An Inexact Ultra-low Power Bio-signal Processing Architecture With Lightweight Error Recovery

SOUMYA BASU, Swiss Federal Institute of Technology Lausanne (EPFL), Switzerland
LORIS DUCH, Swiss Federal Institute of Technology Lausanne (EPFL), Switzerland
RUBÉN BRAOJOS, Swiss Federal Institute of Technology Lausanne (EPFL), Switzerland
GIOVANNI ANSALONI, Università della Svizzera Italiana (USI), Switzerland
LAURA POZZI, Università della Svizzera Italiana (USI), Switzerland
DAVID ATIENZA, Swiss Federal Institute of Technology Lausanne (EPFL), Switzerland

The energy efficiency of digital architectures is tightly linked to the voltage level (Vdd) at which they operate. Aggressive voltage scaling is therefore mandatory when ultra-low power processing is required. Nonetheless, the lowest admissible Vdd is often bounded by reliability concerns, especially since static and dynamic non-idealities are exacerbated in the near-threshold region, imposing costly guard-bands to guarantee correctness under worst-case conditions. A striking alternative, explored in this paper, waives the requirement for unconditional correctness, undergoing more relaxed constraints. First, after a run-time failure, processing correctly resumes at a later point in time. Second, failures induce a limited Quality-of-Service (QoS) degradation. We focus our investigation on the practical scenario of embedded bio-signal analysis, a domain in which energy efficiency is key, while applications are inherently error-tolerant to a certain degree. Targeting a domain-specific multi-core platform, we present a study of the impact of inexactness on application-visible errors. Then, we introduce a novel methodology to manage them, which requires minimal hardware resources and a negligible energy overhead. Experimental evidence show that, by tolerating 900 errors/hour, the resulting inexact platform can achieve an efficiency increase of up to 24%, with a QoS degradation of less than 3%.

CCS Concepts: •**Computer systems organization** →**Reliability;** *Multicore architectures; Embedded and cyber-physical systems;* •**Applied computing** →*Health care information systems;*

Additional Key Words and Phrases: Low-power architectural optimization, Inexact computing, Wireless Body Sensor Nodes

ACM Reference format:
Soumya Basu, Loris Duch, Rubén Braojos, Giovanni Ansaloni, Laura Pozzi, and David Atienza. 0000. An Inexact Ultra-low Power Bio-signal Processing Architecture With Lightweight Error Recovery. *ACM Trans. Embedd. Comput. Syst.* 0, 0, Article 00 (0000), 19 pages.
DOI: 0000001.0000001

This work has been partially supported by the E4Bio (grant no. 200021-159853) and the MagicISEs (grant no. 200021-156397) projects evaluated by the Swiss NSF, the BodyPoweredSenSE (grant no. 20NA21-143069) RTD project evaluated by the Swiss NSF and funded by Nano-Tera.ch with Swiss Confederation financing, and by a joint research grant for ESL-EPFL by IMEC.
Authors' addresses: S. Basu, L. Duch, R. Braojos and D. Atienza are with the Embedded Systems Laboratory (ESL), EPFL, Lausanne, Switzerland. G. Ansaloni, L. Pozzi are with the Faculty of Informatics, Università della Svizzera Italiana (USI), Lugano, Switzerland.
 1539-9087/0000/0-ART00 $15.00
DOI: 0000001.0000001

This article was presented in the International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS) 2017 and appears as part of the ESWEEK-TECS special issue, Vol. 0, No. 0, Article 00. Publication date: 0000.

1 INTRODUCTION

The emergence of embedded devices, able to continuously acquire and wirelessly transmit the sensed data, is fostering a revolution across the IT landscape [3], opening novel and exciting opportunities in many fields, ranging from environmental protection [30] to domotics [16].

Among them, healthcare applications are of particular interest, especially the ones related to monitoring chronic cardiovascular disorders [1]. In this scenario, sensor appliances (named Wireless Body Sensor Nodes, WBSNs) enable the long-term acquisition of bio-signals, outside of a hospital environment and with minimal medical supervision [18].

