論文 / 著書情報
Article / Book Information

題目(和文)	
Title(English)	A Power Side-channel Secure Embedded Processor for Lightweight IoT Applications
著者(和文)	YANGMingyu
Author(English)	Mingyu Yang
出典(和文)	学位:博士(工学), 学位授与機関:東京工業大学, 報告番号:甲第12707号, 授与年月日:2024年3月26日, 学位の種別:課程博士, 審査員:原 祐子,尾形 わかは,髙橋 篤司,本村 真人,佐々木 広,LI YANG
Citation(English)	Degree:Doctor (Engineering), Conferring organization: Tokyo Institute of Technology, Report number:甲第12707号, Conferred date:2024/3/26, Degree Type:Course doctor, Examiner:,,,,,
学位種別(和文)	博士論文
Type(English)	Doctoral Thesis

A Power Side-channel Secure Embedded Processor for Lightweight IoT Applications

Mingyu Yang

Department of Information and Communications Engineering

School of Engineering

Tokyo Institute of Technology

Submitted in partial satisfaction of the requirements for the
Degree of Doctor of Engineering
in Information and Communications Engineering

Advisor: Associate Professor Yuko Hara

January 2024

Abstract

The development of the Internet of Things (IoT) has brought about a transformative shift in daily life, creating a ubiquitous network of interconnected devices with profound impacts across various domains, particularly in healthcare through eHealth systems. This expansion facilitates real-time data processing and enhances medical care by leveraging vast amounts of biomedical data. However, the rapid integration of IoT, especially in eHealth, raises significant challenges in data privacy, security, and the need for secure hardware platforms.

A critical vulnerability in IoT security is the threat posed by power side-channel attacks. These attacks, capable of extracting secret information based on power consumption, are a significant risk in IoT devices that handle sensitive data. The inherent limitations of IoT devices, such as constrained power, memory, and the need for energy efficiency, further complicate the implementation of effective countermeasures.

Addressing these challenges, this research proposes a novel embedded processor optimized for lightweight IoT applications, emphasizing power efficiency, circuit area and security. This processor features a unique Instruction Set Architecture (ISA) designed to minimize circuit area and power consumption while maintaining processing efficiency. This balance is crucial for IoT applications where resources are constrained, such as in wearable devices, sensor networks, and smart homes. Furthermore, the processor design significantly mitigates risks associated with power side-channel attacks. A hardware/software cooperative approach is adopted to optimize system performance and security. On the software side, hardware-

aware optimizations are conducted to ensure secure and efficient operation with the underlying hardware. On the hardware side, a bottom-up methodology defines a minimal ISA and architectural structure, embedding efficiency and security as core considerations.

Experiments demonstrate the effectiveness of the proposed processor design in real-world scenarios. The experimental results in terms of side-channel security validate the processor's robustness in safeguarding secret information against power side-channel attacks. Furthermore, experiments in eHealth applications demonstrate the processor's capability to adeptly manage lightweight IoT applications while operating under stringent resource and power constraints. This dual achievement of operational efficiency and enhanced side-channel security, even in environments with limited energy resources, marks a notable advancement in the field of IoT.

In summary, the proposed embedded processor not only provides the ability of processing lightweight IoT applications with low power consumption and minimal resource usage, but also offers robust defense against power side-channel attacks, safeguarding sensitive information in resource-constrained environments. This research contributes to the development of a secure IoT ecosystem, balancing comprehensive security with the constraints of IoT devices and paving the way for advancements in IoT applications.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

As the Internet of Things (IoT) continues to advance, an ever-expanding network of interconnected devices is reshaping the daily life for individuals across the globe. The exponential expansion of IoT technologies has generated a pressing demand for the real-time processing of data within IoT applications. This need becomes particularly pronounced as IoT devices proliferate across diverse domains, ranging from healthcare to smart cities [1, 2, 3]. Within these domains, there is a specific emphasis on the development of healthcare domain, often denoted as eHealth [4] – a plethora of monitoring applications have been realized, significantly enhancing patient care and health management.

eHealth facilitates the collection of vast amounts of biomedical data, enabling more personalized and predictive medicines. For example, wearable devices can track and analyze a patient's physical activity and physiological parameters, providing valuable insights for preventive healthcare measures [5]. Moreover, IoT technologies are instrumental in managing chronic diseases, where continuous monitoring and timely interventions can significantly improve patient outcomes [6]. However, this rapid integration of eHealth also brings challenges, particularly concerning data privacy [7], security [8], and the need for secure hardware platform [9] to handle the large volume of sensitive data being generated. As such, the future of

eHealth is not only about advancement in applications but also about addressing these challenges to ensure protection to sensitive information.

In the rapidly evolving landscape of IoT that integrates devices as critical aspects of daily life, the paramount challenge lies in fortifying these systems against sophisticated attacks [10], particularly side-channel attacks [11]. Among a variety of side-channel attacks against different sources of information leakage, power side-channel attacks pose a formidable and insidious threat, capable of clandestinely extracting secret information from electronic devices based on power consumption [12]. This vulnerability is particularly concerning in IoT environments where devices often handle sensitive data, from personal health information in wearable devices to security details in smart home systems [13, 14]. Moreover, the inherent resource limitations of IoT devices, such as constrained power, limited memory, and energy efficiency requirements, add a significant layer of complexity to the task of implementing robust countermeasures [10]. These constraints often preclude the use of traditional, resource-intensive countermeasures, necessitating innovative approaches tailored to the unique characteristics of IoT devices. Therefore, striking a balance between comprehensive security and the constraints posed by limited resources remains a central challenge in the development of a secure IoT ecosystem. This entails not only the implementation of lightweight cipher algorithms [15] but also a holistic approach to IoT hardware design that incorporates security as a fundamental component [16].

In this research, we propose a power side-channel secure embedded processor that is optimised for lightweight IoT applications. Rather than aiming to outperform other embedded processors across all applications, this research focuses on providing an embedded processor optimized for specific targeted lightweight applications under its instruction format. This design paradigm not only addresses the crucial need for power efficiency in such applications but also aligns with the prevalent security concerns in IoT environments. This embedded processor possesses a

unique instruction set architecture (ISA) design such that small circuit area and low power consumption can be achieved. The ISA is tailored to enhance power efficiency without compromising much processing efficiency, thereby enabling the processor to handle lightweight IoT tasks effectively. This is particularly vital in IoT applications where resources are limited, such as in wearable devices, sensor networks, and smart homes. In addition, we meticulously determined the optimal memory size for the proposed embedded processor. By carefully balancing the memory allocation, we aimed to ensure that the processor has sufficient memory for its intended tasks while avoiding the excess power consumption that can come with larger memory sizes. The processor's ability to operate with minimal power requirements while securely processing data makes it an ideal solution for the next generation of IoT devices, where power supply and data security are of paramount importance. Furthermore, the innovative design of this embedded processor significantly mitigates the risks associated with power side-channel attacks, a critical vulnerability in many existing IoT devices. Addressing the threats from power side-channel attacks, we adopt a hardware/software cooperative approach, which strategically aligns hardware implementation with software design to optimize overall system performance and security. On the software side, our focus is on conducting hardware-aware optimizations. These optimizations are meticulously tailored to work securely and efficiently with the underlying hardware. On the hardware side, we employ a bottom-up methodology. This involves defining a minimal ISA and its corresponding architectural structure. By integrating the hardware and software perspectives, the processor offers a more robust defense against power side-channel attacks, ensuring that sensitive information remains protected even in the most resource-constrained environments.

This dissertation is organized as follows:

Chapter 2 lays the foundational groundwork essential for understanding the context of this research. Initially, it introduces the concept of embedded processors

specifically tailored for IoT applications, setting the stage for their relevance and application. This is followed by an exploration of the imperative need for side-channel protection in these embedded processors, highlighting the significance of security in this domain. Finally, the chapter introduces SubRISC+, a low-power embedded processor, characterized by its distinctive ISA, thereby establishing a comprehensive background for the subsequent chapters.

Chapter 3 presents the development of our novel embedded processor, designated as the Small and Secure Processor (SSP), designed with a focus on power side-channel security. Initially, this chapter delineates the design of the ISA of the SSP, laying the foundational concepts. Subsequently, it delves into the detailed architectural design of the proposed processor, highlighting its unique features and capabilities in the context of embedded systems.

Chapter 4 introduces a novel hardware/software cooperative design approach for cryptosystems, with the implementation on the embedded processor SSP. This cooperative approach aims to significantly enhance the power-performance-area (PPA) metrics, while concurrently mitigating power side-channel leakages. Through our comprehensive evaluation, we demonstrate that our approach not only outperforms contemporary state-of-the-art solutions in terms of PPA efficiency but also offers superior protection against power side-channel attacks.

Chapter 5 offers an in-depth exploration of eHealth applications developed using the Dynamic Time Warping (DTW) algorithm on the embedded processor SSP. This chapter not only examines the software aspects of these eHealth applications but also provides a critical evaluation of the embedded processor hardware implementations integral to their functionality. By doing so, it offers a comprehensive view of both the software and hardware components, demonstrating the effectiveness of our implementations.

Chapter 6 concludes this dissertation with a comprehensive summary of the key contributions and future directions.

Chapter 2

Background

This chapter lays the foundational groundwork necessary for this research, commencing with an introduction to embedded processors specifically designed for IoT applications. This is followed by a detailed discussion on the critical need for side-channel protection in these embedded processors, highlighting the importance of security countermeasures in this specialized field. Finally, the chapter introduces SubRISC+, a low-power embedded processor, which is notable for its unique ISA.

2.1 Embedded Processors for IoT Applications

Embedded processors, serving as the backbone of a myriad of small-scale and portable electronic devices, are specialized types of processors designed with specific considerations [17]. Unlike conventional processors, which are often built for general-purpose computing and high-performance tasks, embedded processors are tailored for very specific functions within an embedded device [18]. This specialization directly influences their design and operational parameters, necessitating a unique set of constraints due to the nature of their deployment and applications.

One of the primary constraints is power efficiency. Embedded processors are frequently deployed in environments where access to power is limited or where prolonged battery life is essential [19]. This requirement starkly contrasts with

conventional processors, which are typically part of larger systems with a steady power supply. For example, a processor in a desktop computer or server might operate at several gigahertz and consume a substantial amount of power, measured in tens to hundreds of watts. On the other hand, an embedded processor, such as one found in a wearable device or a sensor network, operates at a much lower frequency, typically in the range of a few megahertz, and its power consumption measured in milliwatts [19].

Additionally, embedded processors face limitations in computational capacity and hardware resources. While a conventional processor might boast multiple cores and high-speed memory access, an embedded processor dedicated to IoT applications is designed with a smaller silicon footprint. This compact design is crucial for integration into devices where space is at a premium but comes with trade-offs in terms of processing power and memory capacity [20]. As a result, these processors must be optimized to perform their designated tasks efficiently within these constraints.

Embedded processors in IoT applications are responsible for processing the data collected by sensors, managing communications, and executing control commands, often while operating under stringent power and space constraints [21]. Their ability to perform reliably in a compact form factor with minimal energy consumption makes them essential for the widespread adoption and success of IoT technologies. The property on real-time of IoT applications also add on the need in performance of the embedded processor. Many IoT applications requires real-time data processing [22] such that a result or a decision must be generated within a defined time frame. This also contributes to the contradiction in the need of higher computational power and lower power consumption.

The differences in operational parameters between embedded processors and conventional processors underscore the differences in their design considerations. Embedded processors must prioritize efficiency, compactness, and specific function-

ality, while conventional processors often emphasize computational power and versatility. This divergence is reflective of the distinct roles they play in the technological ecosystem. Embedded processors are integral in applications ranging from sensor networks [23] to industrial control systems [24], where they control specific functions or processes. One specific applications domain to be highlighted is IoT applications, where embedded processors also play a critical role.

In this context, understanding the unique characteristics and design challenges of embedded processors becomes crucial for developing innovative and effective IoT applications.

In the following, we summarize a set of most important factors known as power-performance-area (PPA) and corresponding measurements in embedded processor design.

Power and Energy Consumption:

One of the foremost considerations in the design of embedded processors is power and energy consumption. This is particularly critical in battery-powered or energy-harvesting devices, where energy efficiency dictates not only the performance but also the practical usability and lifespan of the product. Designers of embedded processors strive to optimize power usage at every level, from the choice of fabrication process to the architecture of the processor itself. Techniques such as dynamic voltage and frequency scaling [25], power gating [26], and the use of a energy-efficient ISA [27] are employed to minimize power consumption while maintaining adequate performance. In contemporary digital circuit design, the Complementary Metal–Oxide–Semiconductor (CMOS) technology is predominantly utilized [28]. CMOS, a type of Metal-Oxide-Semiconductor Field-Effect Transistor (MOSFET), employs complementary pairs of p-type and n-type MOSFETs. This technology is widely adopted in the fabrication of digital processors and various other digital circuits. A critical aspect of CMOS-based digital pro-

cessor design is the minimization of power consumption. To comprehensively address this, it is imperative to understand the components of power consumption in CMOS circuits. As detailed in [29], the total power consumption of CMOS circuits comprises two primary components: dynamic power consumption ($P_{dynamic}$) and static power consumption (P_{static}). The dynamic power consumption is associated with the charging and discharging of capacitive loads during the switching of transistors, whereas the static power consumption arises from leakage currents and other passive power dissipations when the circuit is in a non-switching state.

$$P_{total} = P_{dynamic} + P_{static} \tag{2.1}$$

$$P_{dynamic} = \alpha C V_{DD}^2 f \tag{2.2}$$

$$P_{static} = I_{static} V_{DD} \tag{2.3}$$

where α is the activity factor, C is the load capacitance, V_{DD} is the supply voltage, f is the clock frequency, and I_{static} is the total current flowing through the circuit. These formulas shows that, power consumption of CMOS circuits can be reduced by reducing circuit area, V_{DD}, and f.

On the other hand, the energy consumption can be computed from an integral of power and time as following:

$$W = \int P dt \tag{2.4}$$

where W is the energy, P is the power and t is the time. This formula shows that improvements in either reducing power consumption or operational time are key strategies in enhancing the energy efficiency.

Performance:

Performance, or the speed at which the processor completes its tasks, is another crucial factor in embedded processor design. In many applications, such as real-time systems in automotive or industrial control, the ability to process data quickly and respond to inputs react within specific time bounds is essential [30]. However, achieving high processing speeds must be balanced with power constraints, as higher speeds typically result in increased power consumption. Embedded processor designers often need to find the optimal balance between speed and power usage, tailoring the processor's capabilities to the specific needs of the application, whether it be high-speed data processing or prolonged idle periods with intermittent bursts of activity. Measurement of performance in embedded processor design is typically conducted through two primary methods: actual execution and simulation. In actual execution, performance is measured directly as the processor operates in real-time. Alternatively, in simulation, performance can be estimated by considering the number of cycles required for a given task and the target operating frequency of the processor. This simulated data provides a theoretical processing time under specified conditions. Equipped with this information on processing time, designers can critically assess whether the current design meets the predefined performance requirements. If the processing time exceeds acceptable thresholds, it signals the need for further optimizations. This iterative process of measurement and adjustment is crucial for refining the design to achieve the desired balance between speed, efficiency, and resource utilization.

Circuit Area:

The physical size or circuit area of the processor is a significant consideration, especially in applications where space is limited. Smaller processors are desirable in wearable technology [31], eHealth devices [32], and many consumer electronics, where they need to fit into compact spaces. Reducing the circuit area can also have the benefit of reducing power consumption, as both the dynamic and static power are reduced. However, minimizing the size of the processor often

involves trade-offs in terms of processing power and the number of features that can be included. Designers must carefully consider how to integrate essential functions within these physical constraints. In Application-Specific Integrated Circuit (ASIC) designs, the circuit area is quantified either in terms of actual physical dimensions (typically measured in square millimeters or micrometers) or in gate equivalents (GE). The actual area measurement provides designers with insights into how their design will fit into the silicon using the current fabrication process. In contrast, GE offer a standardized measurement of circuit size that is independent of the fabrication process [33], serving as a general reference. Conversely, when it comes to Field-Programmable Gate Array (FPGA) designs, the metric for circuit area shifts to the number of resources utilized on the target FPGA chip. These resources typically include Look-Up Tables (LUTs), Flip-Flops (FFs), Block Memory, and Digital Signal Processors (DSPs). This resource-based measurement is essential in FPGA designs as it reflects the utilization and efficiency of the FPGA design, which directly impacts the feasibility and performance of the design.

2.2 Side-channel Security for Embedded Processors

In the field of embedded processors, particularly within the IoT, cipher algorithms are paramount in safeguarding sensitive information [34]. Cipher algorithms are critical in safeguarding sensitive information across a wide range of applications. Ideally, the security of a cipher algorithm is such that it can only be compromised through brute force attacks, which involve exhaustively searching all possible key combinations. Given the vastness of this search space, executing a brute force attack would typically require an impractically long time period[35]. This makes it virtually impossible to decipher the protected data within any reasonable timeframe, thereby ensuring robust data security. However, in practical scenarios, various advanced attack methods have been devised that can significantly reduce

the required search space. These methods are capable of narrowing down the range of possible keys to a more manageable size [36], allowing the encrypted data to be potentially compromised within a feasible time. These attacks pose a critical challenge to maintaining data security and necessitates continuous advancements and updates in cipher algorithm design to counteract these emerging threats.

The level of security is vital in an era where data privacy and protection are paramount, particularly in IoT systems that often handle personal and sensitive data. In the case of IoT eHealth, devices face significant risks by potentially leaking sensitive personal and medical data, compromising the confidentiality and integrity of health information. Such leakages can lead to incorrect medical diagnoses and treatments, with serious consequences on patients' treatments and health conditions [37]. In extreme cases, it can even lead to life-threatening situations. Despite the sophisticated design and mathematical validation of cipher algorithms to resist conventional attacks, embedded processors face unique vulnerabilities when exposed to side-channel attacks that exploit physical information like power consumption, electromagnetic emission, and processing time [38].

These threats from side-channel attacks are different from covert channels, which are emerging from the complex architectural designs of high-end devices. Examples of such covert channels include branch speculation and complex memory hierarchies. These covert channels can be exploited to surreptitiously transmit information between processes that should ideally be isolated from each other. However, the security landscape within the realm of IoT presents unique challenges that lies beyond the scope of such covert channels.

The challenge in resolving the vulnerabilities for embedded processors is twofold. Firstly, these processors are designed to be compact and energy-efficient, which often leads to a difficult design question on trade-offs in terms of security [39]. This inherent constraint makes it challenging to implement robust countermeasures

without compromising the device's performance and power efficiency. In the field of eHealth, large area and power overheads of such countermeasures, make them not compatible with the resource-constrained eHealth devices [40]. Secondly, the nature of use scenarios of embedded processors means easy accessibility for the attackers [41] such as power consumption can be easily captured and analyzed to reveal secret information [42]. In the context of IoT, where embedded processors are ubiquitous, the implications of these vulnerabilities are significant. For instance, a smart sensor node or a wearable health monitor, might be processing sensitive data that, if intercepted via side-channel analysis, could compromise user privacy and security.

Countermeasures against power side-channel attacks have become a critical area of research in the field of embedded processor and cipher algorithm design. The primary strategies to mitigate these risks are categorized into two approaches: hiding [43] and masking [44].

The hiding approach targets on making it more difficult for an attacker to interpret the power consumption, this method increases the complexity and effort required to successfully execute a side-channel attack. Techniques under this category include reducing the Signal-to-Noise Ratio (SNR) and implementing timing disarrangement [45]. Reducing the SNR involves diminishing the visibility of the power consumption patterns associated with the secret information, making it harder to distinguish them from noise or non-relevant signals[46]. Timing disarrangement, on the other hand, involves altering the temporal characteristics of operations, such as introducing random delays or varying the execution order of instructions [47], thereby disrupting the predictable timing patterns that attackers often rely on.

Masking takes a different approach by severing the discernible relationship between side-channel information and the secret information being processed. This is achieved by randomizing the intermediate values of the cipher algorithm [48].

Fundamentally, the principle of masking aligns with the concept of secret sharing [49]. In secret sharing, sensitive data is divided into multiple random parts, referred to as shares. Each of these shares, on its own, does not reveal any meaningful information about the original secret. However, when combined in a specific manner, these shares can be constructively reassembled to recover the original secret data. Similarly, in masking, secret information are divided into multiple shares, ensuring that the power consumption observed during these operations do not yield any useful information that could facilitate a side-channel attack.

Masking is indeed a powerful and theoretically secure approach to countering power side-channel attacks. However, its practical implementation, especially in embedded processors, often encounters significant challenges due to various constraints of the embedded processors [50]. Nevertheless, one of the other critical issues is the gap that exists between the theoretical design of masking techniques and their secure implementation in actual hardware and software environments [51]. This gap result in a type of vulnerability that arises from the improper integration of software and hardware. In embedded processors, this vulnerability can lead to unintended leakages. For example, if a processor's architectural behavior, like the timing of operations, is not adequately considered in the masking design, it can inadvertently create leakages that expose critical information, despite the theoretical robustness of the masking algorithm. Bridging the gap between masking design and secure implementation requires a meticulous approach that considers the intricate interplay between software algorithms and hardware architecture. Addressing this gap is crucial to ensure that the implemented masking not only theoretically secures the data but also provides robust protection in real-world IoT applications.

2.3 SubRISC+ Processor

SubRISC+ [52], is an innovative low-power embedded processor, characterized by its unique ISA design that achieves both minimal circuit size and low power consumption. This processor is an evolution of the one-instruction set computer (OISC) concept, a minimalistic approach to processor design that utilizes a single instruction to perform all operations, thereby achieving remarkable simplicity and efficiency.

