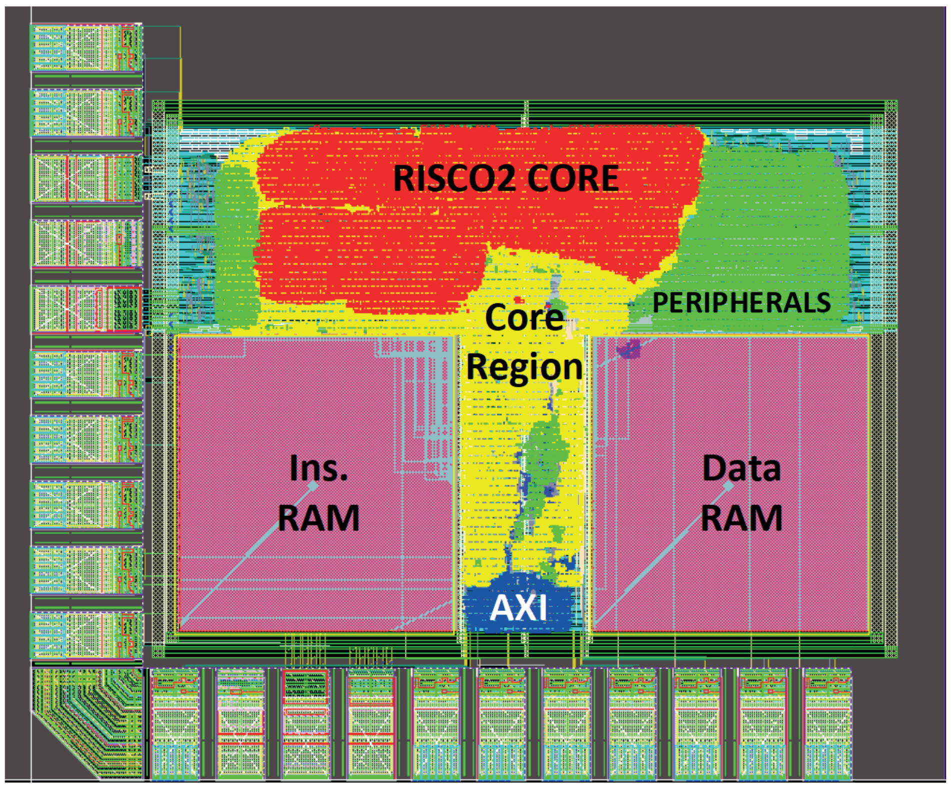


Figure 4.9: Conceptual layout of the PULPino SoC featuring a RisCO2 processor (TSMC's 65 nm).

By describing the layout as 'conceptual,' it should be clarified that not every potential pin associated with the PULPino SoC is depicted, but only a subset is included. This is done to simplify the visual narrative and make it more accessible for viewers to grasp the relative proportions of area utilization.

Understanding area utilization is critical not only for evaluating the efficiency of the design but also for assessing how well the system components integrate with each other. This area overview thus serves as an informative visual guide for stakeholders, offering insights into the balance between computational capability and physical dimensions.

The key elements detailed in this layout are the core region, which is the nexus of computational activity and includes the RisCO2 processor core itself; the memory blocks, which provide the necessary storage facilities; the peripherals, which add specific capabilities and interfaces; and the AXI interconnect, which serves as the communication backbone among these various elements.

Chapter 5

Conclusions and Future Directions

In the landscape of increasing demand for low-power devices across fields such as wearables, smart sensors, and the Internet of Things, energy-efficient processor design has emerged as a pivotal research focus. The experimental results presented in this work demonstrate the effectiveness of custom-designed RISC-V processors for low-power applications, particularly in the context of energy-efficient signal processing in NDIR gas sensors. The study also provides insights for optimizing processors in energy-constrained embedded systems.

In the initial stage, we developed a self-contained system that featured a basic RV32I processor with a 5-stage pipeline. The processor was directly connected to program and instruction memory, minimizing resource utilization. This basic setup was capable of performing only simple integer addition and subtraction operations, necessitating software emulation for the required floating-point operations. In this initial state, operating at 50 MHz, the processor consumed 104 mJ of energy and took 2.88 seconds to compute the CO_2 concentration level in our benchmark application. By implementing targeted improvements, such as the incorporation of the standard 'M' and 'F' ISA extensions and the corresponding functional units, we were able to provide hardware support for more computationally demanding floating-point instructions. As a result, the energy consumption was dramatically reduced to just 0.45 mJ, and the computation time was slashed to 12 ms. Although these optimizations doubled the FPGA resource utilization, the trade-off is deemed acceptable in the context of energy-constrained applications.

In the second stage of our design exploration study, we focused on further optimizing our best-performing RV32IMF processor. Our goal was to minimize the increase in area utilization necessary for the energy-saving optimizations while maintaining the achieved energy performance levels. Remarkably, the strategies we adopted to reduce resource utilization not only achieved that goal but also led to an additional reduction in energy consumption. This near-proportional correlation between reduced resource utilization and decreased energy use is illustrated in Figure 4.2.