Efficiency is key for WBSNs, as the saved energy translates both in smaller form factors (by requiring smaller batteries) and longer autonomies. Herein, we focus our investigation on the energy optimization of the Digital Signal Processing (DSP) applications executing on WBSNs. Such routines analyze acquisitions, deriving compact feature sets, which are then transmitted on the wireless link [7]. They must be supported within a tight energy envelope, because DSP itself is often the efficiency bottleneck in body sensor nodes.

In this context, this paper explores the applicability of inexact processing as an energy-saving strategy. We showcase how, **by relaxing the reliability requirements of the underlying architecture, important energy gains can be achieved**, with negligible impact on the DSP Quality-of-Service (QoS), and therefore on the clinical value of the acquired data. We observe that, to guarantee correct operations, even in worst-case conditions, nominal supply voltages should be set conservatively, providing large guard-bands, which in turn lead to large costs in terms of energy efficiency. We take a different stance: we waive the complete correctness requirement, assuming instead a non-zero probability of run-time errors. The rationale behind our approach, illustrated in Figure 1, is that in WBSN applications acquisitions are always corrupted by noise, while DSP outputs are often qualitative or statistical in nature. Consequently, errors can be tolerated if the system is never trapped in an inconsistent state due to a failure. In other words, we guarantee that the effect of a random error is always limited in time, and eventually, normal execution will resume.

Our methodology bridges the error tolerance exposed by bio-signal DSP, which we characterize in this paper, with the energy saving opportunities deriving from inexact computing [34]. This approach links the characteristics of nano-scale ICs and those of application scenarios in a cross-layer, hardware/software co-design framework.

We enforce relaxed reliability through low-overhead mechanisms based on a run-time synchronization policy, memory protection, and input- and intermediate-buffer surveillance. Moreover, we show how integrity checks can be implemented both before and after the wireless transmission stage, to assess the correctness of the performed DSP routines and discard incorrect outputs. By trading-off failure rates and efficiency, our goal is to **maximize the amount of *correct* outputs**

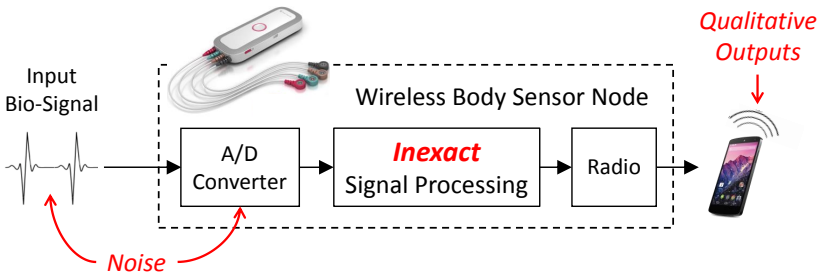

Fig. 1. Bio-signal analysis algorithms performed by WBSNs are resilient to errors, since they process noisy inputs to derive qualitative and/or statistical outputs.

This article was presented in the International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS) 2017 and appears as part of the ESWEEK-TECS special issue, Vol. 0, No. 0, Article 00. Publication date: 0000.

for a given energy envelope, while at the same time minimizing the impact of failures (and therefore incorrect results) on the QoS.

We consider failures which manifest themselves as random bit-flips anywhere in the memory subsystem, caused by a combination of static and dynamic effects [4] [49]. We instead neglect failures due to particle strikes, since they are rare events (few can be expected in a billion hours of operation, with a weak dependency on the operating voltage [48]). SRAM banks, hosting instructions and data, are in fact the least reliable elements of digital architectures when operating at ultra-low voltage levels [24], while combinational circuits (such as Arithmetic Logic Units, ALUs) and Flip-Flop-based registers have higher resiliencies.

We target two applications in the bio-signal DSP domain to verify our proposed strategy. The first one is Compressed Sensing (CS), which performs the concurrent compression of multiple input Electrocardiogram signals (leads). The second is a combination of Multi-Lead Filtering (MF), which removes noise introduced by muscle activity, system AC supply interferences and base drift due to breathing, and Compressed Sensing, termed MF-CS.

Our contribution is three-fold:

- Analyzing the application-visible impact of memory failures in WBSN DSP routines, we explore the benefits and challenges of relaxed reliability as an energy minimization strategy.
- We introduce novel reliability-aware mechanisms at the hardware and software levels, which guarantee the system-level resilience, embodying them in an ultra-low power multi-core architecture [6].
- We investigate the efficiency of the resulting inexact system, and the ensuing trade-off between error rates, energy consumption and QoS degradation.