The foundational instruction of SubRISC+, based on the OISC model, is subtraction and branch on negative. This operation alone is powerful enough to render the processor Turing complete, meaning it can theoretically perform any computation. In the following, this operation is commonly referred to simply as subtraction or sub for brevity.

In an effort to enhance execution speed and optimize energy consumption, the original OISC processor design was further developed into SubRISC+ [52] . This evolution involved extending the ISA of the original OISC design. SubRISC+ operates with four instructions: subtraction, bitwise AND, shift, and memory. These instructions were chosen for their fundamental role in computing operations and their ability to be executed efficiently.

Furthermore, the instructions in SubRISC+ are designed to be flexible in terms of their bit-length, allowing them to be expressed either in 16 bits or 32 bits. This flexibility facilitates compactness in code size, enabling the processor to handle immediate values and branch operations effectively. Such versatility in instruction size is crucial for adapting to the varying demands of different embedded applications, from simple control tasks to more complex computational operations.

The result of these enhancements is a processor that not only improves computing and power efficiency but also maintains simplicity. This is achieved with minimal

circuit overhead, a critical factor in embedded system design where space and power are at a premium. The minimalistic yet versatile nature of SubRISC+ makes it an ideal candidate for a wide range of low-power embedded applications, from wearable devices to sensor networks.

The instructions of SubRISC+ and their functionalities are concisely summarized in Table 2.1, providing a clear overview of how each instruction contributes to the processor's capabilities. The detailed instruction formats are shown Table 2.2. Also, the definitions of operand A and B are shown in Table 2.3 and Table 2.4, respectively.

Table 2.1: Instruction set of SubRISC+

Operation	Mnemonic	Behavior
Sub	$sub\ R_A, R_B, R_D$	R[R_D] = R[R_B] - R[R_A]
	$subi\ I, R_B, R_D$	R[R_D] = R[R_B] - I
	$subi\ R_A, I, R_D$	R[R_D] = I - R[R_B]
AND	$and\ R_A, R_B, R_D$	R[R_D] = R[R_B] & R[R_A]
	$andi\ I, R_B, R_D$	R[R_D] = R[R_B] & I
	$andi\ R_A, I, R_D$	R[R_D] = I & R[R_B]
Shift	$shr/shl\ R_A, I, R_D$	R[R_D] = R[R_A] >> / << I
Memory	$mr\ I_{offset}, R_B, R_D$	R[R_D] = M[R[R_B]-I_{Offset}]
	$mw\ I_{offset}, R_B, R_A$	M[R[R_B]-I_{Offset}] = R[R_A]

R[ ] (M[ ]): register or memory access

R_A and R_B: input registers, R_D: output register

I: immediate value

"/": either of two options is selected

Bits supported by the shift operation can be expressed as $8b + n$, where $b = 0, 1, 2, 3$ and $n = 0, 1, 2, 3$

Table 2.2: Instruction format of SubRISC+

(1) Main Block of *Sub* and *AND*

sub: 2'00 and: 2'01	Branch Flag (1 bit)	Reg.ID of Operand A (4 bits)	Reg. ID of Operand B (5 bits)	Reg. ID of Operand D (4 bits)
Branch Block (optional for *Sub* and *AND*): Relative Branch Target (13 bits)			Jump Condition Bits (3bits)	

(2) Main Block of *Shift*

2'11	Register Flag (1 bit)	Reg.ID of Operand A (4 bits)	Reg. ID or Immediate (5 bits)	Reg. ID of Operand D (4 bits)

(3) Main Block of *Memory*

2'10	mr:1'0 mw:1'1	Address Offset	Reg. ID or Operand B (5 bits)	Reg. ID of Operand D (4 bits)

(4) Main Block of *Subi* and *ANDi*

subi: 2'00 andi: 2'01	17-th Immediate Bit	4'1111	Reg. ID of Operand B (5 bits)	Reg. ID of Operand D (4 bits)
subi: 2'00 andi: 2'01	17-th Immediate Bit	Reg.ID of Operand A (4 bits)	Zero Extension Sign Extension	Reg. ID of Operand D (4 bits)
Branch Block (optional for *Subi* and *ANDi*): Immediate Value (16 bits)				

An architectural overview of SubRISC+ is illustrated in Fig. 2.1. In this figure, the various components of the processor are represented by short-forms, which include PC (Program Counter), ALU (Arithmetic Logic Unit) and RF (Register File).

In the design of SubRISC+, an important aspect is its memory architecture, which adopts the Harvard architecture. This architecture is characterized by the segregation of memory spaces for instructions and data, a decision that brings several

Table 2.3: Definitions of operand A

Operand A	Access Value
0	Constant 0
1	Constant 1
2	Constant -1
3	(Immediate Value)
4-15	RF[ ]

Table 2.4: Definitions of operand B

Operand B	Access Value
0-15	RF[ ]
16	Constant 0
17	Constant 1
18	Constant -1
20	Zero extension
24	Sign extension

advantages to the processor's overall performance and efficiency. In SubRISC+, these separate memory spaces are referred to as Imem (Instruction Memory) and Dmem (Data Memory), respectively. This separation allows for simultaneous access to instructions and data, enhancing processing speed and efficiency, a feature particularly beneficial in embedded systems where performance and power efficiency are paramount.

Further refining this architecture, the Imem in SubRISC+ is partitioned into two distinct sections: high instruction memory (Hi-Imem) and low instruction memory (Lo-Imem). This division is a strategic design choice that aligns with the processor's support for both 16-bit and 32-bit instruction formats. The Hi-Imem is optimized for handling the 32-bit instructions, while the Lo-Imem is tailored for the more compact 16-bit instructions. This bifurcation enables the processor to handle instructions more efficiently, balancing between compact memory usage and enhanced functionalities.

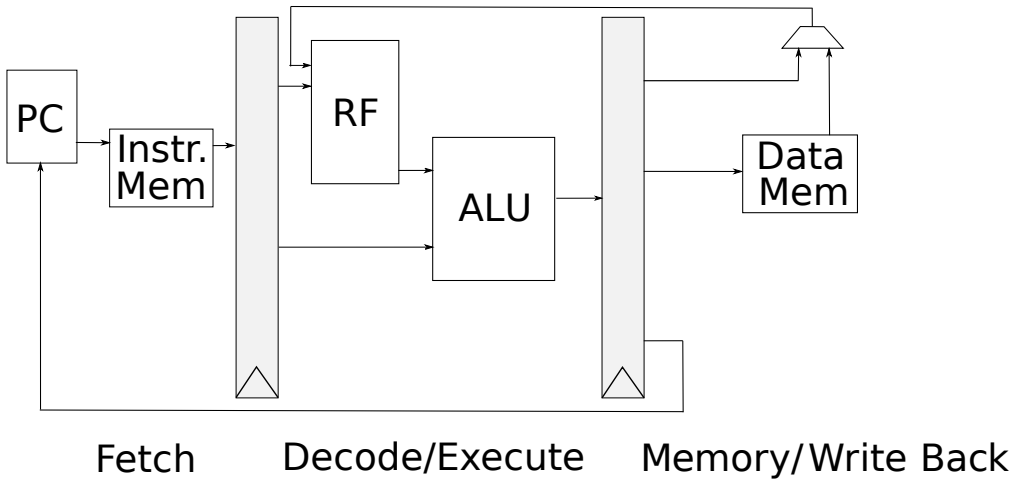


Figure 2.1: Architectural overview of SubRISC+

Because of the small core circuit size and optimized memory size, the entire IC design using the TSMC 65nm low power process can be fit into a die in 1mm×1mm square. The layout of SubRISC+ processor is shown in Figure 2.2. Fabricated SubRISC+ ICs are packaged in quad flat package 64 (QFP64). In addition, a packaged SubRISC+ is shown in Figure 2.3 with a 10 yen coin.

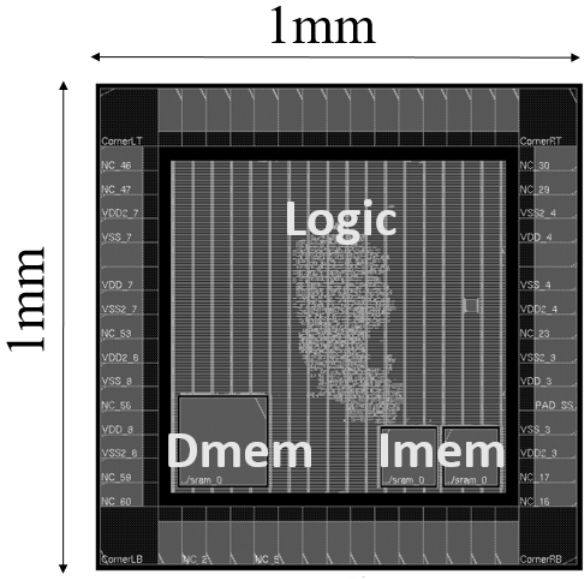


Figure 2.2: Layout of the SubRISC+ processor [53]

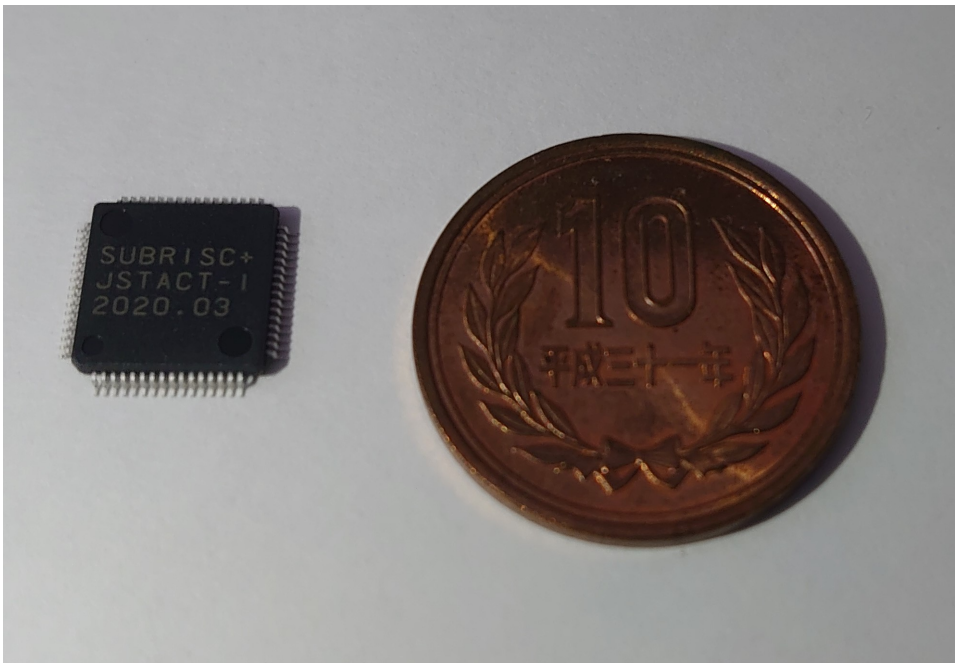


Figure 2.3: Fabricated SubRISC+ processor

Chapter 3

The Small and Secure Processor (SSP)

This chapter introduces our proposed power side-channel secure embedded processor SSP. This chapter begins by elaborating the design of the SSP's ISA in Section 3.1, thereby establishing the essential concepts underpinning our processor design. Following this foundational overview in Section 3.2, this chapter transitions into an in-depth exploration of the architectural design of SSP.

3.1 ISA of the SSP

In response to the increasing demand for power side-channel secure embedded processors tailored for IoT applications, we introduce the SSP. The SSP is specifically designed to offer both robust security against power side-channel attacks and optimal PPA in IoT devices. It achieves this by implementing a carefully selected set of instructions that cater to the needs of targeted applications while minimizing circuit area and power consumption. SubRISC+ has been chosen as the foundation for developing the SSP owing to its compact ISA and superior area and power efficiency. These characteristics are essential for designing a secure embedded processor for IoT applications, where circuit area and power are often limited.

The ISA of the SSP includes support the following instructions, subtraction, logic, shift with arbitrary bits, and memory access. These instructions were chosen not only for their fundamental role in computing but also for their ability to be executed with minimal power and circuit overhead. The ISA definition and the specific format of these instructions are concisely presented in Table 3.1 and Fig. 3.1, respectively. As shown in Fig. 3.1, instructions are basically in a 16-bit format, and an optional 16-bit block is utilized to handle an immediate value or branch. Operations that are not directly supported by this ISA can be computed by combining the instructions listed in Table 3.1.

In the SSP, the ISA utilizes two input registers, denoted as R_A and R_B, and an output register, R_D. Additionally, an immediate value, represented as I, is also supported. The instruction format is versatile, with a basic 16-bit structure and an optional block. This optional block enhances the processor's capability by allowing the handling of immediate values or branch operations, further contributing to the processor's flexibility and efficiency.

The ISA of SSP is carefully designed to meet the necessary functional requirements while avoiding any superfluous complexity that could compromise security or efficiency. The key updates of SSP's ISA are the logic operation which enables bitwise XOR and the shift instruction that support arbitrary bits. These updates are introduced to aid the secure implementing of cipher algorithms, details on how the instructions are implemented are discussed in Section 3.2.

One of the key features of the SSP's instruction set is the function bit, a 1-bit flag signal within each instruction. This flag is crucial for determining the operational mode of various instructions. For instance, in the subtraction instruction (Sub), it decides whether the optional block is used. In the bitwise logic instruction (Logic), it selects between AND and XOR operations. In the memory access instruction (Memory), it specifies the direction of data assignment between the register file and memory, and in the shift instruction (Shift), it determines the shift direction.

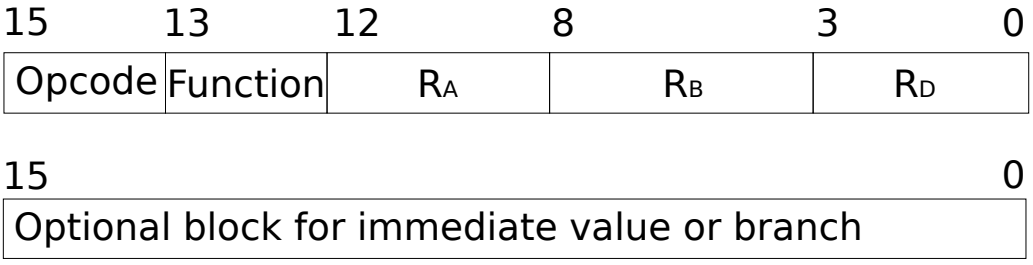


Figure 3.1: Instruction format of the SSP processor [54]

Table 3.1: Instruction set of SSP

Operation	Mnemonic	Behavior
Sub	*sub* R_A, R_B, R_D	R[R_D] = R[R_B] - R[R_A]
	subi I, R_B, R_D	R[R_D] = R[R_B] - I
	subi R_A, I, R_D	R[R_D] = I - R[R_B]
Logic	*and* R_A, R_B, R_D	R[R_D] = R[R_B] & R[R_A]
	xor R_A, R_B, R_D	R[R_D] = R[R_B] $\oplus$ R[R_A]
Shift	*shr/shl* R_A, I, R_D	R[R_D] = R[R_A] >> / << I
Memory	*mr* I_{offset}, R_B, R_D	R[R_D] = M[R[R_B]-I_{Offset}]
	mw I_{offset}, R_B, R_A	M[R[R_B]-I_{Offset}] = R[R_A]

R[ ] (M[ ]): register or memory access

R_A and R_B: input registers, R_D: output register

I: immediate value

"/": either of two options is selected

Bits supported by the shift operation are arbitrary within 32-bit

In our deliberation over the optimal number of instructions for the SSP, extensive consideration was given to the optimal number of instructions. Reducing the number of instructions to one or two, implying no opcode or a 1-bit opcode, was evaluated. Insights from developing SubRISC+ [52] suggested that such a reduction would significantly impair performance in processing IoT applications. Specifically, for the OISC processor which utilize only one instruction, a one bit

shift operation or a bit-wise AND operation requires 230 and 300 instructions, respectively [52].

Conversely, the idea of expanding the instruction set to include a 3-bit opcode or more was also explored. To assess the impact of increasing the number of instructions, we integrated two additional instructions commonly found in embedded processors: addition and bit-wise OR. To ensure a fair comparison, we conducted experiments under the same parameters detailed in Section 5.4. The detailed setup is described as following. We used Synopsis Design Compiler-F2011.09-SP2 with the libraries of TSMC 65nm low power technology to report circuit area. The synthesis conditions were set as following, supply voltage of 1.0 V, clock frequency of 50 MHz, temperature of 25 °C and typical-typical (TT) process corner. The details of the target applications are the same as detailed in Section 4.3. Adding these instructions necessitated using at least one extra bit for the opcode. This requirement led to the usage of an additional bit from other functionalities: a bit from the number of shift bits in shift instructions and one bit from operand B in other instructions. Consequently, this resulted in limitations in the shift instruction's bit support and the use of constant values in operand B. The summary on the discussion of the optimal four-instruction ISA is shown in Table 3.2.

Table 3.2: Summary on the discussion of the number of instructions

Numbers	Below 4	Optimal (4)	Above 4
Circuit Area	Minor decrease (0.9x)	1x	Minor increase (1.1x)
Number of Cycles	Major increase (100x or more)	1x	Minor increase (1.1x)

From a circuit area perspective, this expansion of instructions resulted in a 1.5 times increase in the ALU module's area and a 1.1 times increase in the overall core area. While the increase in core area may seem modest, it did not translate into improved performance. In fact, for three target cipher applications, the number of cycles either increased or remained unchanged. This performance degradation

is attributed to the loss of certain functionalities due to the aforementioned bit reallocation.

Similarly, we conducted a discussion on the number of register file entries. One one hand, by decrease the number of register file entries to 8 or less, the circuit area can be further reduced. However, this will result in the target cipher applications do not have enough register file entries to handled the operations. So that cipher applications must use much more instruction to read and write from the memory. Under this scenario, fulfilling the security requirements also makes the performance even worse. The details of the secure implementation approach are shown in Section 4.3.2. On the other hand, increase of register file entries to 32 or more will significantly increase the circuit area while bringing no observable performance gain. The summary on the discussion the optimal register file entries is shown in Table 3.3

Table 3.3: Summary on the discussion of the register file entries

Numbers	Below 16	Optimal (16)	Above 16
Circuit Area	Decrease (0.7x)	1x	Increase (1.5x)
Number of Cycles	Major increase (3x or more)	1x	No decrease

In summary, both reducing and increasing the current number of instructions or register file entries led to suboptimal outcomes when compared to the existing four-instruction ISA. Thus, after thorough consideration of the trade-offs involved, we have decided to retain the current four-instruction ISA for our processor, balancing efficiency and functionality for our targeted applications.

3.2 Architectural Design of the SSP

In the development of our proposed processor for IoT applications, which aims to be resilient against side-channel attacks, our starting point was the established

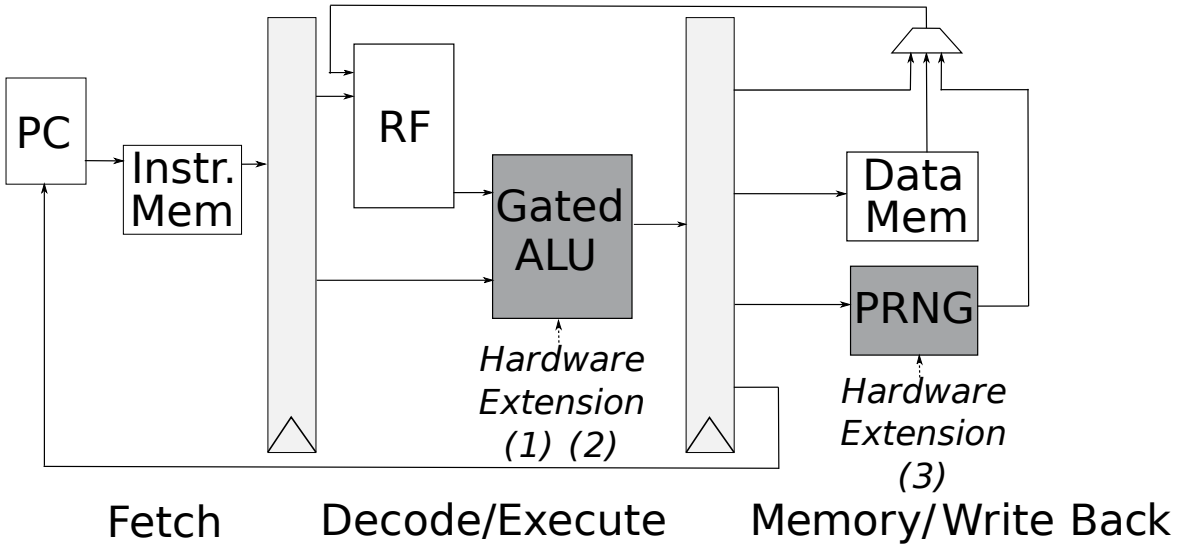


Figure 3.2: Architectural overview of our proposed design [54]

SubRISC+ architecture. Based on our decisions regarding the ISA and identified microarchitectural sources of leakage in [55], we meticulously designed and implemented three critical architectural extensions to SubRISC+. Each extension was conceived with dual objectives: firstly, to significantly enhance the processor's defense mechanisms against potential power side-channel attacks, a prevalent threat in the IoT landscape; and secondly, to maintain the processor's inherent attributes of efficiency and compactness which are vital for IoT applications. This multi-faceted approach ensured that our processor not only stands resilient in the face of evolving security threats but also aligns with the performance and hardware resource constraints characteristic of IoT devices. The three major hardware extensions are introduced as following.

An architectural overview of the proposed processor, highlighting these extensions, is illustrated in Fig. 3.2. In this figure, the various components of the processor are represented by short-forms, which include PC (Program Counter), ALU (Arithmetic Logic Unit), RF (Register File), and PRNG (Pseudo-Random Number Generator).