Our main approaches for reducing resource utilization centered on two strategies. First, we substantially decreased the size of the register file by a factor of 1/4 through the adoption of more specialized ISA extensions, namely 'E' and 'Zfinx'. Second, we removed any logic or hardware support for instructions or features not required by our specific application. The linear relationship observed between the reduction of resources and energy consumption underscores the superfluous nature of the hardware that was eliminated. It also indicates that more streamlined architectures can lead to more energy-efficient results.

The resultant optimized processor design, which we have named RisCO2, demonstrates substantial energy savings. Compared to the RV32IMF design from the first stage of our design exploration study, RisCO2 shows an 82% reduction in energy consumption. Even more remarkably, when RisCO2 is benchmarked against general-purpose reference processors like Ri5cy and CV32E40P, it exhibits a 90% and 87% reduction in energy consumption, respectively. These savings are achieved without compromising the application's runtime performance.

In summary, our findings demonstrate the viability and advantages of using specialized RISC-V architectures like RisCO2 for energy-critical applications across a diverse range of low-power devices—be it in the Internet of Things, wearables, or mobile technology. The gains in both energy and performance that RisCO2 offers make it a compelling solution for a variety of energy-constrained environments, from embedded systems with limited hardware resources to battery-powered devices where energy efficiency is crucial.

This study not only contributes to the ongoing efforts in energy-efficient computing but also aligns with the broader agenda for sustainable technology. Through the development of RisCO2, we have addressed the need to enhance energy efficiency in computing platforms, confirming the RISC-V architecture as a promising avenue for meeting these challenges.

Future Directions

Looking ahead, there are several avenues for extending this work, which can be categorized into two broad areas. The first focuses on specific improvements at the processor level, particularly applicable to RisCO2. The second goes beyond RisCO2 and involves adapting the processor architecture to suit specific software applications, aiming for additional energy and area savings.

Despite RisCO2 featuring a deeper pipeline with an additional stage and utilizing the same FPU as CV32E40P, our findings show that its IPC performance falls slightly short. Specifically, RisCO2's IPC is 18% lower than that of CV32E40P. This discrepancy warrants further study to better optimize performance. Additionally, the impact of extending the pipeline depth on energy efficiency deserves closer examination. While deeper pipelines may seem beneficial, they can introduce issues such as pipeline hazards and stalls, potentially reducing the processor's overall performance.

Another potential avenue for processor optimization lies in the integration of custom instructions that support hardware loops. Such features are already found in processors like CV32E40P and Ri5cy. Implementing hardware loops could significantly expedite the demodulation algorithm, whose main processing loop goes through as many iterations as there are samples produced by the ADC. However, this improvement could add to the processor's hardware complexity and potentially offset any gains in energy efficiency. The implementation would also require compiler modifications to generate code that employs the new instruction opcodes.

A further area to explore could involve the addition of a second Floating Point Unit that operates in parallel with the existing one. This additional unit would have reduced functionality, capable only of performing addition and multiplication operations. This setup would effectively double the number of processed samples per loop iteration, achieving this increase through a loop unrolling technique enabled by the additional FPU. An important aspect to examine would be how the increased power consumption from this additional FPU interacts with the potential halving of application runtime to affect overall energy consumption.

To transition RisCO2 into the sensor industry, several key steps must be taken to ensure it integrates effectively with the PULPino SoC and performs reliably in real-world conditions. Specifically, the following actions and tests are required:

1) Sensor Integration: First, we need to connect a physical NDIR CO_2 sensor to the PULPino SoC through the SPI bus via an ADC. The software should be modified to read from the ADC and either store the samples or process them in real-time for CO_2 level calculations. Moreover, we should explore the adaptability of the platform for measuring concentrations of gases other than CO_2 using NDIR sensors and proper methods for sensor calibration.

2) Wireless Communication: The next step involves adding a wireless communication module to the PULPino platform to enable the transmission of real CO_2 measurements to the cloud. Regarding radio frequency transceivers, new ultra-low power solutions are available that can further justify the energy optimization on the processing side. For example, a new low-power RF transceiver based on the HaLow protocol (IEEE 802.11ah) has a power consumption of just 0.117 mW at 3.6 V [105]—more than two orders of magnitude lower than the processor's power consumption.

3) Real-World Power Measurements: Finally, it is essential to carry out new power measurements under these more realistic conditions, collecting samples from an actual sensor and sending data to the cloud.

By undertaking these actions, we can more thoroughly validate RisCO2 and assess its suitability as a reliable platform for applications such as wireless gas sensors.

Expanding our view beyond RisCO2, one of the most interesting research directions lies in developing a systematic methodology for designing highly parameterizable, generic processors. These processors could be swiftly customized to align with the specific energy requirements of the applications they are intended for. Such an approach would take into account the individual computational demands and energy constraints, allowing for more fine-tuned, application-specific energy optimization.