The rest of the paper is structured as follows. Section 2 summarizes related works in the addressed field. Section 3 categorizes the errors which arise due to supply voltage scaling. Then, in Section 4, we analyze the strategies to mitigate them. Section 5 describes the adopted experimental framework in order to test our developed strategy. The results obtained from the experiments are discussed in Section 6, followed by Section 7, with a summary of the main scientific conclusions obtained.

2 STATE OF THE ART AND TECHNOLOGY FOUNDATIONS

2.1 Smart Wireless Body Sensor Nodes

Cardiovascular diseases are nowadays the major cause of death worldwide, and are projected to become even more prevalent in the future [46]. To observe the cardiac conditions of affected subjects, long-term monitoring sessions are required, which are labor-intensive for the medical staff (leading to costs in the range of billions of Euros [14]) and have a major impact on the quality of life of patients. In this context, sensor nodes devoted to the healthcare domain can bring major benefits for patients and providers alike.

Numerous WBSN platforms have been proposed in recent years, both as research projects [29] and as commercial products [20]. Among the bio-signals monitored by these devices, the most common are Electrocardiograms (ECGs, with a number of inputs varying from one to twelve), oxygen saturation (SpO_2), skin impedance and blood pressure [38]. WBSNs require small form factors and long autonomies, characteristics which demand a high energy efficiency. Toward this end, domain-specific Analog-to-Digital Converters (ADCs) [47] and low-power transmission protocols [21] have been proposed.

An orthogonal approach to increase efficiency is to perform an *interpretation* of the acquired data on the sensor node, retrieving high-level features of clinical relevance from the acquisitions, and only sending these through the energy-hungry wireless link. Retrieved features vary according to medical requirement, ranging from a compressed representation of the inputs, to the fiducial points

This article was presented in the International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS) 2017 and appears as part of the ESWEEK-TECS special issue, Vol. 0, No. 0, Article 00. Publication date: 0000.

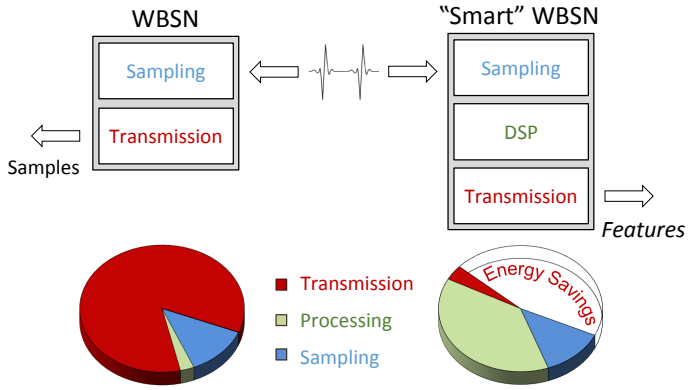

Fig. 2. DSP can tangibly reduce the energy requirements of WBSNs. Data from [6] (features extraction), [47] (ADC) and [21] (transmission).

of the ECG heartbeats, to the identification of normal or pathological events through automated classification [7]. To perform the required processing, the ensuing "smart" WBSNs must embed sizable computing capabilities, which themselves have non-negligible energy footprints. The energy profile of smart WBSNs therefore differs with respect to the one of devices that only sense and transmit data in two important ways, as depicted in Figure 2. First, the total energy is markedly reduced. Second, their energy bottleneck is often in the DSP stage itself, calling for careful architectural optimization in the digital processing system.

Indeed, low-power processors devoted to bio-signal analysis have been proposed in the literature. Digital architectures in this domain mainly leverage aggressive Voltage-Frequency Scaling (VFS) [41] [2] [42], operation-level parallelism [19] [11] or application specific accelerators [9] [26], in order to lower energy requirements. These works consider a two-dimensional design space, trading-off costs (area and power) for performance (clock speed, instructions per cycle). We enrich this view by adding a third dimension, that of the desired QoS. Our study is therefore orthogonal with respect to traditional architectural explorations. In fact, although in Section 6 we show the benefits of our approach when embodied in a state-of-the-art multi-core platform [6], our methodology can be applied as a generic strategy, irrespective of the implementation details of the underlying hardware.