1. Bitwise XOR and Shift Operations:

As a significant extension to the SubRISC+ architecture, we introduced bitwise XOR and shift with arbitrary values instructions. These extensions required modifications to the ALU and the instruction decoder. The ALU was reconfigured to handle these new operations efficiently, while the instruction decoder was updated to recognize and process the extended instruction set. Additionally, the instruction format was expanded to support these instructions, ensuring compatibility and seamless integration within the existing architecture. This enhancement not only broadens the processor's computational capabilities but also aids in implementing cipher algorithms, which are essential for security in IoT applications.

2. Gating in the ALU:

Recognizing the importance of minimizing power leakage, particularly as highlighted in [55] on power side-channel attacks, we implemented a gating scheme in the ALU. While the specifics of the ciphers are publicly disclosed, a key distinction lies in the deliberate concealment of operational details. The primary objective is to safeguard the secrecy of sensitive data, achieved through the use of random numbers and masking techniques. Unintentional computations, which can occur without ALU gating, are undesirable. This ALU gating approach specifically targets the reduction of unnecessary switching activities, which are known to contribute significantly to power leakage. By employing a input gating technique, the ALU now activates only the relevant parts needed for the current operation, based on the opcode of the instruction being executed. This selective activation significantly reduces glitch power, a common source of leakage in busy modules like the ALU. The result is a more power-efficient design that mitigates the risk of side-channel attacks stemming from power analysis.

3. PRNG:

To address the need for fresh randomness, especially for masking techniques, we integrated a PRNG into the processor. This PRNG module generates 32-bit

random numbers using an XORSHIFT-ADD-based algorithm [56], a variant of the Xorshift PRNG. The generated random values are produced on-the-fly and stored directly from the PRNG module to the register file. While the current implementation uses the XORSHIFT-ADD PRNG, future work may explore other PRNG types that could offer even better suitability for our processor.

To sum up, the improvement of SSP's architecture represents a significant advancement in embedded processor design, specifically tailored to meet the growing demands of IoT applications. Its architecture, characterized by a compact ISA and efficient operational handling, directly addresses the critical challenge of enhancing security against power side-channel attacks while simultaneously optimizing for efficiency and compactness. In IoT environments, where devices are often limited in terms of power supply and circuit area, the SSP's architecture offers a significant advantage. Its design focuses on reducing both the power consumption and the circuit footprint, making it an ideal choice for a wide array of IoT applications. From sensor networks, which demand longevity in power-limited scenarios, to smart home appliances requiring compact yet powerful processing capabilities, the SSP fits seamlessly into various IoT ecosystems. The design of the SSP reflects a deep understanding of the needs and challenges inherent in modern IoT applications. By achieving robust security and satisfying PPA requirements of embedded processors, the SSP stands as a testament to the innovative advancements in embedded processor design, catering to the need for secure, efficient, and compact computing solutions in the IoT domain.

Chapter 4

Protection against Power Side-Channel Attacks

This chapter is a modified version of paper "Hardware/Software Cooperative Design against Power Side-channel Attacks on IoT Devices", by the same author, accepted by the IEEE Internet of Things Journal [54].

This chapter is organized as follows. An overview is described in Section 4.1.This is followed by Section 4.2, which delves into the underlying motivation driving this research, providing context and necessities for the implementations of side-channel secure lightweight cryptosystems. Section 4.3 then elucidates the proposed software-hardware cooperative approach, detailing its design and implementation. Section 4.4 is dedicated to presenting the evaluation results, offering a comparative analysis with contemporary state-of-the-art works in the field. Finally, Section 4.5 gives a summary of this chapter.

4.1 Background

As IoT technologies continue to evolve, the protection of confidential data processed on IoT edge devices becomes increasingly critical. The efficiency of cipher algorithms on a variety of IoT devices, particularly those at the lower end, is essential for real-time processing. The hardware and software implementations

of lightweight ciphers represent significant research areas in this context. These lightweight cipher algorithms are designed to be simple, utilizing either basic operations or minimal memory, while their security levels are mathematically proven [57]. Despite this, physical side-channel attacks pose a substantial risk, capable of deciphering secret information from IoT devices. Unlike covert channels in high-end devices, which result from architectural designs like branch speculation and complex memory hierarchies, side-channels in low-end devices emit physical information such as power consumption and electromagnetic waves. This poses a notable risk, as physical access to IoT devices can be easily achieved by hostile third parties, making them vulnerable to such attacks [58].

Masking is recognized as a provably secure method to counteract physical side-channel attacks [59]. It works by dividing secret data into multiple 'shares', introducing fresh randomness for protection. A masking scheme with $d+1$ shares is theoretically secure against d-th order attacks. Masking can be implemented in both software and hardware [60, 61], each with its own set of pros and cons. Software masking is easier to apply compared to hardware masking, which necessitates changes in architecture. However, software masking tends to increase latency as more shares are introduced. Moreover, because hardware resources often need to be shared to minimize circuit area, having only $d+1$ shares might not be adequate to guard against d-th order attacks, necessitating more shares and thereby increasing latency overhead [62, 63]. On the other hand, hardware masking can reduce latency through parallel processing of computations across shares, but it comes with drawbacks like increased circuit area and power consumption. The complexity of the baseline micro-architecture also influences the extent of the increase in circuit footprint in masked implementations [64].

Traditionally, countermeasures against power side-channel attacks have been approached solely from either a software or hardware perspective. However, a combined hardware/software cooperative approach is now anticipated to offer a more

efficient solution in mitigating the drawbacks typically associated with these individual methods. Furthermore, as new threats emerge, it becomes essential to enhance the cipher strength in long-life IoT devices, for example, by increasing key lengths or updating cipher algorithms. Consequently, in the development of cryptosystems for IoT devices, embedded processors are expected to be more advantageous than cipher-dedicated circuits. This preference is due to their flexibility and adaptability in responding to evolving security threats and requirements.

Our research introduces an innovative cryptosystem design that merges the strengths of both hardware and software methodologies, aiming to offer robust resistance against side-channel attacks, with a special emphasis on power side-channel attacks. Focusing on the targeted applications, we achieve an efficient Power, Performance, and Area (PPA) balance while ensuring protection against power side-channel attacks. This is accomplished by integrating masked software implementations with a security-oriented microprocessor. This processor is designed with an instruction set architecture (ISA) and microarchitectural structure that are optimized for lightweight cipher operations, allowing it to safeguard secret data against d-th order attacks using only $d + 1$-masked software, a method that differs from previous strategies outlined in [62]. Unique to our design, the processor avoids the use of a duplicated datapath circuit for masked and unmasked modules (such as the ALU and register file), which is common in most existing hardened processors and cipher-dedicated co-processors [64, 65, 66, 67]. This approach substantially reduces the circuit area and power overheads, facilitating low-power/energy processing not only of ciphers but also for other IoT applications like eHealth monitoring [52, 53]. A recent study proposed a masked hardware design for a bit-serial RISC-V processor to reduce circuit area overhead [68]. However, our hardware/software cooperative approach contrasts with this hardware-specific method, which focuses exclusively on ciphers. The most relevant work to our study is the RISC-V Ibex extension designed to support

masked software [69]. While this is based on a small RISC-V core, our work targets even smaller, more resource-constrained edge devices, for which traditional top-down designs based on existing processors may not be appropriate. By prioritizing targeted applications over a wide range of applications, our work achieves superior results compared to other works, particularly within the realm of targeted applications. Our evaluations on lightweight ciphers, including Chaskey, Simon, and AES, demonstrate our work's effectiveness against state-of-the-art works [69], both in terms of PPA and power side-channel protection.

Our primary contributions are summarized as follows:

- By implementing the hardware/software cooperative approach, we realize a cryptosystem design that simultaneously fulfills good PPA to be suitable for constrained IoT edge devices and resistance to power side-channel attacks.
- From the software perspective, to prevent information leakage that previously masked software suffered due to unintended resource sharing in the underlying hardware architecture, we conduct hardware-aware software optimizations with only minimum shares in a provably secure masking countermeasure.
- From the hardware perspective, unlike previous works that harden existing processors by introducing a duplicated datapath circuit or cipher-dedicated co-processor [64, 65, 66, 67], our work takes a bottom-up approach to define a minimum ISA and its architectural structure. Thus, it can not only achieve good PPA when processing masked software but also can suppress power side-channel leakage while maintaining efficiency for other IoT applications.

4.2 Motivation

With the rise of the IoT era and the escalating necessity to process confidential data, such as personally identifiable information, the importance of lightweight symmetric ciphers has grown. These ciphers are specifically designed for resource-constrained embedded systems and are characterized by their simplicity. They typically consist of a limited number of basic instructions, which are compatible with the ISA of even low-end processors. Additionally, these ciphers are parameterized, with aspects like key length and number of rounds being adjustable to effectively counteract emerging security threats.

Cryptosystems built upon microprocessors are favored over hardwired cipher circuits due to their greater flexibility in updating parameters and cipher algorithms. When developing a microprocessor suitable for resource-constrained devices, a bottom-up approach is essential, particularly in defining the ISA and its architectural structure. As an initial step, we examine various symmetric ciphers that utilize different permutation or substitution designs. These ciphers are chosen as benchmarks for our evaluation because of their distinct operational processes during computation. In future work, we plan to extend our coverage to more lightweight ciphers, including Speck, LED, and Ascon, thereby broadening the scope of our research and evaluation.

- **Chaskey** [70] is an ARX (Addition-Rotation-XOR)-based message authentication code (MAC) algorithm. Chaskey is designed for 32-bit embedded processors. ARX ciphers are composed of three operations: modular addition[1], rotation, and bitwise XOR. The basic computation block of Chaskey is the permutation π as shown in Fig. 4.1(a). A variant, Chaskey-12, which has 12 rounds of permutation, was standardized as a lightweight MAC algorithm by the International Organization for Standardization[71].

[1]Hereafter, we refer to modular addition as "addition" for brevity.

Table 4.1: Details of the Ciphers [54]

Cipher	Block Size	Key Size	Rounds	Masking
Chaskey	128	128	12	Boolean/Arithmetic
Simon	64	128	44	Boolean
AES	128	128	10	Boolean

- **Simon** [72] is a lightweight block cipher developed by the U.S. National Security Agency. A single round of the Simon permutation is illustrated in Fig. 4.1(b). Simon is another type of ARX cipher. In contrast to Chaskey, Simon uses bitwise AND instead of addition. It supports various combinations of block size, key length, and rounds. In this work, Simon64, with a 32-bit processing unit and 128-bit key length, was utilized for implementation and evaluation.

- **Advanced Encryption Standard (AES)** [73]: AES, standardized by the American National Institute of Standards and Technology, is a block cipher with various settings for key length, which defines the number of rounds. Its nonlinear substitution step, the SubBytes or S-box, is crucial. The transformation of input values a to output values b through the S-box is shown in Fig.4.2. Although not considered as a lightweight cipher, AES is used in this study as it is a representative symmetric cipher that employs the S-box and has been extensively evaluated in side-channel attack and countermeasure research. In our evaluation, we focus on the S-box, which is the most critical component in AES, similar to most existing works on side-channel attacks and countermeasures.

Table 4.1 presents the details of the cipher benchmarks used in our evaluation. These ciphers support various block and key lengths, here we selected variants that all have the same key length. While these cipher algorithms are mathematically proven to be secure, they are still vulnerable to side-channel attacks that can extract secret information through physical data emissions, such as power consump-

tion. To counter this, we employ a masking countermeasure that divides secret information into multiple shares based on an XOR-ing scheme. Consequently, in these cipher algorithms, neither the original (unmasked) nor the masked implementation requires complex operations. As summarized in Fig.4.3, both implementations of Chaskey and Simon, when compiled for the RISC-V RV32IMC ISA, use only a few simple types of instructions. The AES S-box utilizes even fewer types of operations by employing a table-based approach, primarily involving load/store operations. This is in contrast to most existing processors, which support a much broader range of instructions (47 to 190, as reported in small-scale embedded processors [53]), suggesting that the majority of these instructions are unnecessary for processing lightweight ciphers. Furthermore, a study [55] identified that even unused hardware resources, such as floating-point arithmetic units (FPU), can become sources of side-channel leakage due to glitch propagation.

Given that the circuit area of microprocessors increases with the complexity of their ISA and functionalities, protecting them against side-channel attacks, primarily by adding a duplicated datapath circuit or a cipher-dedicated co-processor [64, 65, 66, 67] makes it more difficult to develop a cryptosystem that is suitable for constrained devices.

Our work addresses this bottleneck by adopting a bottom-up, hardware/software cooperative approach. We define a minimal ISA and its architectural structure that can efficiently process a masked software implementation of lightweight symmetric ciphers. This approach aims to improve PPA while providing robust side-channel protection by collaboratively addressing power side-channel attacks at both the hardware and software levels.

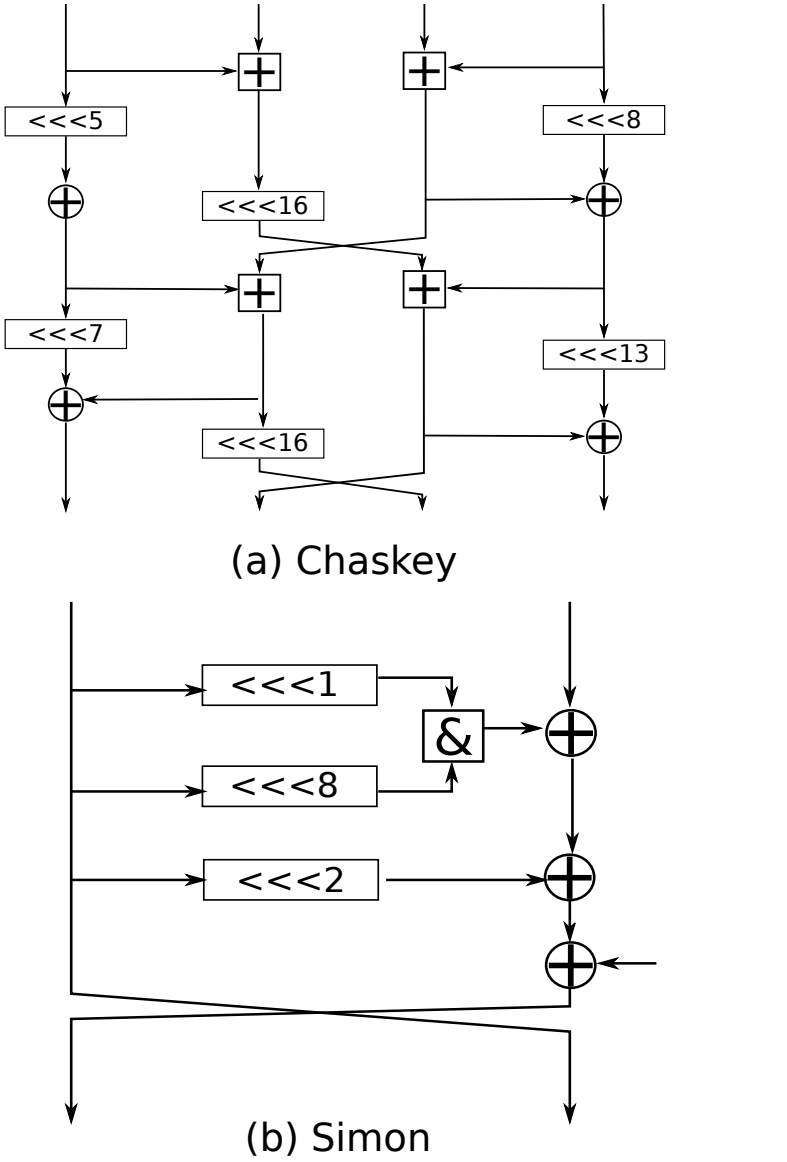


Figure 4.1: One round of permutation of Chaskey and Simon [54]

4.3 Hardware/Software Cooperative Approach against Power Side-channel Attacks

As highlighted in the literature [63], there exists a notable discrepancy between theoretical security proofs and the practical safety of cryptosystem implementa-

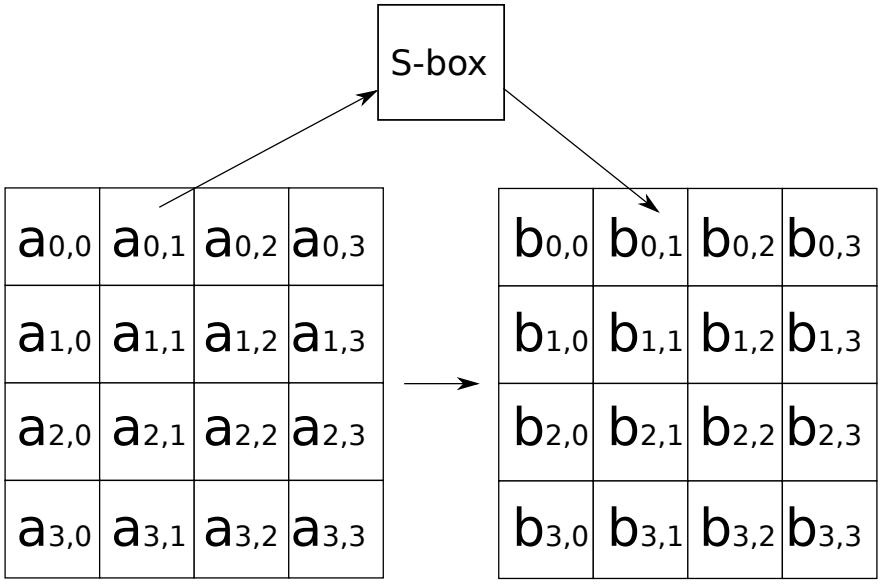


Figure 4.2: AES S-box [54]

tions. This work specifically tackles this issue in constrained IoT devices while focusing on the PPA of these systems. By employing a hardware/software co-operative approach that integrates a security-oriented hardware design with a hardware-aware software implementation, this study aims to develop a cryptosystem that not only resists power side-channel attacks but also minimizes PPA overheads. Note that this work assumes first-order attacks as well as the work [69] and apply two-share masking as a starting point[2]. The hardware aspect of this work follows a bottom-up approach in defining the Instruction Set Architecture (ISA) and architecture of a microprocessor to create tamper-resistant cryptosystems. The software implementation is meticulously designed to prevent unintended data leakage through shared hardware resources, ensuring that a two-share masking scheme is sufficient to protect against first-order attacks.

In the following, we elaborate on the proposed approach relative to hardware and software perspectives.

[2]Resistance against higher order attacks will be explored in future work.

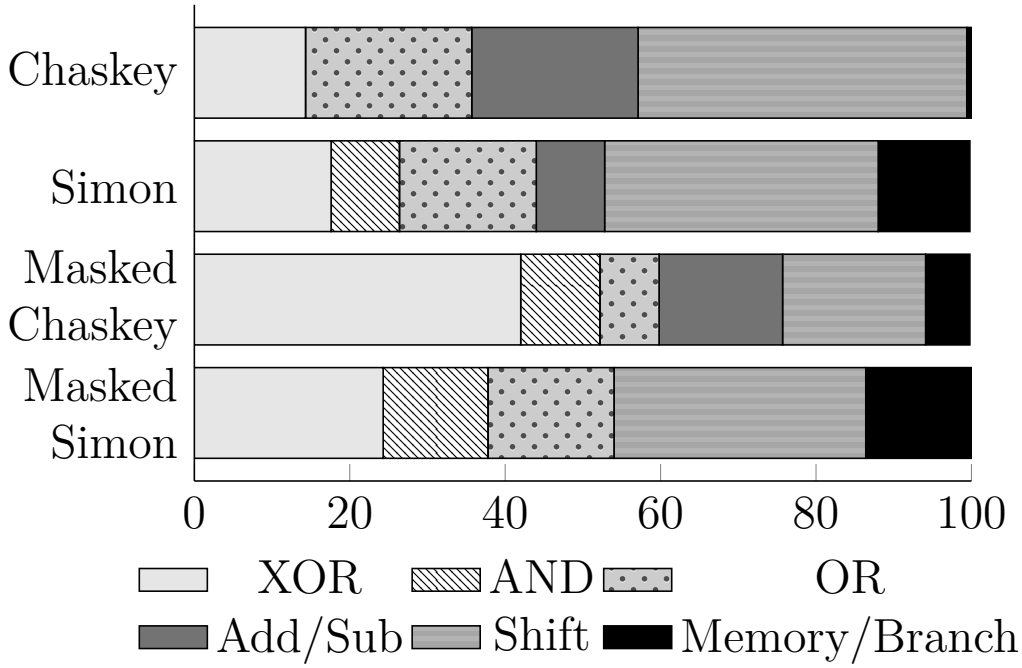


Figure 4.3: Breakdown of operations in unmasked and masked software implementations of representative lightweight ciphers [54]

4.3.1 Security-oriented Hardware Design

To efficiently process masked lightweight ciphers and maintain a requisite level of security within the bounds of limited computing resources, we have developed a uniquely-defined ISA. This ISA is designed to support only a minimal set of necessary instructions and features a simple architecture. This design approach was informed by a analysis on target applications, the details of which are illustrated in Fig. 4.3.

To develop our security-oriented processor, based on our decision on the ISA and microarchitectural source of leakage [55], the following three extensions are applied to the baseline SubRISC+ architecture.The details of the extensions are introduced in Section 3, here we only provide a brief recapitulation.

(1) We introduced *bitwise XOR* and *shift with arbitrary values* into our design,

leading to enhancements in the ALU, instruction decoder, and format to support these instructions.

(2) We employ a gating scheme that disables unnecessary switching on inactive operations in the ALU module. To implement the gated ALU, opcodes are used to determine the current instruction to be executed such that only the parts required for processing its operation are enabled (i.e., low-power input gating [74]).