The urgency of this research is accentuated by the anticipated proliferation of embedded applications, particularly in the rapidly expanding landscape of IoT devices. According to various forecasts, we are looking at an expected deployment of 30 billion IoT devices by 2030 [106]–[108], and some even more optimistic estimates by ARM Ltd. suggest this could balloon to as many as 1 trillion devices by the same year [109].

We believe that the inherent modularity and flexibility of the RISC-V ISA could play a pivotal role in reducing the energy consumption of the rapidly growing number of connected devices. RISC-V not only allows for the integration of custom extensions and specialized instructions to enhance processor energy efficiency but also enables the removal of unused logic. This eliminates unnecessary energy consumption, particularly in application-specific designs.

In this manner, RISC-V can serve as a foundational technology to substantially mitigate the energy demands of the next generation of connected devices, making it pivotal for sustainable technology development.

References

[1] N. Garg and R. Garg, “Energy harvesting in IoT devices: A survey,” in *2017 International Conference on Intelligent Sustainable Systems (ICISS)*, 2017, pp. 127–131. DOI: 10.1109/ISS1.2017.8389371.

[2] K. Asanović and D. A. Patterson, “Instruction sets should be free: The case for RISC-V,” *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2014-146*, 2014. [Online]. Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/2014/EECS-2014-146.html.

[3] X. Lu, I. H. Kim, A. Xhafa, J. Zhou, and K. Tsai, “Reaching 10-years of battery life for industrial IoT wireless sensor networks,” in *2017 Symposium on VLSI Circuits*, 2017, pp. C66–C67. DOI: 10.23919/VLSIC.2017.8008550.

[4] EnABLES - European Research Infrastructure Powering the Internet of Things, *EnABLES Research Infrastructure Position Paper*, https://www.enables-project.eu/wp-content/uploads/2021/02/EnABLES_ResearchInfrastructure_PositionPaper.pdf, Accessed: 2023-07-07, 2019.

[5] A. Nordrum, “Popular Internet of Things Forecast of 50 Billion Devices by 2020 Is Outdated,” *IEEE Spectrum*, Jul. 2016. [Online]. Available: https://spectrum.ieee.org/popular-internet-of-things-forecast-of-50-billion-devices-by-2020-is-outdated.

[6] Peng, Z. *et al.*, “Practical Indicators for Risk of Airborne Transmission in Shared Indoor Environments and their application to COVID-19 Outbreaks,” *medRxiv*, 2021. DOI: 10.1101/2021.04.21.21255898. [Online]. Available: https://www.medrxiv.org/content/early/2021/05/07/2021.04.21.21255898.

[7] K. Asanovic, R. Avizienis, J. Bachrach, *et al.*, “The Rocket Chip Generator,” *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2016-17*, vol. 4, 2016.

[8] Xilinx, Inc., *MicroBlaze Soft Processor Core — xilinx.com*, https://www.xilinx.com/products/design-tools/microblaze.html, [Accessed 09-Jul-2023], 2023.

[9] Intel Corp., *Nios® Processor for FPGAs - Intel® FPGA — intel.com*, https://www.intel.com/content/www/us/en/products/details/fpga/nios-processor.html, [Accessed 09-Jul-2023].

[10] D. Large and J. Farmer, “Chapter 11 - Emerging Architectures,” in ser. The Morgan Kaufmann Series in Networking, 2009, pp. 321–346. DOI: https://doi.org/10.1016/B978-0-12-374401-2.00011-5.

[11] *Jupyter: Free software, open standards, and web services for interactive computing across all programming languages*, https://jupyter.org/, Accessed: 2022-07-25.

[12] A. Traber, F. Zaruba, S. Stucki, *et al.*, “PULPino: A small single-core RISC-V SoC,” in *3rd RISCV Workshop*, 2016.

[13] PULP Team, *PULP Platform. Open hardware, the way it should be!* https://pulp-platform.org/team.html, Accessed: 2022-07-25.

[14] RISC-V International, *RISC-V Specifications*, https://riscv.org/technical/specifications, Accessed: 2023-09-06.

[15] C. Papon, *VexRiscv: A FPGA friendly 32-bit RISC-V CPU implementation*, http://github.com/SpinalHDL/VexRiscv, Accessed: 2023-07-28.

[16] Western Digital, *EH1 SweRV RISC-V Core*, http://github.com/westerndigitalcorporation/swerv_eh1, Accessed: 2023-07-28.

[17] M. Gautschi, P. D. Schiavone, A. Traber, *et al.*, “Near-threshold RISC-V core with DSP extensions for scalable IoT endpoint devices,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 10, pp. 2700–2713, 2017. DOI: 10.1109/TVLSI.2017.2654506.

[18] OpenHW Group, *CV32E40P User Manual*, https://docs.openhwgroup.org/projects/cv32e40p-user-manual/en/latest/index.html, Accessed: 2023-07-28.