2.2 Inexact and Near-Threshold Computing

For decades, the decrease in the transistor physical size ensured that circuits could be operated at increasingly lower supply levels, leading to ever-smaller energy requirements. This trend, known as Dennard scaling [10], came to an abrupt halt around 2006, and voltage supply levels have been mostly stagnant since then. The breakdown of Dennard scaling is compounded by two additional challenges: a) the electrical characteristics of Integrated Circuits (ICs) are increasingly difficult to control in nano-scale technologies, resulting in a high level of fabrication-time *variability* [4] and b) supply voltages, especially at low levels, experience fairly large fluctuations at run-time (*droops*) [12]. Indeed, almost 40% of the energy consumption of the processor studied in [49] is devoted to countering variability and droops.

These effects are especially challenging in the *Near-Threshold Computing* (NTC) domain [33]. The rationale behind NTC is that transistors are most efficient when operated at, or just above, their threshold voltage (around 500mV). In this region, although reductions in supply voltages result in a

decrease of the maximum operating frequency, they are also responsible for a substantial increase in the energy efficiency. Such a trade-off is highly attractive for energy-constrained scenarios, such as the one of WBSN DSP [11]. When instead voltage supplies fall below the threshold voltage, efficiency gains flatten out, while performance penalties steeply increase.

In the NTC range, the reliability of traditional six-transistors (6T) SRAMs is particularly critical. 8T and 10T SRAMs [35], which have separate read and write paths, are more resilient towards voltage scaling, but require additional area and complexity, as do solutions based on Error Correcting Codes (ECC). Logic components can instead better tolerate lower voltage supplies, especially when operating at low frequencies of a few MHz, which is usual for ultra-low power systems. Indeed, a dedicated NTC architecture, having separate voltage domains for memories and logic, has been introduced by the authors of [24]. However, this strategy incurs a large overhead to generate and distribute different supply levels and for the voltage converters, required for signals crossing the different voltage islands [31].

The hefty, and increasing, cost of ensuring correctness motivates research works in *inexact computing*. Inexact strategies either rely on simplified circuits to realize approximate boolean functions [40] or, as in the scenario considered in this paper, on under-supplied and unreliable architectures [5]. While previous investigations have been made on inexact computation in other application domains such as high performance multimedia processing [25], [28], [45], they generally focus on errors related to timing violations, as performance is critical to those systems. However, in the domain of ECG processing, such high frequencies of operation are unnecessary, and hence timing violations are not a concern.

Our strategy shares some similarities with [13], [8], [27]. As opposed to [13], our methodology does not require a strict partitioning of the computations and of the data types between exact and approximate, an approach that places a high burden on application programmers. With respect to [8], we do not restrict the fault-prone memory area to a portion of the data memory; on the contrary, we consider (and manage) faults in the entire memory system, including the whole data memory (comprising dynamic structures such as the stacks) as well as the instruction memory. In [27] and [17], related error recovery strategies are investigated. However, the strategy in [27] is restricted to soft errors and is centered on artificial intelligence applications, while the work presented in [17] primarily deals with mitigating fluctuations arising out of hardware variabilities and mainly focuses on software mitigation strategies.

3 VDD SCALING, ERRORS AND FAILURES

To assess the impact of variability and voltage droops at run-time, we take a two-step approach, illustrated in Figure 3. First, we derive from a nominal supply level (Vdd_{nom}) and a statistical voltage distribution, the failure probability of a target memory structure. Then, failure probabilities are linked to application-visible errors, while the nominal voltage is employed to retrieve the energy consumption of the system. Those two steps are detailed below, while the strategy implemented to minimize the impact of faults at the system level is described in Section 4.

3.1 Failure Model

SRAM failures (bit-flips) at NTC voltages can be caused by different mechanisms [37]. Hold failures happen when the content of a memory cell is lost, due to leakage currents, while not being accessed. Write stability failures occur when the correct value is not stored in a cell, even when the write time is infinite. Read and write timing failures appear when access times are too short to transfer a state to/from a cell. Finally, a read upset erroneously flips a memory content on a read operation.