(3) Finally, as fresh randomness is required for masking (Section 4.3.2), we introduce a pseudo-random number generator (PRNG) module that generates 32-bit random numbers.

4.3.2 Software Implementation: Masking

Masking serves as an effective strategy to safeguard secret data in ciphers from side-channel attacks. This technique involves splitting the secret data into multiple shares, and then performing encryption computations on these shares with added randomness, which helps obscure the analysis of physical characteristics. Since permutation or an S-box are crucial elements in encryption, they require appropriate masking schemes that align with their operational nature. For operations where no carry occurs during computation, such as logic operations, rotation, and table references, Boolean masking is suitable. Conversely, for operations where bit carries might occur based on value, like in addition, arithmetic masking is necessary to ensure security.

In our study, we utilize a conventional masking countermeasure [75] [3] for the software implementations of Chaskey, Simon, and AES. These ciphers were chosen due to their distinct masking properties. Specifically, Chaskey employs both arithmetic and Boolean masking on its permutation, Simon uses only Boolean masking

[3] Other masking countermeasures (e.g., threshold implementation and domain-oriented masking)

on its permutation, and the AES S-box requires Boolean masking for its table-based substitution.

Masked Chaskey Implementation: As discussed in Section 4.2, Chaskey is an ARX cipher where both arithmetic and logic operations are used in the permutation (Fig. 4.1(a)). Thus, Boolean masking is applied to both XOR and rotation, and arithmetic masking is applied to addition. When mixing these masking schemes, the correctness of the final results under masking must be ensured carefully. To realize this, we implement the Goubin conversion [76] between these schemes in two directions (i.e., from Boolean to arithmetic (B2A) and from arithmetic to Boolean (A2B)). The pseudo code for these schemes are described in Algorithms 1 and 2, respectively.

The Bitwise to Arithmetic (B2A) algorithm is implemented as a lightweight solution, involving only a few operations. In contrast, the Arithmetic to Bitwise (A2B) algorithm includes a loop iterated $K-1$ times, where K represents the number of input bits (K=32 in our evaluation). This loop notably becomes a performance bottleneck. Therefore, to effectively mask Chaskey while minimizing latency overhead, it is crucial to strategically insert A2B and B2A conversions at optimal points in the process, along with applying both masking schemes. Fig. 4.4 illustrates our implementation, where each A2B or B2A conversion is marked with an index for easy understanding. In this setup, conversions are placed immediately before and after addition operations. To reduce the latency caused by these conversions, it is possible to bypass "B2A(2)" and "B2A(6)" if the results preceding them are passed directly to the subsequent additions. However, this approach necessitates maintaining both converted and unconverted results, leading to potential register file spilling.

A recent study [77] identified a potential source of leakage in the A2B conversion algorithm, specifically related to the overwriting of the intermediate variable R, as seen in lines 6 and 8 of Algorithm 2. Upon examining the usage of R in the

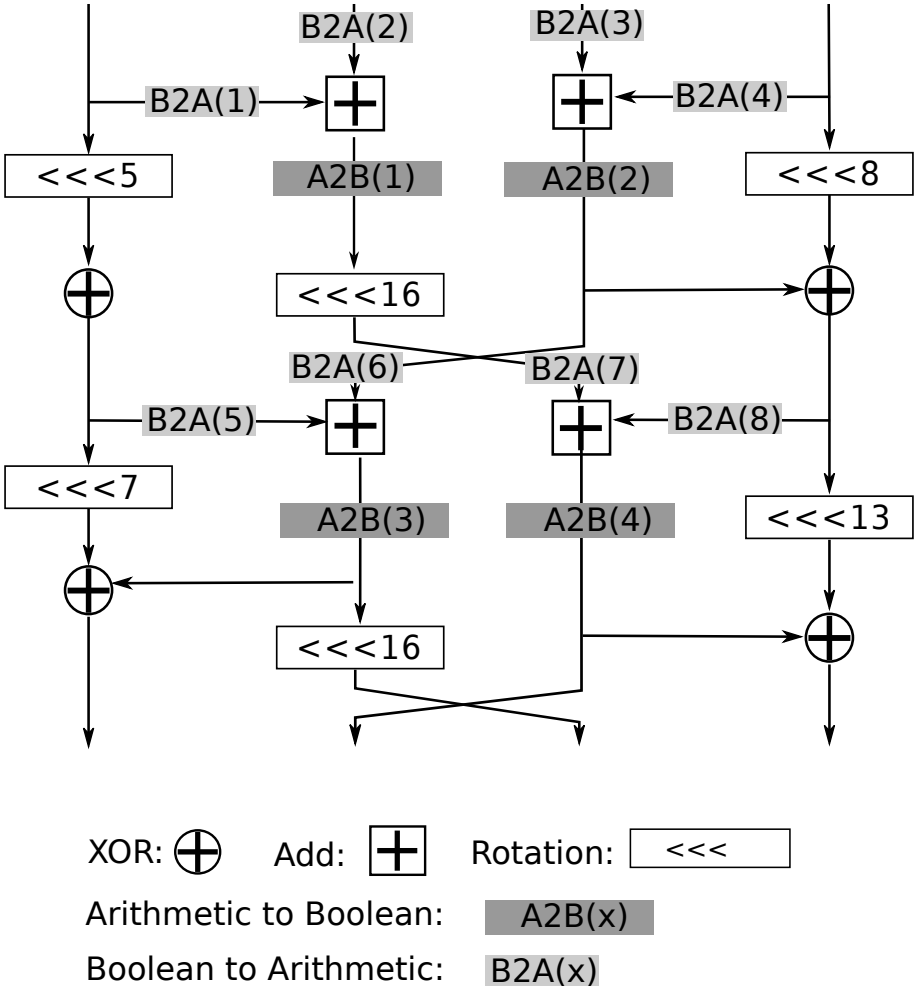


Figure 4.4: Masked Chaskey with A2B and B2A conversions [54]

algorithm, we found that it is not necessary to overwrite R in the same variable (on line 8 of Algorithm 2). To address this potential leakage issue, we modified the algorithm by assigning R to a different variable, thereby eliminating this vulnerability.

Masked Simon Implementation: The Simon permutation, distinct in its operation, does not involve operations with carries. Therefore, it exclusively employs Boolean masking, eliminating the need for any conversions like those required in

Chaskey. An established masking method for the Simon permutation is described in [78] as follows:

$$rout[a] = l[a] \tag{4.1}$$

$$rout[b] = l[b] \tag{4.2}$$

$$\begin{aligned} lout[a] = r[a] \oplus l[a]^2 \oplus (l[a]^1 \wedge l[a]^8) \\ \oplus (l[a]^1 \wedge l[b]^8) \oplus k[a] \end{aligned} \tag{4.3}$$

$$\begin{aligned} lout[b] = r[b] \oplus l[b]^2 \oplus (l[b]^1 \wedge l[b]^8) \\ \oplus (l[b]^1 \wedge l[a]^8) \oplus k[b] \end{aligned} \tag{4.4}$$

where r and l are the inputs on the right and left sides of the round function, $rout$ and $lout$ are the outputs on the right and left sides, and k is a round key. Two shares of each input or key are represented by a and b. The exponent operations mean left rotation with the specified number of bits (e.g., $l[a]^2$ means left rotation by 2 bits). In this masking method, the key point is to handle the AND operation as follows:

$$l^1 \wedge l^8 => (l[a]^1 \oplus l[b]^1) \wedge (l[a]^8 \oplus l[b]^8).$$

By extending the right hand side of this equation, the following format is obtained:

$$(l[a]^1 \wedge l[a]^8) \oplus (l[a]^1 \wedge l[b]^8) \oplus (l[b]^1 \wedge l[b]^8) \oplus (l[b]^1 \wedge l[a]^8)$$

This equation is then divided into two shares in Equations (3) and (4) to ensure correctness under the masking scheme. In addition, before each round, the round key k is preprocessed to be divided into $k[a]$ and $k[b]$ such that $k = k[a] \oplus k[b]$.

Masked AES Implementation: In line with standard practice, we implemented the AES S-box using a table look-up method. For the masking process, we adhered to the principles outlined [79], applying Boolean masking to both the table-based S-box and its indexing. This approach guarantees that during encryption, the

Algorithm 1 Boolean to Arithmetic Conversion (B2A) [54]

Input: (X', r), such that $X' = X \oplus r$, random value R
Output: A, such that $x = A + r$

1: $T <= X' \oplus R$ // T is an intermediate value
2: $T <= T - R$
3: $T <= T \oplus X'$
4: $R <= R \oplus r$
5: $A <= X' \oplus R$
6: $A <= A - R$
7: $A <= A \oplus T$

Algorithm 2 Arithmetic to Boolean Conversion (A2B) [54]

Input: (A, r), such that $X = A + r$, random value R
Output: X', such that $X = X' \oplus r$

1: $T <= 2R$ // T is an intermediate value
2: $X' <= R \oplus r$
3: $O <= R \wedge X'$ // O is an intermediate value
4: $X' <= T \oplus A$
5: $R <= R \oplus X'$
6: $R <= R \wedge r$
7: $O <= O \oplus R$
8: $R <= T \wedge A$
9: $O <= O \oplus R$
10: **for** $k = 1 \ldots K - 1$ **do**
11: $R <= T \wedge r$
12: $R <= R \oplus O$
13: $T <= T \wedge A$
14: $R <= R \oplus T$
15: $T <= 2R$
16: **end for**
17: $X' <= X' \oplus T$

original values are not processed directly. Instead, masked values are processed using a masked index, ensuring enhanced security against side-channel attacks.

4.3.3 Software Implementation: Resource Allocation and Instruction Scheduling

We carefully considered hardware resource allocation and instruction scheduling for the masked software implementations. Theoretically, these implementations, which employ $d + 1$ shares and include conversions between masking schemes, should safeguard secret data from d-th order side-channel attacks. However, as noted in the literature [62, 69], the effectiveness of this protection is contingent on the implementation in the underlying embedded processors. Two critical aspects are: (i) the transition of two shares of the same secret data at the same memory location or entry can lead to leakage. (ii) shares of the same secret data should not be accessed within two successive instructions [69]. Thus, failing to satisfy these two constraints can lead to disclosure of Hamming distance between two shares.

To adhere to these constraints and prevent inadvertent resource sharing of the same secret data, we meticulously crafted the software with consideration of our microprocessor's design (as detailed in Section 4.3.1). This implementation focuses on two key aspects: managing the register file and memory allocation to address constraint (i), and strategically scheduling instructions to satisfy constraint (ii).

To address constraint (i), we allocated dedicated register file entries or memory locations for each share, explicitly preventing transitions between shares. This method, while effective, can be demanding in terms of register usage, especially for embedded processors with limited register file entries (like those with 16 or fewer). If the number of dedicated register file entries isn't sufficient for all intermediate variables, we also allocate dedicated memory locations. When implementing masked software, constraint (i) can be satisfied by specifying these

register/memory allocations in the assembly code explicitly or via proper usage of global variables in the C code.

For constraint (ii), we adjusted instruction scheduling, allowing us to sequence instructions on values that are not related to the same secret data consecutively. Additionally, swapping operands in commutative operations aids in efficient scheduling. While these adjustments are straightforward at the assembly level, when using C code, it's crucial for designers to verify that the compiled code's instruction scheduling aligns with the intended design.

In essence, these hardware-aware optimizations are guided by the principle of avoiding unintentional resource sharing of the same secret data during encryption processes, as emphasized in [62].

4.4 Evaluation

In this section, we first describe our experimental setup. Then, we demonstrate the effectiveness of our work against state-of-the-art works in terms of PPA and security.

4.4.1 Experimental Setup

In our evaluation, the SAKURA-X board, specifically designed for side-channel analysis with dedicated power measurement connectors, was utilized [80]. This board is equipped with two FPGAs: a Xilinx Kintex-7 XC7K160T FPGA and a Xilinx Spartan-6 XC6SLX45 FPGA. The Kintex-7 FPGA is tasked with implementing the cryptosystem, while the Spartan-6 FPGA serves as a controller for the Kintex-7, handling functions such as clock generation.

We set the clock frequency of the Kintex-7 FPGA to 12 MHz for our experiments. To thoroughly evaluate the impact of our method, we implemented the masked

Table 4.2: Comparison of PPA (masked software implementations) [54]

	Performance (cycles)		Time (us)			Dynamic power (mW)			Energy (nJ)			Area		
	Chaskey	Simon	Chaskey	Simon	AES S-box	Chaskey	Simon	AES S-box	Chaskey	Simon	AES S-box	LUTs	FFs	DSPs
Ibex	1,060	46	88.33	3.83	3.33	**15.22**	**9.46**	**10.67**	**1344.38**	**36.23**	**35.53**	2,612	926	**1**
coco-Ibex [69]	1,060	46	88.33	3.83	3.33	13.24	8.35	10.28	1169.49	31.98	34.23	**3,758**	**1,946**	**1**
SubRISC+ [52]	**3,619**	**126**	**301.58**	**10.50**	**4.08**	1.11	0.88	0.95	334.75	9.24	3.88	832	626	0
SSP	1,240	47	103.33	3.92	3.75	1.21	0.96	1.02	125.03	3.76	3.83	856	628	0

Table 4.3: Evaluation on Non-cipher Applications [54]

	Performance (cycles)		Time (us)		Dynamic Power (mW)		Energy (nJ)	
	Ibex	SSP	Ibex	SSP	Ibex	SSP	Ibex	SSP
Sort	**912,889**	588,241	**76,074.08**	49,020.08	**10.40**	4.09	**791,170.43**	200,492.13
Motion	**85,466**	72,006	**7,122.17**	6,000.50	**10.27**	4.84	**73,144.69**	29,042.42
Edge	148,730	**212,919**	12,394.17	**17,743.25**	**11.36**	4.00	**140,797.77**	70,973.00
Histogram	27,348	**30,987**	2,279.00	**2,582.25**	**9.78**	4.60	**22,288.62**	11,878.35
DTW	**716,318**	615,163	**59,693.17**	51,263.58	**9.91**	4.11	**591,559.31**	210,693.31

Table 4.4: Details of the Non-cipher Applications [54]

Applications	Description
Sort	Sort a set of integers using quick sort algorithm.
Motion	Detect unmatched blocks from two images.
Edge	Apply a Laplacian filter to a image.
Histogram	Construct a histogram of pixels from a image
DTW	Execute the Dynamic Time Wraping (DTW) algorithm.

versions of Chaskey, Simon, and the AES S-box on the Kintex-7 FPGA, using various processors.

- **Ibex**[4]: The Ibex is one of the smallest RISC-V cores whose ISA is defined as RV32IMC. The number of register file entries is 32.
- **SubRISC+**: The SubRISC+ is one of the smallest microprocessors based on which our processor was newly developed. In addition, its ISA was uniquely defined for small-scale IoT devices [52]. The number of register file entries is 16.
- **coco-Ibex**: coco-Ibex [69], is the most relevant work to our study in that it was developed by partially refining Ibex to be aware of masked software implementations. A verification tool was used to identify side-channel leakage sources in the Ibex netlist. Then, to mitigate secret-dependent glitches, a gating scheme was applied to modules that had leakage (e.g., register file, ALU, and load/store unit). Because we found some critical bugs in its

[4]github.com/lowRISC/ibex

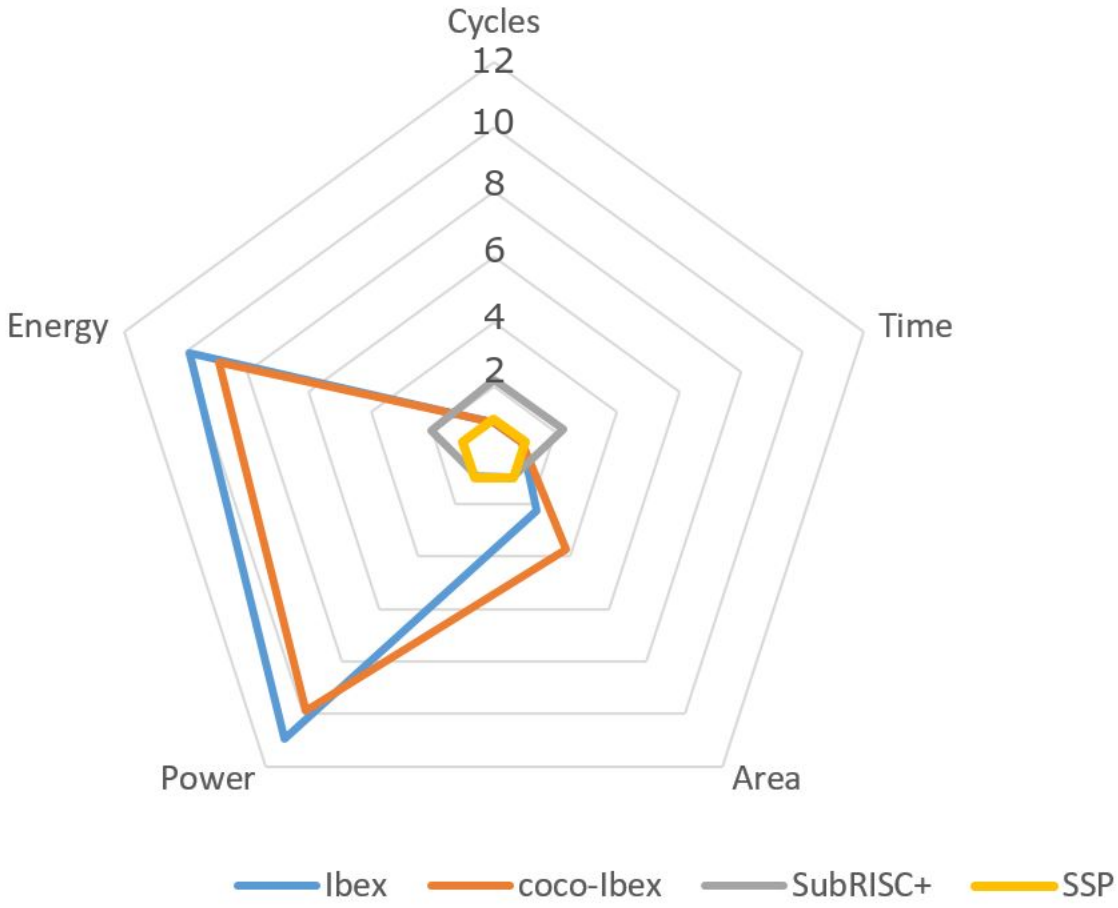


Figure 4.5: Summary of Evaluation on Cipher Applications

open-sourced RTL code from which the bitstream could not be generated, we reproduced coco-Ibex from the original Ibex code.

- **SSP**: This is the proposed processor (Section 4.3.1). In the following, we refer to our processor as the small and secure processor (SSP).

For the assessment of the implemented processors, we focused on their PPA characteristics. Using Xilinx Vivado 2020.2, we evaluated the circuit area (resource utilization) and power consumption, along with the FPGA implementation. The performance, specifically the cycle counts per round for the Chaskey/Simon permutation and the AES S-box, was using a cycle-accurate simulator tailored for

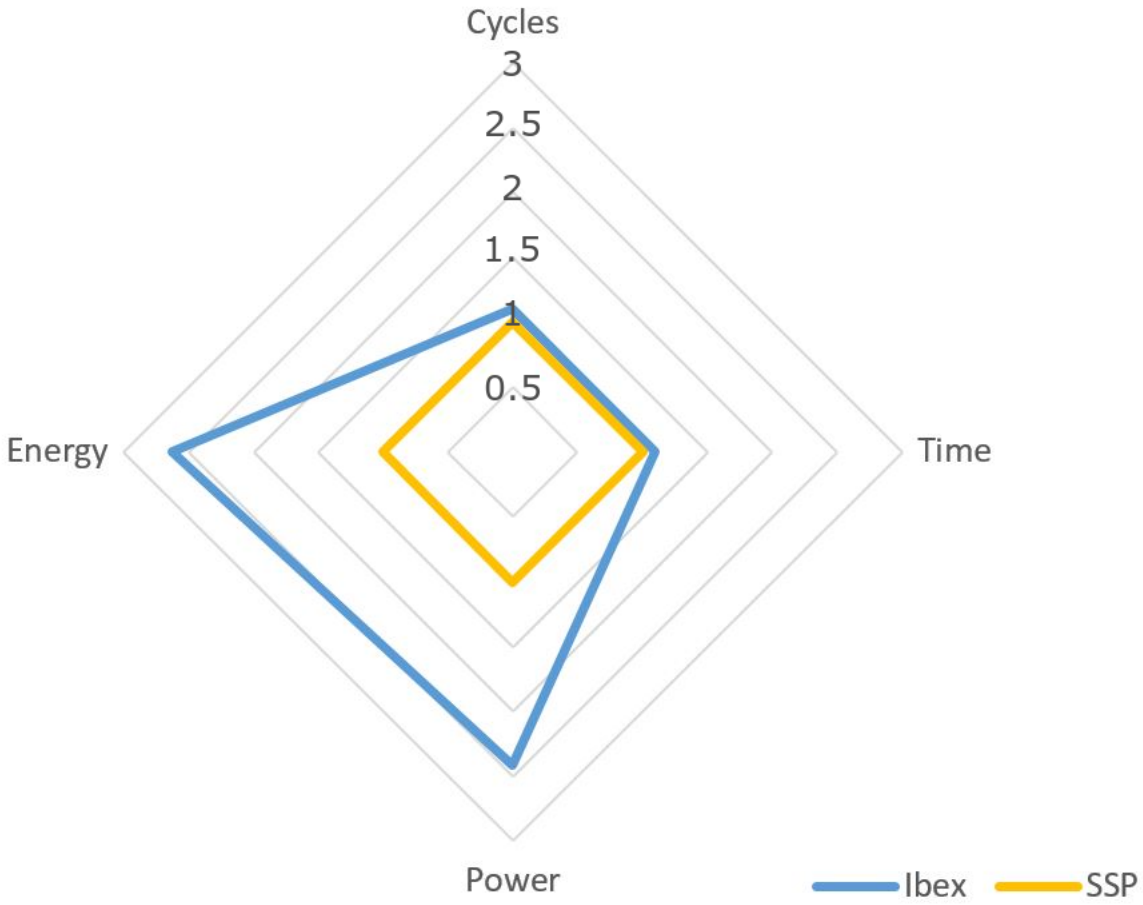


Figure 4.6: Summary of Evaluation on Non-cipher Applications

each processor. For software compilation and assembly, we utilized an in-house toolchain for the Secure Side-channel-aware Processor (SSP) and open-source toolchains for the other processors under comparison. The results concerning execution time were obtained at a clock frequency of 12 MHz, which was consistently used in all subsequent simulations and evaluations. This uniform frequency setting ensures comparability and consistency across the different tests and evaluations conducted.