[19] P. D. Schiavone, F. Conti, D. Rossi, *et al.*, “Slow and steady wins the race? A comparison of ultra-low-power RISC-V cores for Internet-of-Things applications,” in *2017 27th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, IEEE, 2017, pp. 1–8. DOI: 10.1109/PATMOS.2017.8106976.

[20] SiFive, Inc., *SiFive E3 Coreplex Series Manual*, https://static.dev.sifive.com/SiFive-E3-Coreplex-v1.1.pdf, Accessed: 2023-09-07.

[21] S. Jiang and F. lin, "The best SoC solution with AndesCore and Andes's platform," in *Proceedings of Technical Program of 2012 VLSI Design, Automation and Test*, 2012, pp. 1–4. DOI: 10.1109/VLSI-DAT.2012.6212638.

[22] Codasip, *Codasip RISC-V processors*, https://codasip.com/products/codasip-risc-v-processors/, Accessed: 2023-09-07.

[23] J. M. Rabaey, A. Chandrakasan, and B. Nikolic, *Digital integrated circuits- A design perspective*, 2ed. Prentice Hall, 2004.

[24] H. J. M. Veendrick, "Short-circuit dissipation of static CMOS circuitry and its impact on the design of buffer circuits," *IEEE Journal of Solid-state Circuits*, vol. 19, pp. 468–473, 1984.

[25] D. Castells-Rufas, A. Saà-Garriga, and J. Carrabina, "Energy Efficiency of Many-Soft-Core Processors," *CoRR*, vol. abs/1601.07133, 2016. arXiv: 1601.07133. [Online]. Available: http://arxiv.org/abs/1601.07133.

[26] T. Scogland, B. Subramaniam, and W.-C. Feng, "The Green500 List: Escapades to Exascale," *Comput. Sci.*, vol. 28, no. 2–3, pp. 109–117, May 2013, ISSN: 1865-2034. DOI: 10.1007/s00450-012-0212-6. [Online]. Available: https://doi.org/10.1007/s00450-012-0212-6.

[27] T. Burd and R. Brodersen, "Energy efficient CMOS microprocessor design," in *Proceedings of the Twenty-Eighth Annual Hawaii International Conference on System Sciences*, vol. 1, 1995, 288–297 vol.1. DOI: 10.1109/HICSS.1995.375385.

[28] S. Arya, H. Sachs, and S. Duvvuru, "An Architecture for High Instruction Level Parallelism," in *Proceedings of the Twenty-Eighth Annual Hawaii International Conference on System Sciences*, vol. 1, 1995, 153–162 vol.1. DOI: 10.1109/HICSS.1995.375398.

[29] S. Misra, A. A. Alfa, M. O. Olaniyi, and S. O. Adewale, "Exploratory Study of Techniques for Exploiting Instruction-Level Parallelism," in *2014 Global Summit on Computer & Information Technology (GSCIT)*, 2014, pp. 1–6. DOI: 10.1109/GSCIT.2014.6970103.

[30] R. Puri, L. Stok, J. Cohn, *et al.*, "Pushing ASIC Performance in a Power Envelope," in *Proceedings of the 40th Annual Design Automation Conference*, ser. DAC '03, Anaheim, CA, USA: Association for Computing Machinery, 2003, pp. 788–793, ISBN: 1581136889. DOI: 10.1145/775832.776032. [Online]. Available: https://doi.org/10.1145/775832.776032.

[31] B. Peccerillo, M. Mannino, A. Mondelli, and S. Bartolini, “A survey on hardware accelerators: Taxonomy, trends, challenges, and perspectives,” *Journal of Systems Architecture*, vol. 129, p. 102 561, 2022, ISSN: 1383-7621. DOI: https://doi.org/10.1016/j.sysarc.2022.102561. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1383762122001138.

[32] M. K. Jain, M. Balakrishnan, and A. Kumar, “ASIP design methodologies: survey and issues,” in *VLSI Design 2001. Fourteenth International Conference on VLSI Design*, 2001, pp. 76–81. DOI: 10.1109/ICVD.2001.902643.

[33] S. Triantafyllis, M. Vachharajani, N. Vachharajani, and D. August, “Compiler optimization-space exploration,” in *International Symposium on Code Generation and Optimization, 2003. CGO 2003.*, 2003, pp. 204–215. DOI: 10.1109/CGO.2003.1191546.

[34] A. Hartstein and T. Puzak, “The optimum pipeline depth for a microprocessor,” in *Proceedings 29th Annual International Symposium on Computer Architecture*, 2002, pp. 7–13. DOI: 10.1109/ISCA.2002.1003557.

[35] J. E. Smith, “A Study of Branch Prediction Strategies,” in *Proceedings of the 8th Annual Symposium on Computer Architecture*, ser. ISCA ’81, Minneapolis, Minnesota, USA: IEEE Computer Society Press, 1981, pp. 135–148.