This article was presented in the International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS) 2017 and appears as part of the ESWEEK-TECS special issue, Vol. 0, No. 0, Article 00. Publication date: 0000.

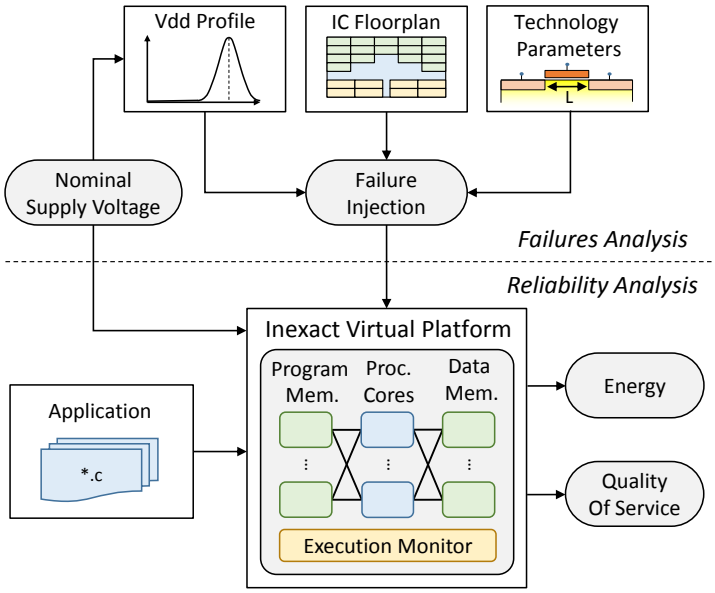

Fig. 3. The developed framework analyzes the impact of hardware failures (top) at the application level, when executing on the proposed error-resilient platform (bottom).

The probability of these events depends on a number of factors: the adopted voltage supply, the implementation of the cells, the memory organization and its static variability. In particular, hold failures hamper the reliability of SRAMs at much lower voltages than other effects, while read upsets are averted by construction when employing 8T cells (which we use in our target system). We do not therefore consider these two failure modes in our experimental setup.

We rely on VARIUS-NTV [23] to obtain failure rates due to stability and timing violations for an instantaneous voltage supply level ($p_{err}(Vdd_t)$), according to the floorplan and the memory structure of the target architecture (detailed in Section 5): the number of banks, the bank size, their word width, etc. Similarly to [23], we examine the effect of static variability on the memory reliability by considering multiple (100) variation maps, generated by VARIUS-NTV. In a traditional setting, all faults must be avoided, requiring a supply level such that $p_{err}(Vdd_t) = 0$ under all

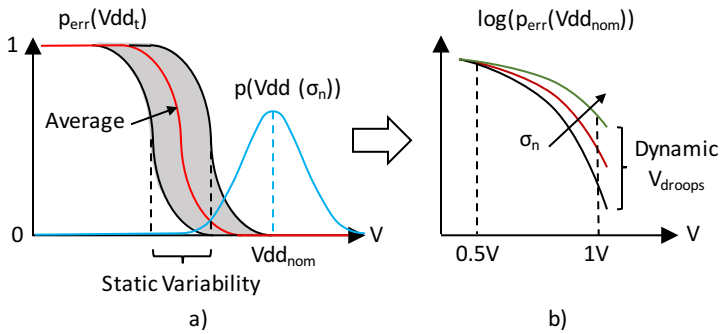

Fig. 4. Error rates at nominal voltage (right), dependent on static variability (left, grey area) and voltage noise profile (left, blue line).

This article was presented in the International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS) 2017 and appears as part of the ESWEEK-TECS special issue, Vol. 0, No. 0, Article 00. Publication date: 0000.

variability conditions. By adopting an inexact computing stance, a non-zero fault probability is instead tolerable at the system level. Such a scenario allows us to focus on the average (instead of worst-case) dependency between faults and voltage supply.