For the security aspect, both coco-Ibex and SSP were evaluated for their resistance to first-order power side-channel attacks. In this security evaluation, we employed

Welch's t-test [81] to assess the presence of first-order side-channel leakage in the measured power traces. The data acquisition was performed using a Keysight MSOX3104T oscilloscope. We conducted a non-specific leakage assessment, feeding both random and fixed plaintexts to the processors. The aim was to compute t-values and determine if there was a significant difference between these two sets of data. If the peak t-value exceeded the threshold of |4.5|, the null hypothesis would be rejected with a confidence level exceeding 99.999%. This implies that the power traces for random and fixed plaintexts are statistically distinguishable, potentially revealing secret information. To gain a comprehensive understanding of the security of our design, we performed the security evaluation at both the instruction and cipher levels. For these assessments, the oscilloscope's sampling rates were set at 5 GSa/s and 313 MSa/s, respectively.

4.4.2 Results

PPA Evaluation: In Table 4.2, the PPA of four processors is compared. In the table, bold text indicates the worst result for each metric to compare. The overheads of other processors compared to SSP are summarized in Figure 4.5. The results of SSP are set to one and results of other processors are normalized accordingly and then averaged for each metric. To ensure fairness, data/instruction memory and the PRNG module were excluded from all processors. Pseudo-random number are generated in advance and stored in the memory. The cycles for pseudo-random number generation are not counted in the performance results. Cycle count results show that SSP, despite supporting fewer instructions than Ibex and coco-Ibex, achieves comparable cycle counts and execution times across all ciphers. This efficiency is attributed to SSP's security-oriented ISA, especially its bitwise XOR and shift operations. SubRISC+ and SSP share similar ISAs, indicating that SSP's design specifically improves cipher processing. Moreover, as

discussed in Section 4.2, many instruction types in Ibex and coco-Ibex are unused, despite their relatively compact RISC-V ISAs.

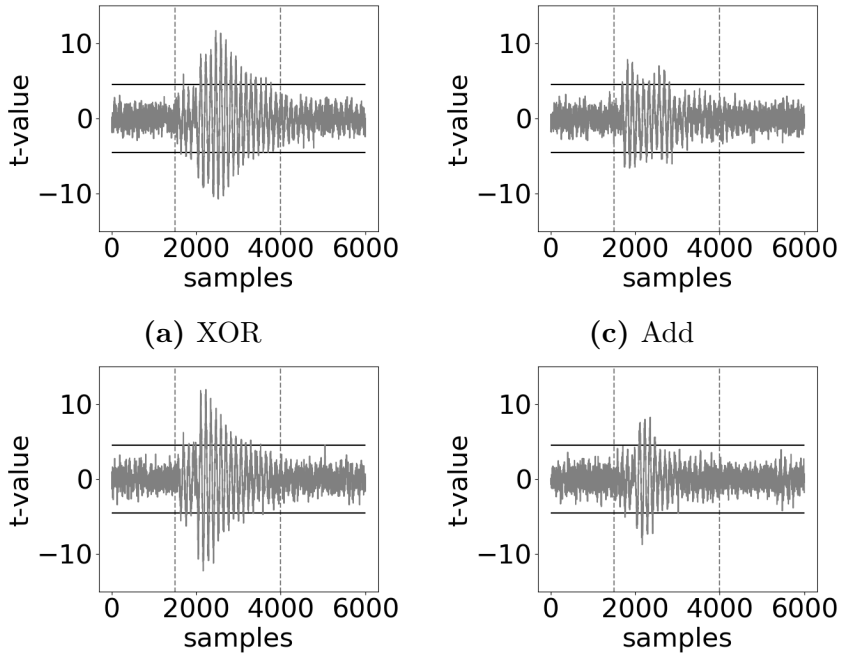
In our comparison of power consumption and circuit area, we observed that Ibex has a larger circuit area due to its support for a broader range of instructions and functionalities. Coco-Ibex, despite its modifications for side-channel leakage mitigation, incurred a noticeable area overhead. However, interestingly, coco-Ibex showed lower power consumption than Ibex, attributed to its gating scheme. SubRISC+ and SSP, with their simpler microarchitectural structures, achieved significantly less power consumption and area compared to Ibex and coco-Ibex. SSP showed a slight increase in both metrics compared to SubRISC+, but this was offset by a greater reduction in cycle counts, especially beneficial for Chaskey and Simon due to their frequent XOR and rotation operations. The cycle count reduction in SSP, leading to reduced execution times, contributed to lower overall energy consumption, outperforming both the RISC-V processors and SubRISC+. Although SSP has a slightly higher overhead in LUTs (less than 2.9%) compared to SubRISC+, its efficiency in processing other applications, like eHealth [53], remains high. The cycle count reduction was notably significant for Chaskey and Simon, which involve numerous XOR and rotation operations. These operations are efficiently executed by the ISA of SSP. Due to this reduced cycle count, leading to shorter execution times, SSP's energy consumption was lower than both the RISC-V processors and SubRISC+. In summary, SSP demonstrated superior PPA performance compared to other state-of-the-art processors across various cipher algorithms.

In this work, while our primary focus was on cipher applications and side-channel protection, we also recognized the importance of considering the performance of non-cipher applications. To understand the PPA differences between the proposed SSP, with its compact ISA, and the RISC-V Ibex, which features a more commonly used embedded ISA, we conducted evaluations. These assessments

aimed to verify potential performance losses in non-cipher applications due to SSP's compact ISA. We evaluated a set of typical applications performed on IoT edge devices for data processing and analysis. The comparative analysis of SSP and RISC-V Ibex on these applications is presented in Table 4.3. In the table, bold text indicates the worse result for each metric to compare. These non-cipher applications are briefly outlined in Table 4.4. The overheads of Ibex compared to SSP are summarized in Figure 4.6. The results of SSP are set to one and results of Ibex are normalized accordingly and then averaged for each metric. The results showed that SSP and Ibex have comparable cycle counts and execution times. Surprisingly, despite its more compact ISA, SSP outperformed Ibex in several applications. This performance advantage is attributed to the fact that memory and branch instructions, which take multiple cycles on Ibex, reduced its relative performance compared to SSP. In applications like Edge and Histogram, where there are fewer multiple cycle instructions, Ibex demonstrated better performance. In terms of power consumption, SSP was found to be more power-efficient due to its architectural design. This efficiency led to better energy performance for SSP, even in scenarios where it required a longer execution time. The area results were consistent with those in Table 4.2, as there were no changes in the circuit design. Overall, the findings indicated that the proposed SSP achieved superior PPA compared to state-of-the-art processors in non-cipher IoT applications.

Security Evaluation at the Instruction Level:

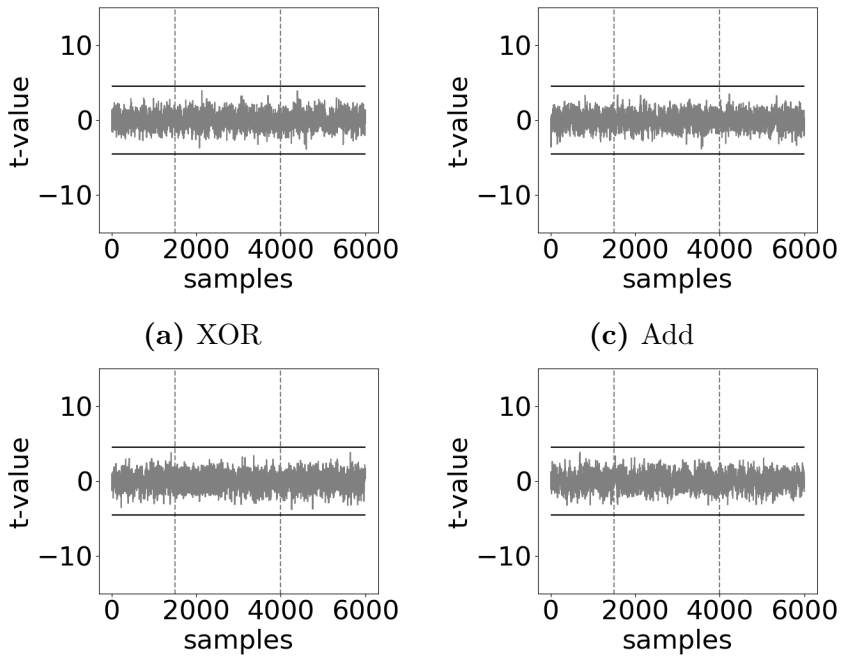
To thoroughly investigate potential information leakage with precise detail, we conducted instruction-level masking evaluations on the proposed SSP. These evaluations compared the performance of unmasked and masked instructions, with the results depicted in Fig.4.7 for unmasked instructions and Fig.4.8 for masked ones. In both figures, auxiliary horizontal lines are drawn at the |4.5| level to aid in analysis. We focused on the four most critical operations used in our benchmarks: XOR, AND, addition, and shift operations. These were evaluated using the non-



(a) XOR **(c)** Add

(b) AND **(d)** Shift

Figure 4.7: Unmasked instructions on SSP



(a) XOR **(c)** Add

(b) AND **(d)** Shift

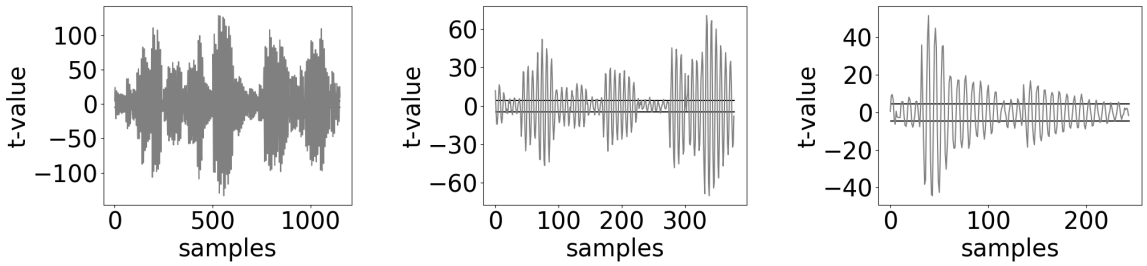
Figure 4.8: Masked instructions on SSP

specific t-test, employing 50,000 traces for both random and fixed plaintexts. In the software, these operations were masked using a two-share masking scheme. To minimize interference from other instructions, NOP (No Operation) instructions were inserted before and after the targeted instruction(s). The evaluation period for the targeted instruction(s) is indicated by the region between vertical dotted lines in the figures. The results demonstrate that with the application of two-share masking, SSP is effectively able to protect secret information against first-order attacks, as evidenced by the performance within the defined evaluation period.

Security Evaluation at the Cipher Level: Finally, we conducted security evaluations at the cipher level. This involved implementing both unmasked and masked versions of ciphers on the proposed SSP and assessing them using a non-specific t-test. The results for the unmasked and masked ciphers on SSP are shown in Figs.4.9 and 4.10, respectively. Additionally, we evaluated the masked ciphers on coco-Ibex, as depicted in Fig.4.11. For the masked ciphers on both coco-Ibex and SSP, we applied the two-share masking along with the optimizations discussed in Sections 4.3.2 and 4.3.3. To thoroughly assess the impact on security, we used the largest number of traces for evaluations involving the combination of masked ciphers and SSP (as shown in Fig. 4.10). This approach allowed us to conduct an in-depth examination of how these implementations influence security, particularly in terms of resistance to side-channel attacks.

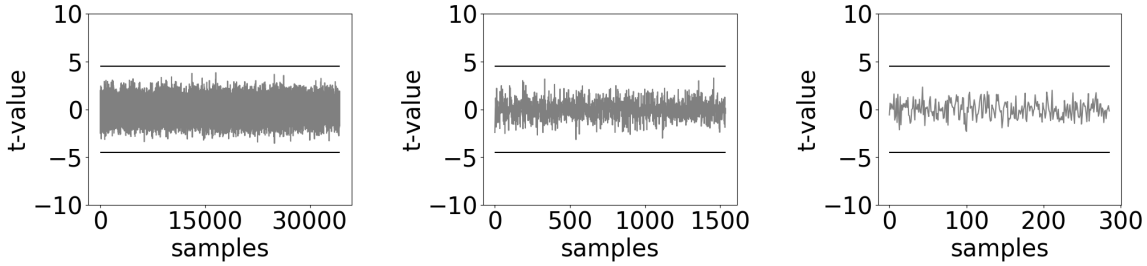
The unmasked instruction results in Fig.4.7 align with expectations, revealing that t-values exceed the |4.5| threshold across almost all cipher operations in Fig.4.9. This pronounced leakage, more severe than in the instruction-level evaluations, underscores the necessity for a hardware-aware software implementation to mitigate secret leakage. Regarding coco-Ibex, as shown in Fig.4.11, even though the

[6]The reason for much fewer samples than the samples in Fig. 4.10(a) is because only a part of the results is presented here to clearly show potential leakage on coco-Ibex. As shown in Table 4.2, the total cycle counts are comparable between coco-Ibex and SSP.



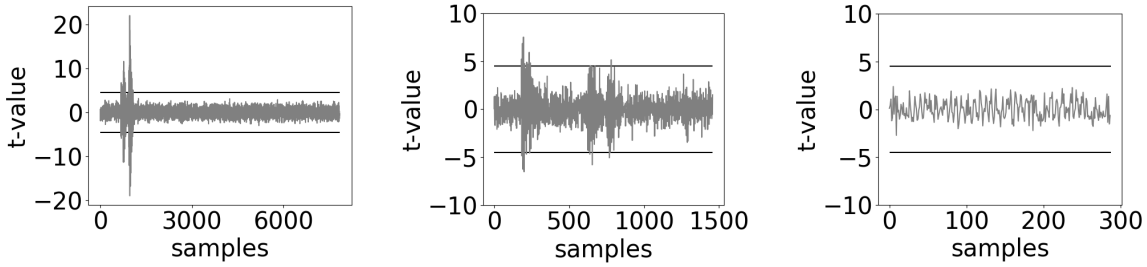
(a) Chaskey **(b)** Simon **(c)** AES

Figure 4.9: Unmasked ciphers on SSP (20,000 traces)



(a) Chaskey **(b)** Simon **(c)** AES

Figure 4.10: Masked ciphers on SSP (500,000 traces)



(a) Chaskey[6] **(b)** Simon **(c)** AES

Figure 4.11: Masked ciphers on coco-Ibex (100,000 traces)

ciphers were masked, it successfully suppressed t-values below |4.5| only for the AES S-box. This is in line with findings reported in the work [69]. However, coco-Ibex did not achieve the same level of success for Chaskey and Simon. In the cited work, the authors used an ASIC verification tool to identify leakage sources before evaluating coco-Ibex on an FPGA. It's important to note that behaviors on ASIC and FPGA are not identical, and this disparity might prevent the identifica-

tion of all leakage sources in the FPGA implementation. Consequently, coco-Ibex still exhibits leakage in Chaskey and Simon. Addressing these additional leakage sources in coco-Ibex would likely lead to further PPA overhead.

For the masked ciphers, the proposed SSP effectively managed to keep the t-values within the |4.5| threshold for all evaluated ciphers. This performance indicates that SSP successfully eliminates leakage sources across different ciphers, benefiting from its security-oriented bottom-up microprocessor design. This achievement is particularly noteworthy as it was accomplished without resorting to expensive strategies such as datapath duplication [64, 65, 66] or the specific refinements implemented in coco-Ibex [69].

In conclusion, when considering the results for both unmasked and masked cipher versions, as well as the PPA outcomes, SSP demonstrates an effective balance between PPA efficiency and side-channel protection. This balance is achieved through a cooperative approach, combining a hardware perspective (with the security-oriented processor design) and a software perspective (through microarchitecture-aware optimizations). This integrated strategy effectively addresses the challenges of implementing secure and efficient cryptosystems for resource-constrained environments.

4.5 Summary

In our study, we presented a hardware/software cooperative design tailored for cryptosystems that are resistant to power side-channel attacks and optimized for resource-constrained IoT devices. Adopting a bottom-up approach, we developed a security-oriented microprocessor and implemented masked software, incorporating hardware-aware optimizations to enhance security and efficiency. Through comparative evaluations against current low-power and security-aware processors,

our work demonstrated superior PPA efficiency, along with robust security against first-order power side-channel attacks, across various types of lightweight ciphers.

Chapter 5

Implementation of eHealth Applications

This chapter is a modified version of paper "Implementation of Lightweight eHealth Applications on a Low-Power Embedded Processor"[53], by the same author, published in the Proceedings of Institute of Electrical and Electronics Engineers (IEEE Access), Copyright© 2020 IEEE Access.

This chapter is organized as follows. An overview is described in Section 5.1. Detailed descriptions of the target eHealth applications are provided in Section 5.2. This chapter then progresses to articulate the intricacies of the DTW algorithm, with the initial focus on the naive implementation as expounded in Section 5.3, followed by the refined, optimized version of the algorithm discussed, and an in-depth exploration of the software development flow. Section 5.4 present the evaluation results and the comparisons with the state-of-the art works. Finally, Section 5.5 gives a summary of this chapter.

5.1 Background

Advances in Internet of Things (IoT) technologies have highlighted the value of IoT devices particularly emphasizing their utility in healthcare systems. Utilizing information and communication technologies in the healthcare sector, specific-

ally eHealth, enables the development of various monitoring applications. These applications leverage real-time data, including accelerometer readings and biomedical signals such as electrocardiogram (ECG), electroencephalogram (EEG), and blood pressure, which are essential in facilitating diagnoses and decision-making processes [4]. Due to the on-site necessity of monitoring tasks, eHealth devices are predominantly battery-powered embedded systems. The challenge lies in processing applications in real-time on small, low-power IoT devices. The IoT eHealth devices comprise sensing (hereafter, sensor), communication, and processing (hereafter, hardware architecture) subsystems. On one hand, eHealth systems are increasingly incorporating a range of small-scale, low-power sensors, such as microelectromechanical (MEMS) technologies [82, 83]. These systems are supported by toolsets for gathering and analyzing motion data from these sensors [84]. Additionally, the implementation of energy-efficient communication technologies, like wireless sensors and body sensor networks, is gaining traction [85]. On the other hand, despite the preference for general microprocessors in various IoT applications due to design productivity, their high power consumption renders them less suitable for IoT-based eHealth devices [86]. To adhere to the *always-on* needs of eHealth applications, a significant reduction in power consumption is imperative [87]. Furthermore, considering the challenges of real-time processing on small, resource-limited processors, meticulous design and optimization are crucial for processor design/selection and the implementation of target application software.

To address the issues previously mentioned, this study introduces a practical implementation of eHealth applications, specifically monitoring tasks, designed for deployment on low-power IoT devices with significant resource limitations. To resolve the challenge of maintaining constant system operation while ensuring power efficiency, this work builds upon and enhances our prior development, SubRISC+ [52], a compact and energy-efficient embedded processor. This enhancement involves a hardware-level approach and a software-level strategy that incor-

porates dynamic time warping (DTW), a scalable, lightweight algorithm widely used in various eHealth monitoring and detection applications. Additionally, we have optimized our software by reducing memory usage. Our evaluations highlight the efficiency of our hardware design in terms of both circuit area and power consumption, as well as the scalability of our software implementation.

The contributions of this work are summarized as follows:

- The implementation of realistic eHealth systems on both the hardware and software levels for IoT-based portable devices: On the hardware level, we defined the appropriate memory size for the proposed low-power embedded processor to further improve its power efficiency and implemented it using TSMC 65nm LP technology. On the software level, we applied a memory-oriented optimization to the DTW algorithm that is lightweight and can be deployed within a limited memory size. It is worth mentioning that although existing works (e.g., [88, 89, 90, 91]) have studied DTW-based eHealth implementation at the software level, our work holistically considers both hardware and software optimizations for effectively reducing power consumption.
- Evaluations that demonstrate the effectiveness of our work on both the hardware level (i.e., against state-of-the-art processors in terms of circuit area and power consumption) and the software level (i.e., in terms of scalability by varying the data amount used for various applications): Overall, these hardware and software approaches can cooperatively realize power-efficient eHealth systems.

5.2 eHealth Monitoring and Detection Applications

To aid the healthcare system with IoT edge devices, eHealth monitoring and detection applications need to be implemented on wearable devices. To realize power-efficient and constantly running systems for eHealth monitoring applications, it is essential to focus on lightweight algorithms that do not require heavy arithmetic operations [4]. Importantly, lightweight algorithms are not inferior to more computationally intensive algorithms in terms of detection accuracy. The DTW algorithm outperformed computationally heavy algorithms such as an inertial navigation algorithm, a K-nearest neighbor classifier, and a support vector machine (SVM) as shown by a recent work [92].