[36] D. A. Jiménez and C. Lin, “Neural Methods for Dynamic Branch Prediction,” *ACM Trans. Comput. Syst.*, vol. 20, no. 4, pp. 369–397, Nov. 2002, ISSN: 0734-2071. DOI: 10.1145/571637.571639. [Online]. Available: https://doi.org/10.1145/571637.571639.

[37] P. Arroba, J. M. Moya, J. L. Ayala, and R. Buyya, “Dynamic Voltage and Frequency Scaling-aware dynamic consolidation of virtual machines for energy efficient cloud data centers,” *Concurrency and Computation: Practice and Experience*, vol. 29, 2017.

[38] Z. Hu, A. Buyuktosunoglu, V. Srinivasan, V. V. Zyuban, H. M. Jacobson, and P. Bose, “Microarchitectural techniques for power gating of execution units,” *Proceedings of the 2004 International Symposium on Low Power Electronics and Design (IEEE Cat. No.04TH8758)*, pp. 32–37, 2004.

[39] J. R. Shinde and S. S. Salankar, “Clock gating — A power optimizing technique for VLSI circuits,” *2011 Annual IEEE India Conference*, pp. 1–4, 2011.

[40] Ü. Y. Ogras, R. Marculescu, P. Choudhary, and D. Marculescu, "Voltage-Frequency Island Partitioning for GALS-based Networks-on-Chip," *2007 44th ACM/IEEE Design Automation Conference*, pp. 110–115, 2007.

[41] N. G. Tsoutsos and M. Maniatakos, "Investigating the Application of One Instruction Set Computing for Encrypted Data Computation," in *Security, Privacy, and Applied Cryptography Engineering*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 21–37, ISBN: 978-3-642-41224-0.

[42] K. Inoue, T. Ishihara, and K. Murakami, "Way-predicting set-associative cache for high performance and low energy consumption," *Proceedings. 1999 International Symposium on Low Power Electronics and Design (Cat. No.99TH8477)*, pp. 273–275, 1999.

[43] R. M. Banakar, S. Steinke, B.-S. Lee, M. Balakrishnan, and P. Marwedel, "Scratchpad memory: A design alternative for cache on-chip memory in embedded systems," *Proceedings of the Tenth International Symposium on Hardware/Software Codesign. CODES 2002 (IEEE Cat. No.02TH8627)*, pp. 73–78, 2002.

[44] X. Liu, S. Cheng, H. Liu, S. Hu, D. Zhang, and H. Ning, "A Survey on Gas Sensing Technology," *Sensors (Basel, Switzerland)*, vol. 12, pp. 9635–9665, 2012.

[45] A. Dey, "Semiconductor metal oxide gas sensors: A review," *Materials Science and Engineering B-advanced Functional Solid-state Materials*, vol. 229, pp. 206–217, 2018.

[46] N. Joshi, T. Hayasaka, Y. Liu, H. Liu, O. N. Oliveira, and L. Lin, "A review on chemiresistive room temperature gas sensors based on metal oxide nanostructures, graphene and 2D transition metal dichalcogenides," *Microchimica Acta*, vol. 185, pp. 1–16, 2018.

[47] R. Alrammouz, J. Podlecki, P. Abboud, B. Sorli, and R. Habchi, "A review on flexible gas sensors: From materials to devices," *Sensors and Actuators A: Physical*, vol. 284, pp. 209–231, 2018, ISSN: 0924-4247. DOI: https://doi.org/10.1016/j.sna.2018.10.036. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0924424718308100.

[48] Á. Molina, V. A. Escobar-Barrios, and J. Oliva, "A review on hybrid and flexible CO2 gas sensors," *Synthetic Metals*, vol. 270, p. 116 602, 2020.

[49] R. Bogue, "Detecting gases with light: a review of optical gas sensor technologies," *Sensor Review*, vol. 35, pp. 133–140, 2015.

[50] T. Yamate, G. Fujisawa, and T. Ikegami, “Optical Sensors for the Exploration of Oil and Gas,” *Journal of Lightwave Technology*, vol. 35, pp. 3538–3545, 2017.

[51] D. Popa and F. Udrea, “Towards Integrated Mid-Infrared Gas Sensors,” *Sensors*, vol. 19, no. 9, 2019, ISSN: 1424-8220. DOI: 10.3390/s19092076. [Online]. Available: https://www.mdpi.com/1424-8220/19/9/2076.

[52] D. F. Swinehart, “The Beer-Lambert Law,” *Journal of Chemical Education*, vol. 39, p. 333, 1962.

[53] L. Kocsis, P. Herman, and A. Eke, “The modified Beer–Lambert law revisited,” *Physics in Medicine & Biology*, vol. 51, N91–N98, 2006.

[54] K. Fuwa and B. L. Vallé, “The Physical Basis of Analytical Atomic Absorption Spectrometry. The Pertinence of the Beer-Lambert Law.,” *Analytical Chemistry*, vol. 35, pp. 942–946, 1963.