Table 5.1: Various eHealth applications realized by DTW [53]

Ref.	[88]	[91]	[89]	[90]	[92]
Application	Epileptic seizure	Cardiac arrhythmia	Activity recognition	Parkinson's disease	Alzheimer's disease
Sensor Type	Accel.	ECG	Accel.	Gyroscope	Force sensitive
Sampling Rate	100Hz	0.5-40Hz	100Hz	50Hz	N/A
Data Amount	128	256	100	200	N/A

Based on our literature review, in Table 5.1, we summarize the features of five different eHealth monitoring applications: epileptic seizure detection, cardiac arrhythmia detection, activity recognition, Parkinson's disease and Alzheimer's disease. In these eHealth applications, it is demanded to suppress power consumption in processing as long as the computation is done earlier than the input sampling of each application. Sampling rate and data amount of the first four applications are summarized in Table 5.1 but in the fifth one, details were not clearly given

in the literature. Descriptions of the various eHealth monitoring applications are given as follows:

Epilepsy is a neurological disorder that affects the nervous system. Developing a portable and low-power monitoring system can help many epilepsy patients. Detecting an epileptic seizure is realized through as time-domain or frequency-domain processing using DTW, singular value decomposition, and principal component analysis [93] algorithms by analysing electroencephalogram (EEG) signals, video data, and accelerometer output. As an example, differences between signals in normal and seizure periods can be used for time-domain detection.

Cardiac arrhythmia is a class of irregular heart behaviors. Cardiac arrhythmia is reflected in heart rate or rhythm. Normally, the cardiac conduction system transmits signals to the heart muscle to make the heart operate. However, cardiac arrhythmia occurs if the cardiac conduction system has a dysfunction. Cardiac arrhythmia diagnoses are commonly accomplished using electrocardiogram (ECG) data. The DTW-based diagnoses achieve less computational complexity and memory usage [94] than those of other detection algorithms such as artificial neural networks, linear discriminant analysis, and Bayesian classifiers. Thus, detection is expected to be performed on mobile monitoring devices [91].

Activity recognition from portable monitoring devices can be used for detecting accidents and diseases especially in the elderly. By processing accelerometers and gyroscopes data as acceleration and rotation angles on different axes, recognition on current movement status can be achieved. Real-time detection of daily activities using a dedicated hardware [95] or off-the-shelf microcontroller [89] was done by using DTW-based activity recognition.

Parkinson's disease is a complex nervous system disorder. Parkinson's disease has distinct clinical features with neuropsychiatric and non-motor manifestations. The cause of Parkinson's disease is not clear but the patient's environment and

genetic factors are thought to affect the disease. Becasue Parkinson's disease is a progressive disease with significant effects on daily life, it must be monitored constantly. Data from a gyroscope can be used for long-term monitoring because symptoms may appear when walking and during daily activities,

Alzheimer's disease is one kind of brain disorders that could cause memory and behavior problems. Although the causes of the Alzheimer's disease are not clearly determined yet, it is considered to be related to the effects of the aging and genetic factors on the brain. As this is a progressive disease with large effects on the daily life, it needs to be monitored constantly. Since the symptoms may appear in the walking, the force sensitive resistors can be used to detect the diagnostic [92].

Aforementioned eHealth applications can all be effectively realized with the DTW algorithm. Figure 5.1 gives some biomedical signals from different datasets that have been used in DTW-based detection schemes. Although some input signals are provided with multiple dimensions, our implementation utilizes only one dimension as research [96] has shown that reducing the number of utilized dimensions for detection using DTW causes negligible or no degradation in detection accuracy.

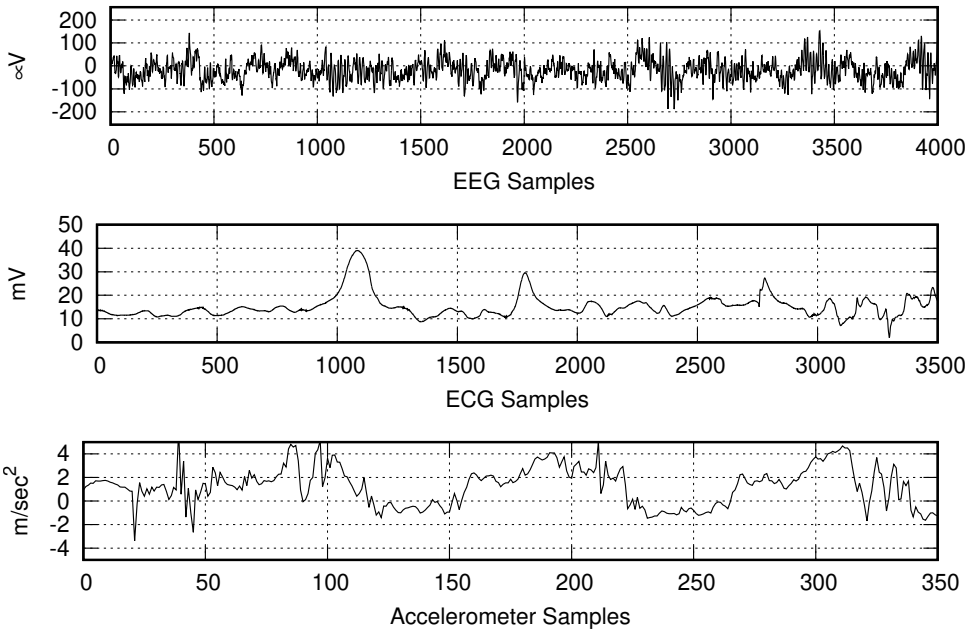


Figure 5.1: Bio-medical signals [53]

5.3 Detection Algorithm and Development Flow

5.3.1 Dynamic Time Warping

Dynamic Time Warping (DTW) is an algorithm proposed to effectively detecting patterns in data streams or time series [97]. The DTW algorithm has wide applications such as eHealth, financial analysis and network monitoring [98]. As a lightweight algorithm that computes the minimum distance between two data sequences in the time domain, DTW is effective for detecting the targets of the individual eHealth applications listed in Table 5.1.

Compared to lightweight versions of various algorithms including reinforcement learning [85] and SVM [99], DTW was selected because it can be efficiently implemented without using power-hungry hardware resources such as multiplier, divider, and floating-point arithmetic units. In addition, DTW can be easily ex-

tended to handle the types of streaming data. In our applications, different types of data are generally obtained by various sensors in a steam manner.

Each eHealth application takes two signals as input. DTW compares a time-series sampled data sequence with a stored data pattern sequence for reference. Hereafter, we refer the sequences of sampled input data and reference data as "sample" and "pattern", respectively. A minimum distance is then computed between the sample and pattern. Target detection is accomplished if the computed minimum distance is below a defined threshold.

A naive implementation of the DTW algorithm is described in Algorithm 3. Firstly, a two-dimensional array dp of size $n \times m$ is initialized. In this two-dimensional array, n and m represent the sample amount and the pattern amount, respectively (lines 1 to 8). Secondly, the distance dp is computed from the cost c and the minimum value for each of the array's elements (lines 10 to 16). Lastly, the distance between the sample and the pattern is stored in the last element of array dp (i.e., $dp[n][m]$).

Algorithm 3 Dynamic Time Warping (DTW) [53]

Input: $sample[1 \ldots n], pattern[1 \ldots m]$

Output: $dp[n][m]$

```
1: dp = [0 ... n][0 ... m]
2: // set a 2D array for the distance
3: for i = 0 ... n do
4:     for j = 0 ... m do
5:         // initialize the array
6:         dp[i][j] = ∞
7:     end for
8: end for
9: dp[0][0] = 0
10: for i = 1 ... n do
11:     for j = 1 ... m do
12:         // compute the cost and find the minimum
13:         c = distance(sample[i], pattern[j])
14:         dp[i][j] = c + min(dp[i − 1][j], dp[i][j − 1], dp[i − 1][j − 1])
15:     end for
16: end for
```

An example of DTW processing is illustrated in Figure 5.2. The left and bottom graphs show the pattern and sample data sequences, respectively. Based on pattern and sample data, the minimum warping path is computed as shown in the middle figure. The x and y-axes of the middle figure represent time steps and data value, respectively. The gray cells in the middle figure represent the accumulated distance computed between the array *sample* and the array *pattern* for each index. That is to say, grids in the middle figure represents the two-dimensional array *dp* shown in Algorithm 3. The minimum cost is dynamically computed on the basis

of three neighboring cells. Thus, a value in the highlighted box is calculated from the three cells indicated by three arrows in the middle figure.

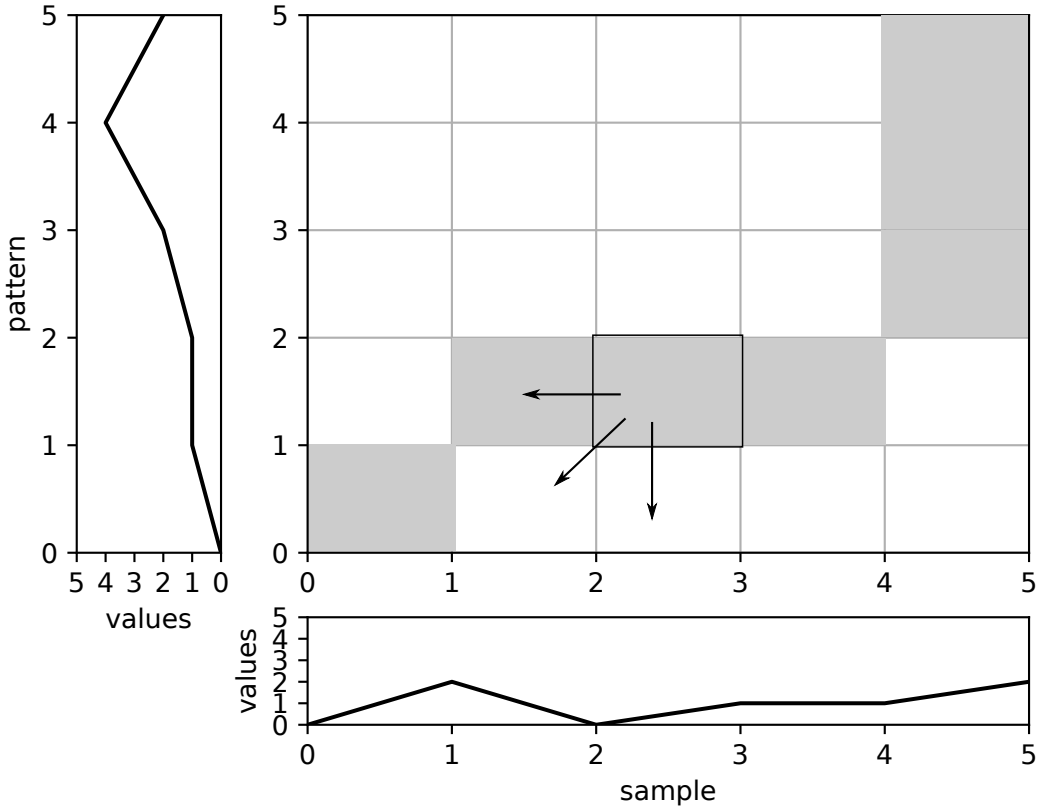


Figure 5.2: An example of dynamic time warping processing [53]

5.3.2 Optimized Dynamic Time Warping

The aforementioned naive DTW algorithm shown in Algorithm 3 has a critical problem of using a large two two-dimensional array. The memory space used by the two two-dimensional array has a size of $O(mn)$ and increases as m and n increase. In most of our eHealth use cases, m equals to n. For example, if 32-bit values are used for processing, a case where the pattern and sample each have 128 data points requires memory of $128 \times 128 \times 4\ B = 64\ KB$. The required memory

size of the naive DTW is far larger than the constraints in the resource-limited hardware architecture.

To overcome the memory constraint issue, we adopted a memory-conscious DTW algorithm [98]. This optimized DTW algorithm effectively reduces the space complexity from $O(mn)$ to $O(m)$. Additional modifications were applied to efficiently handle the target applications. The modified algorithm is shown in Algorithm 4.

Algorithm 4 Memory-Conscious Dynamic Time Warping [53]

Input: $sample[1 \ldots n], pattern[1 \ldots m]$

Output: $dp[m]$

1: $d = [0 \ldots m]$ // set a 1D array for the temporal distance
2: $dp = [0 \ldots m]$ // set a 1D array for the distance
3: **for** $i = 0 \ldots m$ **do**
4: // initialize the arrays
5: $d[i] = \infty$
6: $dp[i] = \infty$
7: **end for**
8: $d[0] = 0$
9: **for** $i = 1 \ldots n$ **do**
10: **for** $j = 1 \ldots m$ **do**
11: // compute the cost and find the minimum
12: $c = distance(sample[i], pattern[j])$
13: $d[j] = c + min(d[j-1], dp[j], dp[j-1])$
14: **end for**
15: **for** $ii = 1 \ldots m$ **do**
16: $dp[ii] = d[ii]$ // substitute the array d
17: **end for**
18: **for** $jj = 0 \ldots m$ **do**
19: $d[jj] = \infty$ // reset the array d
20: **end for**
21: **end for**

Firstly, two one-dimensional arrays d and dp of size m are initialized (lines 1 to 7). Secondly, the distance d is computed from the cost c and the minimum value for each of the array's elements (lines 9 to 14). Thirdly, array dp is substituted for array d to reset d (lines 15 to 21). Lastly, the distance between two data sequences

is stored in the last element of array dp (i.e., $dp[m]$). As a result, using two arrays of size m whose values are dynamically updated instead of using an array of size $m \times n$ reduces the space complexity from $O(mn)$ to $O(m)$.

Considering our applications on embedded IoT devices, we further reduced memory usage by one-third by removing arrays that was used for the starting position of the detection in [98] but is not necessary in our target applications. Additionally, this work used the Euclidean distance as the distance metric (line 13) instead of other metrics that can be used in DTW such as the squared Euclidean distance and the normalized Euclidean. This metric can make distance calculation lightweight, thus improving computation speed with little to no degradation of precision [100].

In the literature, several extensions of DTW have been studied to improve the alignment between data sequences. For example, Derivative DTW (DDTW) [101] is also applicable to our work. In DDTW, each input data point s_i is processed by the following equation. In our implementation, using fixed-point arithmetic maintains lightweight processing while making use of DDTW.

$$s_i = ((s_i - s_{i-1}) + (s_{i+1} - s_{i-1})/2)/2 \tag{5.1}$$

5.3.3 Software Development Flow

The aforementioned optimized DTW software algorithm is first developed in the C programming language. Thereafter, in order to execute the software algorithm on SSP, a compilation process is required to convert the C source code to an assemble language program and then to a binary program. Considering possible hardware changes in the future, a flexible and efficient software development flow is necessary for our work. Figure 5.3 shows the software development flow we built.

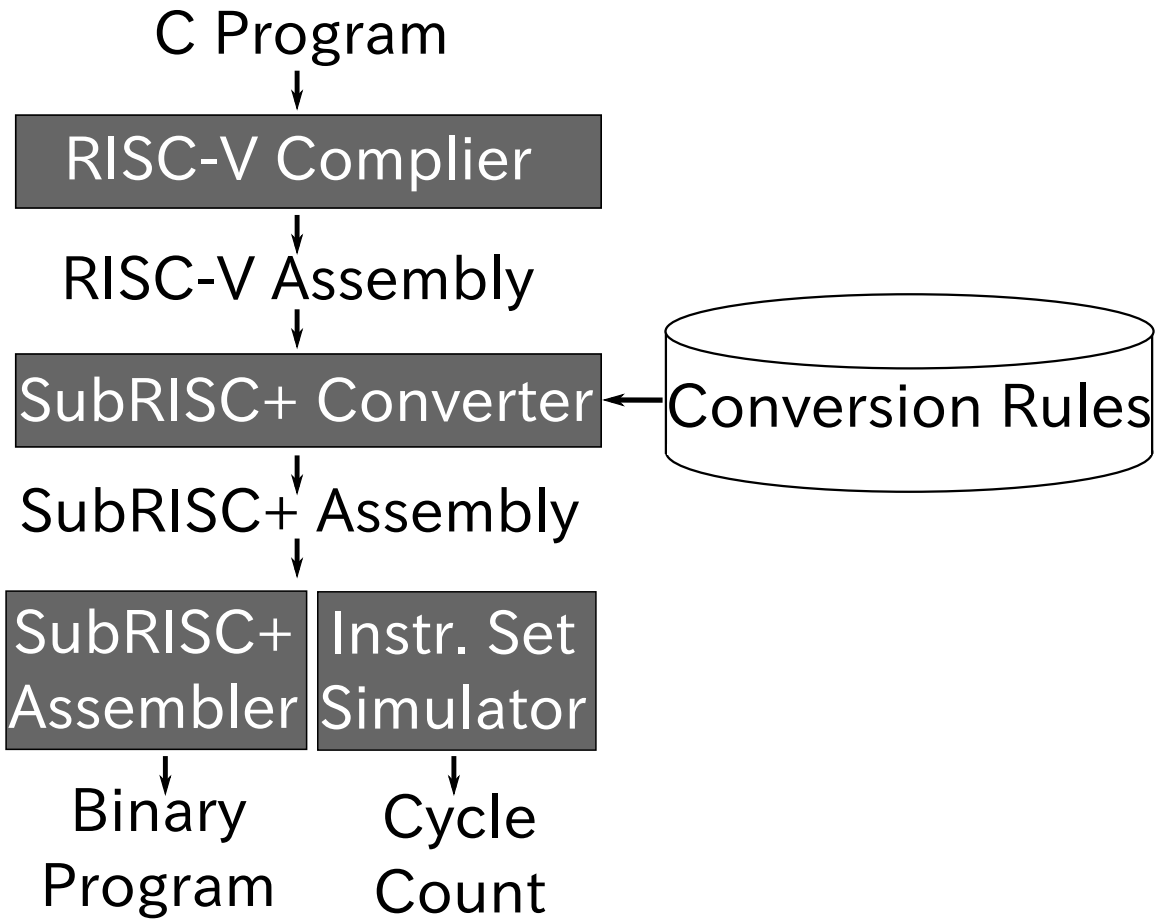


Figure 5.3: Software development flow [53]

We developed an intermediate-level assembly program converter to enable fast software development while maintaining flexibility for future hardware architecture specification changes. This assembly program converter was developed targeting minimize the design time used on assembly programming. We used the RISC-V GNU compiler toolchain[1] in our work to aid the development process. A variety of ISAs were defined for RISC-V processors that are used for different purposes. Among those ISAs of RISC-V processors, we selected the RV32E base integer instruction set [102] as its specification is most similar to the SSP ISA's specification. In addition, compiler's `-Os` option was used to reduce code size for

[1]https://github.com/riscv/riscv-gnu-toolchain

Table 5.2: Assembly code conversion examples [53]

RV32E		SSP	Note
`bgt t1,t2,.L1`	→	`sub t1,t2,t3,.L1`	one 32-bit instruction supporting a branch
`bne a0,t1,.L2`	→	`sub a0,t1,t3,.L2` `sub t1,a0,t3,.L2`	two 32-bit instructions supporting a branch
`not a0,a1`	→	`sub -1,a1,a0`	one 16-bit instruction

our evaluation because size of the program is also an important factor for our SSP processor.

After compiling and obtaining assembly code in the RISC-V RV32E format, we then used our assembly program converter to covert the instructions into the SSP ISA's format. The converter is supplied with rules to convert RISC-V RV32E to the SSP ISA. These conversion rules can be easily extended, for example, to support complex built-in functions. In addition, conversion rules are also flexible for changes by modifying the existing rules or adding new rules. The conversion has two steps: (1) change the operands order and (2) replace with instruction sequences composed of SSP's instructions. Conversion (1) is applied to all types of instructions because the order of operands differs between the RV32E and SSP ISA. In general, the destination register of the SSP ISA is the third operand while the destination register of the RISC-V RV32E is the first operand. Conversion (2) is applied to only instructions that are not supported in the SSP ISA. RV32E instructions are replaced by one or more SSP instructions only if one-to-one replacement is not possible Table 5.2 illustrates three examples for conversion.

The first example shows a conversion of a "branch if greater than" (*bgt*) instruction. A jump in *bgt* instruction occurs if the first operand is greater than the second operand. For SSP, *bgt* instruction is achieved by one 32-bit subtraction instruction. A subtraction between the second operand and the first operand is negative is equivalent to the first operand is greater than the second operand.

In this case, a jump to the specified label will occur. This example illustrates a RISC-V jump instruction is replaced by one SSP subtraction instruction.

The second example shows a conversion of a "branch if not equal" (*bne*) instruction. In a *bne* instruction jump occurs if the two operands' values are not equal to each other. For SSP, this operation is achieved using two 32-bit subtraction instructions, each of which supports a branch. Here each instruction takes four operands: three temporal registers and one label. The branch occurs if the first value is smaller than the second or if the second value is smaller than the first. This is equivalent to a case when the two values are not equal. This example illustrates a RISC-V jump instruction is replaced by two SSP subtraction instructions only when one-to-one replacement is not possible.

The third example shows the conversion of a *not* instruction. In a *not* instruction the value of the operand is flipped on every bit. All bits in the operand can be flipped when the input value is subtracted from 111...1, where the length of the series of 1s equals that of the input value. This can be shown by a simple theorem that on the bit level $0-1 = 1$ and $1-1 = 0$. Using the two's complement ($111...1 = -1$), a single subtraction operation can replace a *not* operation. This example illustrates a RISC-V arithmetic instruction is replaced by one SSP subtraction instruction.

Binary code is generated by processes the assembly code using the SSP assembler. In the hardware evaluation, switching factors for power consumption are generated by the binary program. The binary program is also used in our in-house SSP instruction-set simulator (ISS) to evaluate the cycle counts.