[55] T. G. Mayerhöfer, S. Pahlow, and J. Popp, “The Bouguer-Beer-Lambert Law: Shining Light on the Obscure,” *Chemphyschem*, vol. 21, pp. 2029–2046, 2020.

[56] G. Casasanta, F. Falcini, and R. Garra, “Beer–Lambert law in photochemistry: A new approach,” *Journal of Photochemistry and Photobiology A: Chemistry*, vol. 432, p. 114 086, 2022, ISSN: 1010-6030. DOI: https://doi.org/10.1016/j.jphotochem.2022.114086. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1010603022003100.

[57] J. Hodgkinson and R. P. Tatam, “Optical Gas Sensing: A Review,” *Measurement Science and Technology*, vol. 24, no. 1, p. 012 004, Nov. 2012. DOI: 10.1088/0957-0233/24/1/012004. [Online]. Available: https://dx.doi.org/10.1088/0957-0233/24/1/012004.

[58] R. K. Jha, “Non-Dispersive Infrared Gas Sensing Technology: A Review,” *IEEE Sensors Journal*, vol. 22, pp. 6–15, 2022.

[59] A. Rosencwaig, “Photoacoustic spectroscopy.,” *Annual review of biophysics and bioengineering*, vol. 9, pp. 31–54, 1980.

[60] C. Haisch, “Photoacoustic spectroscopy for analytical measurements,” *Measurement Science and Technology*, vol. 23, p. 012 001, 2011.

[61] A. Fathy, Y. Sabry, I. Hunter, D. Khalil, and T. Bourouina, “Direct Absorption and Photoacoustic Spectroscopy for Gas Sensing and Analysis: A Critical Review,” *Laser & Photonics Reviews*, vol. 16, May 2022. DOI: 10.1002/lpor.202100556.

[62] P. Werle, R. Mücke, and F. Slemr, “The limits of signal averaging in atmospheric trace-gas monitoring by tunable diode-laser absorption spectroscopy (TDLAS),” *Applied Physics B*, vol. 57, pp. 131–139, 1993.

[63] M. Lackner, “Tunable Diode Laser Absorption Spectroscopy (TDLAS) in the Process Industries – A Review,” *Reviews in Chemical Engineering*, vol. 23, pp. 147–65, 2007.

[64] J. Li, B. Yu, W. Zhao, and W. Chen, “A Review of Signal Enhancement and Noise Reduction Techniques for Tunable Diode Laser Absorption Spectroscopy,” *Applied Spectroscopy Reviews*, vol. 49, pp. 666–691, 2014.

[65] S. Khan, D. Newport, and S. Le Calvé, “Gas Detection Using Portable Deep-UV Absorption Spectrophotometry: A Review,” *Sensors*, vol. 19, no. 23, 2019, ISSN: 1424-8220. DOI: 10.3390/s19235210. [Online]. Available: https://www.mdpi.com/1424-8220/19/23/5210.

[66] S. Khan, D. Newport, and S. Le Calvé, “Development of a Toluene Detector Based on Deep UV Absorption Spectrophotometry Using Glass and Aluminum Capillary Tube Gas Cells with a LED Source,” *Micromachines*, vol. 10, no. 3, 2019, ISSN: 2072-666X. DOI: 10.3390/mi10030193. [Online]. Available: https://www.mdpi.com/2072-666X/10/3/193.

[67] M. G. Ruppert, D. M. Harcombe, M. R. P. Ragazzon, S. O. R. Moheimani, and A. J. Fleming, “A review of demodulation techniques for amplitude-modulation atomic force microscopy,” *Beilstein Journal of Nanotechnology*, vol. 8, pp. 1407–1426, 2017.

[68] C. Gu, C. Li, J. Lin, J. Long, J. Huangfu, and L. Ran, “Instrument-Based Noncontact Doppler Radar Vital Sign Detection System Using Heterodyne Digital Quadrature Demodulation Architecture,” *IEEE Transactions on Instrumentation and Measurement*, vol. 59, pp. 1580–1588, 2010.

[69] D. Zheng, S. Zhang, S. Wang, C. Hu, and X. Zhao, “A Capacitive Rotary Encoder Based on Quadrature Modulation and Demodulation,” *IEEE Transactions on Instrumentation and Measurement*, vol. 64, pp. 143–153, 2015.

[70] S. Gajjar, N. Choksi, M. Sarkar, and K. Dasgupta, “Comparative Analysis of Wireless Sensor Network Motes,” in *2014 International Conference on Signal Processing and Integrated Networks (SPIN)*, 2014, pp. 426–431. DOI: 10.1109/SPIN.2014.6776991.

[71] R. P. Narayanan, T. V. Sarath, and V. V. Vineeth, "Survey on Motes Used in Wireless Sensor Networks: Performance & Parametric Analysis," *Wireless Sensor Network*, vol. 8, pp. 67–76, Apr. 2016. DOI: 10.4236/wsn.2016.84005.

[72] F. Karray, W. Jmal, A. Garcia-Ortiz, M. Abid, and A. Obeid, "A Comprehensive Survey on Wireless Sensor Node Hardware Platforms," *Computer Networks*, vol. 144, 2018. DOI: 10.1016/j.comnet.2018.05.010.

[73] P. Kumar and S. Reddy, "Wireless Sensor Networks: A Review of Motes, Wireless Technologies, Routing Algorithms and Static Deployment Strategies for Agriculture Applications," *CSI Transactions on ICT*, 2020. DOI: 10.1007/s40012-020-00289-1.

[74] A. W. Bhat and A. Passi, "Wireless Sensor Network Motes: A Comparative Study," in *2022 9th International Conference on Computing for Sustainable Global Development (INDIACom)*, 2022, pp. 141–144. DOI: 10.23919/INDIACom54597.2022.9763269.

[75] E. Flamand, D. Rossi, F. Conti, *et al.*, "GAP-8: A RISC-V SoC for AI at the Edge of the IoT," in *2018 IEEE 29th ASAP International Conference*, 2018, pp. 1–4. DOI: 10.1109/ASAP.2018.8445101.

[76] A. Pullini, D. Rossi, I. Loi, G. Tagliavini, and L. Benini, "Mr.Wolf: An Energy-Precision Scalable Parallel Ultra Low Power SoC for IoT Edge Processing," *IEEE Journal of Solid-State Circuits*, vol. 54, no. 7, 2019. DOI: 10.1109/JSSC.2019.2912307.

[77] P. D. Schiavone, D. Rossi, A. Di Mauro, *et al.*, "Arnold: An eFPGA-Augmented RISC-V SoC for Flexible and Low-Power IoT End Nodes," *IEEE Transactions on VLSI Systems*, vol. 29, no. 4, 2021. DOI: 10.1109/TVLSI.2021.3058162.

[78] M. Vieira, C. Coelho, D. da Silva, and J. da Mata, "Survey on Wireless Sensor Network Devices," in *EFTA 2003. 2003 IEEE Conference on Emerging Technologies and Factory Automation. Proceedings (Cat. No.03TH8696)*, vol. 1, 2003, 537–544 vol.1. DOI: 10.1109/ETFA.2003.1247753.

[79] J. Wei, L. Wang, F. Wu, Y. Chen, and L. Ju, "Design and Implementation of Wireless Sensor Node based on Open Core," in *2009 IEEE Youth Conference on Information, Computing and Telecommunication*, 2009. DOI: 10.1109/YCICT.2009.5382416.

[80] B. Bengherbia, M. Ouldzmirli, A. Toubal, and A. Guessoum, "FPGA-based Wireless Sensor Nodes for Vibration Monitoring System and Fault Diagnosis," *Measurement*, vol. 101, Jan. 2017. DOI: 10.1016/j.measurement.2017.01.022.

[81] *IGLOO: The Industry's Low-Power FPGAs*, https://www.microsemi.com/product-directory/fpgas/1689-igloo, Accessed: 2023-08-04.

[82] A. Engel and A. Koch, "Heterogeneous Wireless Sensor Nodes that Target the Internet of Things," *IEEE Micro*, vol. 36, no. 6, 2016. DOI: 10.1109/MM.2016.100.

[83] D. A. Patterson and J. L. Hennessy, *Computer Organization and Design RISC-V Edition: The Hardware Software Interface*, 1st. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2017, ISBN: 0128122757.

[84] D. G. Bailey, "Space efficient division on FPGAs," in *Electronics New Zealand Conference (EnzCon'06)*, 2006, pp. 206–211.

[85] S. Mach, F. Schuiki, F. Zaruba, and L. Benini, "FPnew: An open-source multiformat floating-point unit architecture for energy-proportional transprecision computing," *IEEE Transactions on VLSI Systems*, vol. 29, no. 4, 2020.

[86] Xilinx, Inc., *7 Series DSP48E1 Slice - User Guide*, https://docs.xilinx.com/v/u/en-US/ug479_7Series_DSP48E1, Accessed: 2023-07-30, 2018.

[87] D. Castells-Rufas, *RISC-V ISA Coverage*, https://github.com/davidcastells/riscvisacoverage, Accessed: 2023-08-27.

[88] SEGGER, *emFloat — The floating-point library*, https://www.segger.com/products/development-tools/runtime-library/technology/floating-point-library/, Accessed: 2023-07-31.

[89] P. Schregle, *A C++ template class for fixed point math*, https://www.codeproject.com/Articles/37636/Fixed-Point-Class, Accessed: 2023-07-31, 2009.

[90] R. Núñez–Prieto, D. Castells–Rufas, N. Avellana, R. Martínez, and L. Terés, "Processor Optimization of an Energy-Efficient NDIR CO2 Wireless Sensor Node," in *2022 37th Conference on Design of Circuits and Integrated Circuits (DCIS)*, 2022, pp. 01–06. DOI: 10.1109/DCIS55711.2022.9970089.