5.4 Evaluation

5.4.1 Experimental Setup

Firstly, we introduce a setup for circuit synthesis evaluation. By comparing it against other state-of-the-art processors, we demonstrate the effectiveness of the proposed processor in terms of circuit area. We also compare the power and energy consumption of the Cortex-M0, SubRISC+ and SSP under the same conditions to clearly measure the power efficiency of our work against the Cortex-M0, which is a widely-used commercial processor[2]. In our eHealth applications, compare to than energy consumption, power consumption is a more suitable metric. Because most eHealth devices are required to work constantly by reducing power consumption as long as the processing deadline is not exceeded.

We utilized the Cortex-M0 gate-level netlist provided by the ARM University Program in the comparison. Peripheral modules in the gate-level netlist were excluded to give a fair comparison. For the RISC-V processors, we adjusted some parameters on register files and multiplier/divisor module in the open-source RTL code based on the descriptions in [103]. The RISC-V Ibex adopts the RV32IMC ISA, which supports a multiplier/divisor module and has a register file of 32 entries. The RISC-V micro-riscy adopts the RV32E ISA, which supports no multiplier-/divisor module and has a register file of 16 entries. We utilized the RTL code of our processor and fine-tuned some implementations such as replacing the flip-flop-based RF with a latch-based RF to further optimize the proposed processor. We synthesized the RTL designs for these processor cores using Synopsis Design Compiler-F2011.09-SP2 with the libraries of TSMC 65nm low power technology to report circuit area and power consumption. The synthesis conditions were set as following, supply voltage of 1.0 V, clock frequency of 50 MHz, temperature of

[2]https://github.com/lowRISC/ibex

25 °C and typical-typical (TT) process corner. The clock frequency of 50 MHz was set referring to a real commercial product based on the ARM Cortex-M0 [104].

The switching activity in DTW processing was used to report reasonably accurate power results. In addition to the processor cores, we also compared the memory size of our work (6 KB total, with 4 KB for Dmem and 2 KB for Imem) against the memory size of the three processors based on their original setups [103, 105].

Secondly, we describe the setup for execution time results. We used the One Instruction-Set Computer Simulator and Assembler[3] developed for OISC processors and the proposed processor. Target processor of the simulator was set to proposed to measure the execution time of DTW algorithm on different data amounts. We made minor modifications on the simulator to improve the execution efficiency. The execution time was calculated based on the cycle count given by the instruction-set simulator and frequency of the proposed processor.

5.4.2 Experimental Results

We compared the circuit area and power/energy consumption of SSP against those of three state-of-the-art processors together with the SubRISC+ for proving the effectiveness of our proposed processor. Table 5.3 shows the circuit area in gate equivalents (GE) of the processor cores and total memory size in the second and fourth columns, respectively. GE shows the total circuit area relative to the area of a NAND gate so that GE could provide us less process technology dependent results on the circuit area. Results on synthesis were obtained by putting all of the processors under the same conditions. Similarly, Table 5.4 shows processor core power and energy consumption, respectively. In Tables 5.3 and 5.4, values in parentheses are normalized to SSP's results. This design employs a power reduction implementation known as clock-gating that depends on the device technology utilized (UMC 65nm). Because we could not measure RISC-

[3]https://github.com/Hara-Laboratory/oiscsim

Table 5.3: Comparisons in terms of circuit area [53]

Processor	Core Area [GE]		Mem [KB]	
Cortex-M0	17,043.8	(×2.92)	128	(×21.33)
Ibex	21,627.0	(×3.71)	64	(×10.67)
micro-riscy	16,195.0	(×2.77)	64	(×10.67)
SubRISC+	5,619.0	(×0.96)	6	(×1.00)
SSP	5,836.3	(×1.00)	6	(×1.00)

V power consumption under our conditions (TSMC 65nm LP) or refer to their results by excluding the device-dependent features in a fair manner, we compared SSP against the Cortex-M0 and SubRISC+ only.

Although the compared processor cores shown in Table 5.3 are relatively small compared to other embedded processors, the proposed processor processor is even smaller, at approximately one-third to one-fourth the size of the other processors. As an example of much larger embedded processors, the ARM Cortex-A9 has 6.5 million GE [106]. We also observed a slight increase in the circuit area of the SSP compared to SubRISC+, attributable to the enhancements made to the SSP design. In addition, the proposed processor has a much smaller memory size which is one order of magnitude smaller than the others embedded processors. This shows the small scale of the entire circuit and low fabrication cost of the proposed processor.

The circuit area reduction also contributes to decreases in both the dynamic power and leakage power as shown in Table 5.4. Although the compared processors support power-gating to suppress power consumption, their power consumption is still much higher than that of SSP (×2.70 and ×3.28 greater for dynamic and leakage power consumption, respectively). Based on the power consumption and cycle count data, the SSP shows better energy efficiency results than Cortex-M0 and comparable results against SubRISC+. This indicates that our approach of simplifying functionality is more effective than other conventional power-reduction

techniques. These results demonstrate our work's effectiveness on the hardware level in terms of circuit area (or fabrication cost) and power/energy consumption.

Table 5.4: Comparisons in terms of power and energy consumption [53]

Processor	Dyn. [uW/MHz]		Leak. [uW]		Energy [uJ]	
Cortex-M0	7.08	(×2.70)	0.541	(×3.28)	6.27	(×3.94)
SubRISC+	2.62	(× 1.00)	0.167	(× 1.01)	1.59	(× 1.00)
SSP	2.62	(× 1.00)	0.165	(× 1.00)	1.59	(× 1.00)

In this section, we show that our software implementation satisfies the application's requirements for the different monitoring-and-detection eHealth applications listed in Table 5.1. We also demonstrate that the memory limitations of the SSP processor are satisfied. The execution time was calculated by the product of the cycle counts and the inverse of the clock frequency for different data amount settings (i.e., m). Additionally, memory usage on both data memory and instruction memory were provided. The data amount, execution time, memory usage for instructions and data, and the deadline for each application are shown from left to right in Table 5.5. The deadline was calculated on the basis of the data amount and input sampling rate. Although the deadline indicates the time to collect the sample data of the amount m, our system can be easily extended to handle streaming data.

As shown in Table 5.5, by achieving shorter execution times than the requirements, our work can successfully satisfy deadlines for all target applications. Based on the execution time and required time, there is a significantly large design opportunity to enhance the low-power feature while still satisfying the deadline. This provides more flexibility in power supply and clock supply choices from the embedded system design perspective. The deadlines are still satisfied when the frequency is reduced from 50MHz to 5MHz. This shows that our proposed processor can work at a even lower power consumption level.

Table 5.5: Results of execution time and memory usage [53]

Amount	Exec. Time [ms]		Mem. Usage [Byte]		App. Req. [ms]
	(50 MHz)	(5 MHz)	(Instr.)	(Data)	
100	7.37	73.7	212	1,600	990 [88]
128	12.10	121.0	212	2,048	1,270 [89]
200	29.65	296.5	212	3,200	3,980 [90]
256	48.65	486.5	212	4,096	6,375 [91]

Assuming data is in 32-bit data format, memory usage is also within the limitations of the SSP processor. As discussed in the previous section, data memory usage increases linearly with the data amount. However, instruction memory usage does not change with the data amount because the only the values of the parameters are changed when iteration counts increase. We can handle up to 256 data points under an assumption of using 32-bit data. In this case, each of four arrays *sample*, *template*, *d*, and *dp* stores contains 256 entries of 32-bit data. In total, the data memory usage is 4 KB. A shorter bitwidth often suffices for some data types For example, EEG signal in dataset [107] can be represented by 16-bit data) in IoT devices. This means that our work can handle larger amount of data by using data with less bits. It is useful for IoT devices to reduce the length of data (i.e., resolution of analog-to-digital converters (ADC)) in order to improve power efficiency while the requirement on accuracy can still be satisfied.

5.5 Summary

This chapter addressed a monitoring and detection of target diseases in various healthcare applications on our proposed low-power embedded processor, SSP. We adopted a lightweight algorithm, DTW, to realize effective-yet-simple monitoring and detection applications. Memory-conscious optimizations on the DTW

algorithm were done for implementation on a resource-constrained embedded processor. We also introduced the rapid software development flow we developed and used for development of applications.

Through our evaluations, we demonstrated that our hardware and software approaches cooperatively achieved greater efficiency that state-of-the-art embedded processors. In terms of circuit area (directly related to fabrication cost) and power/energy consumption, our work is better than three state-of-the-art embedded processors while satisfying application performance requirements (execution time) for different eHealth monitoring-and-detection applications.

Chapter 6

Conclusions

This research signifies a substantial step forward in addressing the fundamental challenges associated with designing embedded processors specifically for IoT applications. Our development of a novel embedded processor, specifically designed for resource-constrained IoT devices to process targeted lightweight applications, addresses the objectives on security requirements under various hardware constraints. The processor's unique ISA, which emphasizes low power consumption and compact circuit design, stands as a significant innovation. This approach enables IoT devices operate securely, even in environments with severe resource limitations, thus bridging the gap between the need for optimal PPA and the imperative for robust security in IoT devices.

The experimental results validate the effectiveness of the processor in handling lightweight IoT applications and fortifying them against power side-channel attacks, a prevalent and insidious security threat. This achievement is particularly relevant in the eHealth sector, where the protection of sensitive health data is of utmost importance. The processor demonstrates exceptional capability in ensuring data security while maintaining the efficiency and compactness, thereby offering a viable solution to one of the most pressing challenges in the IoT landscape.

Additionally, the implications of this research extend beyond the immediate realm

of IoT security. The principles and methodologies developed here can be adapted to a wide range of applications, potentially leading to broader innovations in both IoT and other technology-dependent sectors. The balance achieved between security and hardware resources serves as a guiding framework for future IoT developments, particularly in the context of resource-limited devices.

Looking ahead, embedded processors are facing more and more threats from evolving attacks, such as fault injection attacks. These attacks, which involve deliberately causing errors in a device's operation to gain insights into its internal workings, represent a significant challenge in side-channel security. Therefore, future work will also involve developing strategies and mechanisms to protect against such advanced attacks. This will include researching and implementing fault detection and tolerance techniques, enhancing the robustness of the processor against intentional disruptions. By incorporating defenses against such advanced attacks, we aim to further strengthen the security profile of our processor, ensuring it remains resilient against a broader spectrum of threats. This aspect is especially critical as embedded devices increasingly become targets of sophisticated attack methods in the rapidly evolving landscape of IoT security.

Additionally, an evaluation on ASIC implementation of the proposed processor would be insightful. This step will allow us to conduct more comprehensive evaluations and assessments, providing deeper insights into the processor's security capabilities in a widely adopted hardware environment. These future endeavors will contribute to the ongoing development and improvement of secure processing solutions in the field of IoT devices.

Acknowledgement

この研究を遂行するにあたり、終始適切な助言を賜り、また丁寧に指導して下さった原祐子先生に感謝いたします。 電気通信大学の崎山一男先生、李陽先生には，ハードウェアセキュリティの研究方法において指導を頂きました。心より感謝いたします。 また、実験にご協力をいただいたTanvir Ahemdさん、稲垣沙耶さんは感謝しております。 論文審査を務めていただいた尾形わかは先生、高橋篤司先生、本村真人先生、佐々木広先生、李陽先生には、貴重なご助言をいただきました。感謝を申し上げます。 そして、原研究室の皆様には、日頃より多大なご協力とご支援を頂きました。深く感謝いたします。 最後に、これまで自分を支えてくれた両親に感謝いたします。

List of Publications

Journals:

- Mingyu Yang, Tanvir Ahmed, Saya Inagaki, Yang Li, Kazuo Sakiyama, and Yuko Hara-Azumi,Hardware/Software Cooperative Design against Power Side-channel Attacks on IoT Devices, IEEE Internet of Things Journal, 2024 (Accepted)
- Saya Inagaki, Mingyu Yang, Yang Li, Kazuo Sakiyama, and Yuko Hara-Azumi, Power Side-channel Attack Resistant Circuit Designs of ARX Ciphers Using High-level Synthesis, ACM Transactions on Embedded Computing Systems, 2023
- Mingyu Yang and Yuko Hara-Azumi, Implementation of Lightweight eHealth Applications on a Low-Power Embedded Processor, IEEE Access, vol.8, pp.121724-121732, Jul. 2020.

International Conferences:

- Maki Tsukahara, Haruka Hirata, Mingyu Yang, Daiki Miyahara, Yang Li, Yuko Hara-Azumi, and Kazuo Sakiyama, On the Practical Dependency of Fresh Randomness in AES S-box with Second-Order TI, International Symposium on Computing and Networking (CANDAR), Nov. 2023
- Mingyu Yang and Yuko Hara-Azumi, Co-Design of Lightweight eHealth Applications on a IoT Egde Processor, Ph.D. Forum of Design, Automation & Test in Europe (DATE), Apr. 2023

- Saya Inagaki, Mingyu Yang, Yang Li, Kazuo Sakiyama and Yuko Hara-Azumi, Power Side-channel Countermeasures for ARX Ciphers using High-level Synthesis, International Symposium on Field-Programmable Gate Arrays (ISFPGA), p.52, Feb. 2023
- Saya Inagaki, Mingyu Yang, Yang Li, Kazuo Sakiyama and Yuko Hara-Azumi, Examining Vulnerability of HLS-designed Chaskey-12 Circuits to Power Side-Channel Attacks, International Symposium on Quality Electronic Design (ISQED), Apr. 2022
- Mingyu Yang and Yuko Hara-Azumi, Implementation and Evaluation of an Embedded Processor for Lightweight IoT eHealth, University Booth of Design, Automation & Test in Europe (DATE), Mar. 2020.

Patent:

- 発明者：原 祐子，佐宗 馨，楊 明宇，少命令セット組込みプロセッサ, 出願日：2019年9月24日，出願人：東京工業大学

Others:

- 楊 明宇， 崎山 一男， 李 陽， 原 祐子， IoTエッジ向け組込みプロセッサにおける軽量暗号Simonの閾値法の実装に関する検討，LSIとシステムのワークショップ，東京，2023年5月10日
- 稲垣 沙耶, 楊 明宇, 李 陽, 崎山 一男, 原 祐子, "電力サイドチャネル攻撃に対して堅牢なARX型暗号回路の高位合成," 暗号と情報セキュリティシンポジウム (SCIS), 北九州, 2023年1月26日
- 楊 明宇，Tanvir Ahmed, 稲垣 沙耶, 崎山 一男, 李 陽, 原 祐子, "ハードウェア/ソフトウェア協調設計によるIoTデバイスの電力解析攻撃対策," 第199回システムとLSIの設計技術研究発表会，2022-SLDM-199(4), 2188-8639, 京都，2022年11月11日
- 楊 明宇, 崎山 一男, 李 陽, 原 祐子, "低電力組込みプロセッサの電力

解析攻撃耐性に関する検討,"LSIとシステムのワークショップ，東京，2022年5月17日

- 楊 明宇, 卯木 あゆ美, 李 陽, 崎山 一男, 原 祐子, "少命令セット組込みプロセッサにおけるARX型暗号アルゴリズムの実装と評価," 暗号と情報セキュリティシンポジウム (SCIS), 大阪/オンライン, 2022年1月18日
- 渡辺 陸, 楊 明宇, 原 祐子, 崎山 一男, 李 陽, "RISC−VとSubRISC+におけるLED暗号のBitslice実装の評価," 暗号と情報セキュリティシンポジウム (SCIS), 大阪/オンライン, 2022年1月18日
- 稲垣 沙耶, 楊 明宇, 李 陽, 崎山 一男, 原 祐子, “高位合成による軽量暗号 Chaskey の FPGA実装およびサイドチャネル攻撃耐性の評価,” ハードウェアセキュリティ研究会, VLD2020-79, HWS2020-54 (2021-03), pp.61-66, オンライン, 2021 年 3 月 4 日

References

[1] A. Ukil, S. Bandyoapdhyay, C. Puri and A. Pal, 'IoT Healthcare Analytics: The Importance of Anomaly Detection,' in *2016 IEEE 30th international conference on advanced information networking and applications (AINA)*, IEEE, 2016, pp. 994–997 (cit. on p. 1).

[2] L. Catarinucci *et al.*, 'An IoT-Aware Architecture for Smart Healthcare Systems,' *IEEE internet of things journal*, vol. 2, no. 6, pp. 515–526, 2015 (cit. on p. 1).

[3] A. Gaur, B. Scotney, G. Parr and S. McClean, 'Smart City Architecture and its Applications Based on IoT,' *Procedia computer science*, vol. 52, pp. 1089–1094, 2015 (cit. on p. 1).

[4] F. Firouzi, B. Farahani, M. Ibrahim and K. Chakrabarty, 'From EDA to IoT eHealth: Promises, Challenges, and Solutions,' *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 12, pp. 2965–2978, Dec. 2018 (cit. on pp. 1, 59, 61).

[5] S. M. Iqbal, I. Mahgoub, E. Du, M. A. Leavitt and W. Asghar, 'Advances in Healthcare Wearable Devices,' *NPJ Flexible Electronics*, vol. 5, no. 1, p. 9, 2021 (cit. on p. 1).

[6] M. N. K. Boulos, S. Wheeler, C. Tavares and R. Jones, 'How Smartphones are Changing the Face of Mobile and Participatory Healthcare: an Overview, with Example from eCAALYX,' *Biomedical engineering online*, vol. 10, no. 1, pp. 1–14, 2011 (cit. on p. 1).

[7] A. Ullah, M. Azeem, H. Ashraf, A. A. Alaboudi, M. Humayun and N. Z. Jhanjhi, 'Secure Healthcare Data Aggregation and Transmission in IoT—A Survey,' *IEEE Access*, vol. 9, pp. 16 849–16 865, 2021 (cit. on p. 1).

[8] P. Gope and T. Hwang, 'BSN-Care: A Secure IoT-Based Modern Healthcare System Using Body Sensor Network,' *IEEE sensors journal*, vol. 16, no. 5, pp. 1368–1376, 2015 (cit. on p. 1).

[9] H. Tao, M. Z. A. Bhuiyan, A. N. Abdalla, M. M. Hassan, J. M. Zain and T. Hayajneh, 'Secured Data Collection with Hardware-Based Ciphers for

IoT-Based Healthcare,' *IEEE Internet of Things Journal*, vol. 6, no. 1, pp. 410–420, 2018 (cit. on p. 1).

[10] J. Deogirikar and A. Vidhate, 'Security attacks in IoT: A survey,' in *2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC)*, IEEE, 2017, pp. 32–37 (cit. on p. 2).

[11] A. A. Pammu, K.-S. Chong, W.-G. Ho and B.-H. Gwee, 'Interceptive Side Channel Attack on AES-128 Wireless Communications for IoT Applications,' in *2016 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*, IEEE, 2016, pp. 650–653 (cit. on p. 2).

[12] M. Randolph and W. Diehl, 'Power Side-channel Attack Analysis: A Review of 20 Years of Study for the Layman,' *Cryptography*, vol. 4, no. 2, p. 15, 2020 (cit. on p. 2).

[13] L. Yan, Y. Guo, X. Chen and H. Mei, 'A Study on Power Side Channels on Mobile Devices,' in *Proceedings of the 7th Asia-Pacific Symposium on Internetware*, 2015, pp. 30–38 (cit. on p. 2).

[14] B. L. R. Stojkoska and K. V. Trivodaliev, 'A Review of Internet of Things for Smart Home: Challenges and Solutions,' *Journal of cleaner production*, vol. 140, pp. 1454–1464, 2017 (cit. on p. 2).

[15] S. Singh, P. K. Sharma, S. Y. Moon and J. H. Park, 'Advanced Lightweight Encryption Algorithms for IoT Devices: Survey, Challenges and Solutions,' *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–18, 2017 (cit. on p. 2).

[16] K. Yang, D. Blaauw and D. Sylvester, 'Hardware Designs for Security in Ultra-low-power IoT Systems: An Overview and Survey,' *IEEE Micro*, vol. 37, no. 6, pp. 72–89, 2017 (cit. on p. 2).

[17] M. Wolf, *Computers as Components: Principles of Embedded Computing System Design*. Elsevier, 2012 (cit. on p. 6).

[18] S. Wong, S. Vassiliadis, S. Cotofana *et al.*, 'Future Directions of Programmable and Reconfigurable Embedded Processors,' *Domain-specific processors: systems, architectures, modeling, and simulation*, pp. 149–235, 2004 (cit. on p. 6).

[19] S. Mittal, 'A Survey of Techniques for Improving Energy Efficiency in Embedded Computing Systems,' *International Journal of Computer Aided Engineering and Technology*, vol. 6, no. 4, pp. 440–459, 2014 (cit. on pp. 6, 7).

[20] S. F. Johann, M. T. Moreira, L. S. Heck, N. L. Calazans and F. P. Hessel, 'A Processor for IoT Applications: An Assessment of Design Space and Trade-offs,' *Microprocessors and Microsystems*, vol. 42, pp. 156–164, 2016 (cit. on p. 7).

[21] Z. Sheng, C. Mahapatra, C. Zhu and V. C. Leung, 'Recent Advances in Industrial Wireless Sensor Networks toward Efficient Management in IoT,' *IEEE access*, vol. 3, pp. 622–637, 2015 (cit. on p. 7).

[22] X. Zeng, S. K. Garg, P. Strazdins, P. P. Jayaraman, D. Georgakopoulos and R. Ranjan, 'IOTSim: A Simulator for Analysing IoT Applications,' *Journal of Systems Architecture*, vol. 72, pp. 93–107, 2017 (cit. on p. 7).

[23] Q. Wang and I. Balasingham, 'Wireless Sensor Networks-an Introduction,' *Wireless sensor networks: application-centric design*, pp. 1–14, 2010 (cit. on p. 8).