[91] R. Núñez-Prieto, D. Castells-Rufas, and L. Terés-Terés, "RisCO2: Implementation and Performance Evaluation of RISC-V Processors for Low-Power CO2 Concentration Sensing," *Micromachines*, vol. 14, no. 7, 2023, ISSN: 2072-666X. DOI: 10.3390/mi14071371. [Online]. Available: https://www.mdpi.com/2072-666X/14/7/1371.

[92] Y. Wang, M. Nakayama, M. Yagi, M. Nishikawa, M. Fukunaga, and K. Watanabe, "The NDIR CO_2 monitor with smart interface for global networking," *IEEE Transactions on Instrumentation and Measurement*, vol. 54, no. 4, pp. 1634–1639, 2005. DOI: 10.1109/TIM.2005.851474.

[93] J. A. Seitz and C. Tong, "Texas Instruments LMP 91051 NDIR CO_2 Gas Detection System," 2013.

[94] L. Scholz, A. Ortiz Perez, B. Bierer, P. Eaksen, J. Wöllenstein, and S. Palzer, "Miniature Low-Cost Carbon Dioxide Sensor for Mobile Devices," *IEEE Sensors Journal*, vol. 17, no. 9, pp. 2889–2895, 2017. DOI: 10.1109/JSEN.2017.2682638.

[95] D. Gibson and C. MacGregor, "A Novel Solid State Non-Dispersive Infrared CO_2 Gas Sensor Compatible with Wireless and Portable Deployment," *Sensors*, vol. 13, no. 6, pp. 7079–7103, 2013, ISSN: 1424-8220. DOI: 10.3390/s130607079.

[96] *SpectraPlot v2.0*, https://spectraplot.com/absorption, Accessed: 2023-08-12.

[97] *HITRAN - High-resolution Transmission Molecular Absorption Database*, https://hitran.org/, Accessed: 2023-08-12.

[98] L. S. Rothman, I. E. Gordon, Y. L. Babikov, *et al.*, "The HITRAN 2008 molecular spectroscopic database," *Journal of Quantitative Spectroscopy & Radiative Transfer*, vol. 96, pp. 139–204, 2005.

[99] A. Borodinecs, A. Palcikovskis, and V. Jacnevs, "Indoor Air CO_2 Sensors and Possible Uncertainties of Measurements: A Review and an Example of Practical Measurements," *Energies*, vol. 15, no. 19, 2022, ISSN: 1996-1073. DOI: 10.3390/en15196961. [Online]. Available: https://www.mdpi.com/1996-1073/15/19/6961.

[100] SEGGER, *Embedded Studio for RISC-V, by SEGGER*, https://www.segger.com/products/development-tools/embedded-studio/editions/risc-v/, Accessed: 2023-06-22.

[101] *Specification for the Zfinx RISC-V extension*, https://github.com/riscv/riscv-zfinx/tree/main, Accessed: 2023-08-08.

[102] AMD, Inc. Xilinx, *Alveo U200 Data Center Accelerator Card*, https://www.xilinx.com/products/boards-and-kits/alveo/u200.html, Accessed: 2023-09-17.

[103] AMD, Inc. Xilinx, *Vitis Unified Software Platform*, https://www.xilinx.com/products/design-tools/vitis/vitis-platform.html, Accessed: 2023-09-17.

[104] Cadence Design Systems, Inc., *Genus Synthesis Solution: Massively parallel RTL synthesis and physical synthesis*, https://www.cadence.com/content/dam/cadence-www/global/en_US/documents/tools/digital-design-signoff/genus_rebrand_ds-v1.pdf, Accessed: 2023-08-29.

[105] I.-G. Lee, D. B. Kim, J. Choi, *et al.*, "WiFi HaLow for Long-Range and Low-Power Internet of Things: System on Chip Development and Performance Evaluation," *IEEE Communications Magazine*, vol. 59, no. 7, pp. 101–107, 2021. DOI: 10.1109/MCOM.001.2000815.

[106] Statista GmbH, *Number of Internet of Things (IoT) connected devices worldwide from 2019 to 2023, with forecasts from 2022 to 2030*, https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/, Accessed: 2023-08-31.

[107] Fabio Duarte (Exploding Topics), *Number of IoT Devices (2023)*, https://explodingtopics.com/blog/number-of-iot-devices, Accessed: 2023-08-31.

[108] Transforma Insights, *Global IoT connections to hit 29.4 billion in 2030*, https://transformainsights.com/news/global-iot-connections-294, Accessed: 2023-08-31.

[109] ARM Ltd., *White Paper: The economics of a trillion connected devices*, https://community.arm.com/arm-community-blogs/b/internet-of-things-blog/posts/white-paper-the-route-to-a-trillion-devices, Accessed: 2023-08-31.