[24] A. Malinowski and H. Yu, 'Comparison of Embedded System Design for Industrial Applications,' *IEEE transactions on industrial informatics*, vol. 7, no. 2, pp. 244–254, 2011 (cit. on p. 8).

[25] P. Pillai and K. G. Shin, 'Real-time Dynamic Voltage Scaling for Low-power Embedded Operating Systems,' in *Proceedings of the eighteenth ACM symposium on Operating systems principles*, 2001, pp. 89–102 (cit. on p. 8).

[26] K. Usami and N. Ohkubo, 'A Design Approach for Fine-grained Run-time Power Gating Using Locally Extracted Sleep Signals,' in *2006 International Conference on Computer Design*, IEEE, 2006, pp. 155–161 (cit. on p. 8).

[27] D. She, Y. He and H. Corporaal, 'Energy Efficient Special Instruction Support in an Embedded Processor with Compact ISA,' in *Proceedings of the 2012 international conference on Compilers, architectures and synthesis for embedded systems*, 2012, pp. 131–140 (cit. on p. 8).

[28] S. Annaratone, *Digital CMOS circuit design*. Springer Science & Business Media, 2012, vol. 16 (cit. on p. 8).

[29] N. H. Weste and D. Harris, *CMOS VLSI Design: a Circuits and Systems Perspective*. Pearson Addison-Wesley, 2011 (cit. on p. 9).

[30] J. Baillieul and P. J. Antsaklis, 'Control and Communication Challenges in Networked Real-time Systems,' *Proceedings of the IEEE*, vol. 95, no. 1, pp. 9–28, 2007 (cit. on p. 10).

[31] A. Jafari, N. Buswell, M. Ghovanloo and T. Mohsenin, 'A low-power wearable stand-alone tongue drive system for people with severe disabilities,' *IEEE transactions on biomedical circuits and systems*, vol. 12, no. 1, pp. 58–67, 2017 (cit. on p. 10).

[32] H. Ghasemzadeh and R. Jafari, 'Ultra Low-power Signal Processing in Wearable Monitoring Systems: A Tiered Screening Architecture with Optimal Bit Resolution,' *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 13, no. 1, pp. 1–23, 2013 (cit. on p. 10).

[33] C. Rolfes, A. Poschmann, G. Leander and C. Paar, 'Ultra-lightweight implementations for smart devices–security for 1000 gate equivalents,' in *Smart Card Research and Advanced Applications: 8th IFIP WG 8.8/11.2 International Conference, CARDIS 2008, London, UK, September 8-11, 2008. Proceedings 8*, Springer, 2008, pp. 89–103 (cit. on p. 11).

[34] S. K. Mousavi, A. Ghaffari, S. Besharat and H. Afshari, 'Security of internet of things based on cryptographic algorithms: A survey,' *Wireless Networks*, vol. 27, pp. 1515–1555, 2021 (cit. on p. 11).

[35] A. J. Clark, 'Optimisation heuristics for cryptology,' Ph.D. dissertation, Queensland University of Technology, 1998 (cit. on p. 11).

[36] Y. Xiao, M. Li, S. Chen and Y. Zhang, 'Stacco: Differentially analyzing side-channel traces for detecting ssl/tls vulnerabilities in secure enclaves,' in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 859–874 (cit. on p. 12).

[37] A. I. Newaz, A. K. Sikder, M. A. Rahman and A. S. Uluagac, 'A Survey on Security and Privacy Issues in Modern Healthcare Systems: Attacks and Defenses,' *ACM Transactions on Computing for Healthcare*, vol. 2, no. 3, pp. 1–44, 2021 (cit. on p. 12).

[38] G. Joy Persial, M. Prabhu and R. Shanmugalakshmi, 'Side Channel Attack-Survey,' *Int. J. Adv. Sci. Res. Rev*, vol. 1, no. 4, pp. 54–57, 2011 (cit. on p. 12).

[39] K. Tiri, 'Side-channel Attack Pitfalls,' in *Proceedings of the 44th annual Design Automation Conference*, 2007, pp. 15–20 (cit. on p. 12).

[40] M. Zhang, A. Raghunathan and N. K. Jha, 'Towards Trustworthy Medical Devices and Body Area Networks,' in *Proceedings of the 50th Annual Design Automation Conference*, 2013, pp. 1–6 (cit. on p. 13).

[41] D. D. Hwang, P. Schaumont, K. Tiri and I. Verbauwhede, 'Securing Embedded Systems,' *IEEE Security & Privacy*, vol. 4, no. 02, pp. 40–49, 2006 (cit. on p. 13).

[42] M. Méndez Real and R. Salvador, 'Physical side-channel attacks on embedded neural networks: A survey,' *Applied Sciences*, vol. 11, no. 15, p. 6790, 2021 (cit. on p. 13).

[43] A. A. Pammu, K.-S. Chong and B.-H. Gwee, 'Highly secured arithmetic hiding based s-box on aes-128 implementation,' in *2016 International Symposium on Integrated Circuits (ISIC)*, IEEE, 2016, pp. 1–4 (cit. on p. 13).

[44] F.-X. Standaert, E. Peeters and J.-J. Quisquater, 'On the masking countermeasure and higher-order power analysis attacks,' in *International Conference on Information Technology: Coding and Computing (ITCC'05)-Volume II*, IEEE, vol. 1, 2005, pp. 562–567 (cit. on p. 13).

[45] T. Güneysu and A. Moradi, 'Generic side-channel countermeasures for reconfigurable devices,' in *International workshop on cryptographic hardware and embedded systems*, Springer, 2011, pp. 33–48 (cit. on p. 13).

[46] P. Sasdrich, A. Moradi and T. Güneysu, 'Hiding higher-order side-channel leakage: Randomizing cryptographic implementations in reconfigurable hardware,' in *Topics in Cryptology–CT-RSA 2017: The Cryptographers' Track at the RSA Conference 2017, San Francisco, CA, USA, February 14–17, 2017, Proceedings*, Springer, 2017, pp. 131–146 (cit. on p. 13).

[47] P. Ravi, S. Bhasin, J. Breier and A. Chattopadhyay, 'Ppap and ippap: Pll-based protection against physical attacks,' in *2018 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, IEEE, 2018, pp. 620–625 (cit. on p. 13).

[48] M. Montoya, T. Hiscock, S. Bacles-Min, A. Molnos and J. Fournier, 'Adaptive Masking: a Dynamic Trade-off between Energy Consumption and Hardware Security,' in *2019 IEEE 37th International Conference on Computer Design (ICCD)*, IEEE, 2019, pp. 559–566 (cit. on p. 13).

[49] A. Moradi, 'Advances in Side-channel Security,' Ph.D. dissertation, Bochum, Ruhr-Universität Bochum, Habil.-Schr., 2015, 2016 (cit. on p. 14).

[50] C. Herbst, E. Oswald and S. Mangard, 'An AES Smart Card Implementation Resistant to Power Analysis Attacks,' in *International conference on applied cryptography and network security*, Springer, 2006, pp. 239–252 (cit. on p. 14).

[51] J. Gaspoz and S. Dhooghe, 'Threshold implementations in software: Micro-architectural leakages in algorithms,' *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 155–179, 2023 (cit. on p. 14).

[52] K. Saso and Y. Hara-Azumi, 'Revisiting Simple and Energy-Efficient Embedded Processor Designs Towards the Edge Computing,' *IEEE Embedded Systems Letters*, vol. 12, no. 2, pp. 45–49, 2020 (cit. on pp. 15, 23, 24, 31, 46, 47, 59).

[53] M. Yang and Y. Hara-Azumi, 'Implementation of Lightweight eHealth Applications on a Low-Power Embedded Processor,' *IEEE Access*, vol. 8, pp. 121 724–121 732, 2020 (cit. on pp. 19, 31, 35, 51, 58, 61, 64, 66, 67, 69, 71, 72, 76–78).

[54] M. Yang, A. Tanvir, S. Inagaki, Y. Li, K. Sakiyama and Y. Hara-Azumi, 'Hardware/Software Cooperative Design against Power Side-channel Attacks on IoT Devices,' *IEEE Internet of Things Journal (Accepted)*, 2024 (cit. on pp. 23, 26, 29, 34, 36–38, 41, 43, 46, 47).

[55] V. Samadi Bokharaie and A. Jahanian, 'Power Side-channel Leakage Assessment and Locating the Exact Sources of Leakage at the Early Stages of ASIC Design Process,' *The Journal of Supercomputing*, vol. 78, pp. 2219–2244, 2022 (cit. on pp. 26, 27, 35, 38).

[56] D. Blackman and S. Vigna, 'Scrambled Linear Pseudorandom Number Generators,' *ACM Transactions on Mathematical Software (TOMS)*, vol. 47, no. 4, pp. 1–32, 2021 (cit. on p. 28).

[57] A. Bogdanov *et al.*, 'PRESENT: An Ultra-lightweight Block Cipher,' in *Proc. of International Workshop on Cryptographic Hardware and Embedded Systems*, 2007, pp. 450–466 (cit. on p. 30).

[58] H. D. Tsague and B. Twala, 'Practical Techniques for Securing the Internet of Things (IoT) Against Side Channel Attacks,' in *Internet of Things and Big Data Analytics toward Next-generation Intelligence*, Springer, 2018, pp. 439–481 (cit. on p. 30).

[59] E. Prouff and M. Rivain, 'Masking against Side-channel Attacks: A Formal Security Proof,' in *Proc. of International Conference on the Theory and Applications of Cryptographic Techniques*, 2013, pp. 142–159 (cit. on p. 30).

[60] M. Rivain, E. Prouff and J. Doget, 'Higher-order Masking and Shuffling for Software Implementations of Block Ciphers,' in *Proc. of International Workshop on Cryptographic Hardware and Embedded Systems*, 2009, pp. 171–188 (cit. on p. 30).

[61] J. D. Golic, 'Techniques for Random Masking in Hardware,' *IEEE Trans. on Circuits and Systems I*, vol. 54, no. 2, pp. 291–300, 2007 (cit. on p. 30).

[62] J.-S. Coron *et al.*, 'Conversion of Security Proofs from One Leakage Model to Another: A New Issue,' in *Proc. of Constructive Side-channel Analysis and Secure Design*, 2012, pp. 69–81 (cit. on pp. 30, 31, 44, 45).

[63] L. De Meyer, E. De Mulder and M. Tunstall, 'On the Effect of the (Micro) Architecture on the Development of Side-channel Resistant Software,' *Cryptology ePrint Archive*, 2020 (cit. on pp. 30, 36).

[64] E. De Mulder, S. Gummalla and M. Hutter, 'Protecting RISC-V against Side-channel Attacks,' in *Proc. of Design Automation Conference*, 2019, pp. 1–4 (cit. on pp. 30–32, 35, 56).

[65] S. Gao *et al.*, 'An Instruction Set Extension to Support Software-Based Masking,' *IACR Trans. on Cryptographic Hardware and Embedded Systems*, vol. 2021, no. 4, pp. 283–325, Aug. 2021 (cit. on pp. 31, 32, 35, 56).

[66] W. Diehl *et al.*, 'Side-channel Resistant Soft Core Processor for Lightweight Block Ciphers,' in *Proc. of International Conference on ReConFigurable Computing and FPGAs*, 2017, pp. 1–8 (cit. on pp. 31, 32, 35, 56).

[67] B. Marshall, D. Page and T. Hung Pham, 'A Lightweight ISE for ChaCha on RISC-V,' in *Proc. of International Conference on Application-specific Systems, Architectures and Processors*, 2021, pp. 25–32 (cit. on pp. 31, 32, 35).

[68] K. Stangherlin and M. Sachdev, 'Design and Implementation of a Secure RISC-V Microprocessor,' *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 30, no. 11, pp. 1705–1715, 2022 (cit. on p. 31).

[69] B. Gigerl *et al.*, 'Coco: Co-Design and Co-Verification of Masked Software Implementations on CPUs,' in *Proc. of USENIX Security Symposium*, 2021, pp. 1469–1468 (cit. on pp. 32, 37, 44, 46, 47, 55, 56).

[70] N. Mouha, B. Mennink, A. Van Herrewege, D. Watanabe, B. Preneel and I. Verbauwhede, 'Chaskey: An efficient mac algorithm for 32-bit microcontrollers,' in *International Conference on Selected Areas in Cryptography*, Springer, 2014, pp. 306–323 (cit. on p. 33).

[71] N. Mouha, 'Chaskey: A MAC Algorithm for Microcontrollers–Status Update and Proposal of Chaskey-12–,' Ph.D. dissertation, Inria Paris Rocquencourt, 2015 (cit. on p. 33).

[72] R. Beaulieu, S. Treatman-Clark, D. Shors, B. Weeks, J. Smith and L. Wingers, 'The SIMON and SPECK Lightweight Block Ciphers,' in *Proc. of Design Automation Conference*, 2015, pp. 1–6 (cit. on p. 34).

[73] M. Dworkin, 'Recommendation for Block Cipher Modes of Operation. Methods and Techniques,' National Inst. of Standards and Technology Gaithersburg MD Computer security Div, Tech. Rep., 2001 (cit. on p. 34).

[74] H. Kapadia, L. Benini and G. De Micheli, 'Reducing Switching Activity on Datapath Buses with Control-signal Gating,' *IEEE Journal of Solid-State Circuits*, vol. 34, no. 3, pp. 405–414, 1999 (cit. on p. 39).

[75] J.-S. Coron and L. Goubin, 'On Boolean and Arithmetic Masking against Differential Power Analysis,' in *Proc. of International Workshop on Cryptographic Hardware and Embedded Systems*, 2000, pp. 231–237 (cit. on p. 39).

[76] L. Goubin, 'A Sound Method for Switching between Boolean and Arithmetic Masking,' in *Proc. of International Workshop on Cryptographic Hardware and Embedded Systems*, 2001, pp. 3–15 (cit. on p. 40).

[77] B. Gigerl, R. Primas and S. Mangard, 'Formal Verification of Arithmetic Masking in Hardware and Software,' *Cryptology ePrint Archive*, 2022 (cit. on p. 40).

[78] A. Shahverdi, M. Taha and T. Eisenbarth, 'Silent Simon: A Threshold Implementation under 100 Slices,' in *Proc. of International Symposium on Hardware Oriented Security and Trust*, 2015, pp. 1–6 (cit. on p. 42).

[79] M. Nassar, Y. Souissi, S. Guilley and J.-L. Danger, 'RSM: A Small and Fast Countermeasure for AES, Secure against 1st and 2nd-order Zero-Offset SCAs,' in *Proc. of Design, Automation & Test in Europe Conference*, 2012, pp. 1173–1178 (cit. on p. 42).

[80] Y. Hori, T. Katashita, A. Sasaki and A. Satoh, 'SASEBO-GIII: A Hardware Security Evaluation Board Equipped with a 28-nm FPGA,' in *Proc. of Global Conference on Consumer Electronics*, 2012, pp. 657–660 (cit. on p. 45).

[81] T. Schneider and A. Moradi, 'Leakage Assessment Methodology,' *Journal of Cryptographic Engineering*, vol. 6, pp. 85–99, 2016 (cit. on p. 50).

[82] R. Bogue, 'Recent Developments in MEMS Sensors: A Review of Applications, Markets and Technologies,' *Sensor Review*, 2013 (cit. on p. 59).

[83] J.-C. Liou, 'The Advent of Application Specific Integrated Circuits (ASIC)-MEMS within the Medical System,' in *Telehealth*. IntechOpen, 2018 (cit. on p. 59).

[84] T. R. Bennett *et al.*, 'MotionSynthesis Toolset (MoST): A Toolset for Human Motion Data Synthesis and Validation,' in *Proc. of MobiHoc Workshop on Pervasive Wireless Healthcare*, 2014, pp. 25–30 (cit. on p. 59).

[85] C. Savaglio, P. Pace, G. Aloi, A. Liotta and G. Fortino, 'Lightweight Reinforcement Learning for Energy Efficient Communications in Wireless Sensor Networks,' *IEEE Access*, vol. 7, pp. 29 355–29 364, 2019 (cit. on pp. 59, 64).

[86] T. Adegbija, A. Rogacs, C. Patel and A. Gordon-Ross, 'Microprocessor Optimizations for the Internet of Things: A Survey,' *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 1, pp. 7–20, 2018 (cit. on p. 59).

[87] N. Scarpato, A. Pieroni, L. Di Nunzio and F. Fallucchi, 'E-health-IoT Universe: A Review,' *management*, vol. 21, no. 44, p. 46, 2017 (cit. on p. 59).

[88] A. Dalton *et al.*, 'Development of a Body Sensor Network to Detect Motor Patterns of Epileptic Seizures,' *IEEE Trans. on Bio-medical Engineering*, vol. 59, pp. 3204–11, 2012 (cit. on pp. 60, 61, 78).

[89] N. Kale, J. Lee, R. Lotfian and R. Jafari, 'Impact of Sensor Misplacement on Dynamic Time Warping Based Human Activity Recognition Using Wearable Computers,' in *Proc. of the Conf. on Wireless Health*, 2012 (cit. on pp. 60–62, 78).

[90] J. Barth *et al.*, 'Subsequence Dynamic Time Warping as a Method for Robust Step Segmentation Using Gyroscope signals of Daily Life Activities,' in *Proc. of Annual Inter'l Conf. of the IEEE Engineering in Medicine and Biology Society*, 2013, pp. 6744–6747 (cit. on pp. 60, 61, 78).

[91] B. Raghavendra, D. Bera, A. S. Bopardikar and R. Narayanan, 'Cardiac Arrhythmia Detection Using Dynamic Time Warping of ECG Beats in e-Healthcare Systems,' in *Proc. of IEEE Int'l Symp. on a World of Wireless, Mobile and Multimedia Networks*, 2011, pp. 1–6 (cit. on pp. 60–62, 78).

[92] R. Varatharajan and G. Manogaran, 'Wearable Sensor Devices for Early Detection of Alzheimer Disease Using Dynamic Time Warping Algorithm,' *Cluster Computing*, vol. 21, 2018 (cit. on pp. 61, 63).

[93] T. N. Alotaiby *et al.*, 'EEG Seizure Detection and Prediction Algorithms: a Survey,' *EURASIP Journal on Advances in Signal Processing*, vol. 2014, no. 1, p. 183, 2014 (cit. on p. 62).

[94] W. Zhang *et al.*, 'A Novel Cardiac Arrhythmia Detection Method Relying on Improved DTW Method,' in *Proc. of Advanced Information Technology, Electronic and Automation Control Conf.*, 2017, pp. 862–867 (cit. on p. 62).

[95] R. Lotfian and R. Jafari, 'An Ultra-Low Power Hardware Accelerator Architecture for Wearable Computers Using Dynamic Time Warping,' in *Proc. of Design, Automation Test in Europe Conference Exhibition*, 2013, pp. 913–916 (cit. on p. 62).

[96] G. ten Holt, M. Reinders and E. Hendriks, 'Multi-Dimensional Dynamic Time Warping for Gesture Recognition,' in *Proc. of Annual Conf. of the Advanced School for Computing and Imaging*, 2007 (cit. on p. 63).

[97] D. J. Berndt and J. Clifford, 'Using dynamic time warping to find patterns in time series.,' in *KDD workshop*, Seattle, WA, USA: vol. 10, 1994, pp. 359–370 (cit. on p. 64).

[98] Y. Sakurai, C. Faloutsos and M. Yamamuro, 'Stream Monitoring under the Time Warping Distance,' in *Proc. of Int'l Conf. on Data Engineering*, 2007, pp. 1046–1055 (cit. on pp. 64, 68, 70).

[99] S. U. Jan, S. Ahmed, V. Shakhov and I. Koo, 'Toward a Lightweight Intrusion Detection System for the Internet of Things,' *IEEE Access*, vol. 7, pp. 42 450–42 471, 2019 (cit. on p. 64).

[100] N. Kulkarni, 'Effect of Dynamic Time Warping using Different Distance Measures on Time Series Classification,' *International Journal of Computer Applications*, vol. 179, pp. 34–39, Dec. 2017 (cit. on p. 70).

[101] E. J. Keogh and M. J. Pazzani, 'Derivative Dynamic Time Warping,' in *Proc. of Int'l Conf. on Data Mining. Society for Industrial and Applied Mathematics*, 2001, p. 11 (cit. on p. 70).

[102] A. Waterman *et al.*, 'The RISC-V Instruction Set Manual,' *Volume I: User-Level ISA', version*, vol. 2.1, 2016 (cit. on p. 71).

[103] P. D. Schiavone *et al.*, 'Slow and Steady Wins the Race? A Comparison of Ultra-Low-Power RISC-V Cores for Internet-of-Things Applications,' in *Proc. of Int'l Symp. on Power and Timing Modeling, Optimization and Simulation*, 2017, pp. 1–8 (cit. on pp. 74, 75).

[104] STMicroelectronics, 'STM32F038x6 ARM®-based 32-bit MCU with 32 Kbyte Flash, 9 Timers, ADC and Communication Interfaces, 1.8 V,' *DoclD026079 Rev 5*, 2017 (cit. on p. 75).

[105] ARM, *Cortex^{TM}-M0 Revision: r0p0 Technical Reference Manual*, 2009 (cit. on p. 75).

[106] A. Thakur, 'ARM (Advanced RISC Machines) Processors,' *EngineersGarage.com*, 2011 (cit. on p. 76).

[107] R. G. Andrzejak *et al.*, 'Indications of Nonlinear Deterministic and Finite-dimensional Structures in Time Series of Brain Electrical Activity: Dependence on Recording Region and Brain State,' *Physical Review E*, vol. 64, no. 6, p. 061 907, 2001 (cit. on p. 78).