Voltage droop effects are accounted by modelling Vdd fluctuations as a normal distribution centered on Vdd_{nom} ($N(Vdd_{nom}, \sigma_n)$). This formulation is in good accordance with the empirical evidence reported in [15] and [49], while, at the same time, abstracting the details of power delivery system implementations (such as the design of the voltage regulators and of the power delivery network), whose study is outside the scope of this paper. The error probability for a given nominal supply voltage and noise profile is then the area below both curves, as depicted in Figure 4. The error rates when varying Vdd_{nom} can therefore be derived as follows:

$$p_{err}(Vdd_{nom}, \sigma_n) = \int_0^\infty p_{err}(Vdd_t) \times N(Vdd_{nom}, \sigma_n)\, dVdd_t$$

3.2 Application-level Errors

Bit-flips manifest as application-visible outcomes that may affect both the code execution flow and the produced data, leading to a degradation of the QoS. According to the severity of the impact, errors are typically classified into three different groups [36]:

- *Masked errors:* A bit-flip does not influence the runtime flow of the application nor the output or nature of the results. Masked errors are generated when an erroneous bit does not have any significance on the content a word represents (e.g.: unused bits in instruction encodings), or does not have any effect on the result of a computation.
- *Silent Data Corruption errors (SDC):* From a runtime point of view the application does not suffer any major change and it produces the correct amount of results at the required rate. However, the bit-flip influences the output by corrupting it to some degree with respect to the expected one. While some of these errors can be the result of data modifications (e.g., a bit-flip while a processing core reads a word representing a sample of an acquired signal), others can be produced by changes in the execution flow (i.e. premature exit form a loop, or a function). These incorrect run-time behaviors can be monitored and detected up to some extent, as later explained in Section 4. It is important to note that the impact of some of these errors can be unrecoverable, like the unwanted modification of values in the read-only section of the data memory. Such operations must therefore be always avoided.
- *Unrecoverable errors:* A bit-flip leads to a runtime change in the application that either breaks the execution flow, as when a processing core jumps to an uninitialized region of the instruction memory (i.e. out of the memory footprint of an application), or drastically interferes with its execution (e.g., by getting trapped in an infinite loop). These errors result in an inconsistent system state, because as the processor runs unexpected instructions, it finishes processing before / after it should, or its execution is blocked. Unrecoverable errors can be the result of bit-flips affecting both the instructions (e.g., a conditional branch changes to be unconditional when evaluating a loop iteration), or the data words (e.g., the content of the link register stored in the stack is modified resulting in a jump to an incorrect memory location).

As explained in Section 3.1, we model memory failures depending on fluctuations of the voltage supply and on static variability parameters, which are both agnostic of the memory content. Instead, the severity of the effect of a bit-flip depends on the entity that is stored at the affected memory location. Moreover, the probability of an error occurrence heavily depends on the access frequency

(read or write) of the memory words. Therefore, given a memory footprint of an application in which only a single bit-flip can result in an unrecoverable error, the probability of an unrecoverable error to happen depends on the execution trace of the application. More precisely, it is proportional to the amount of times that the word containing that bit is used. As a result, to calculate the probability of each type of error events, we consider in this work both the consequences of bit-flips depending on their position in memory and the benchmarks' access patterns at run-time.

4 ERROR MITIGATION

We propose a lightweight scheme for mitigating the effects resulting from the bit-flips described in the preceding section. Its aim is to detect, on the node itself, all the unrecoverable errors, and to appropriately cope with them by restoring the system to a coherent state. While our strategy results in data loss (by requiring a system reset), it also ensures that computation will resume correctly afterwards, effectively localizing faults in small time windows without suffering permanent effects. Moreover, the proposed strategy intends to maximize the detection of SDC errors on-board, by monitoring abnormalities in the execution flow. Further SDC errors are also filtered out on the receiver side, by analyzing the characteristics of the transmitted data.

4.1 Strategies

Our approach to error mitigation monitors critical system components and run-time events while processing, as well as the computation outcomes:

a) Runtime coherence: DSP systems, especially when featuring multiple cores, must implement a policy to synchronize their run-time execution. Synchronization events, in turn, must follow a legal pattern (e.g., a processor should only be notified to resume its execution due to an event, only if it was waiting for it). An illegal synchronization sequence can only be the result of an unrecoverable or a SDC error, which resulted in an incorrect execution behavior.

b) FIFO surveillance: Data is exchanged between peripherals (e.g., ADCs) and processors, or between processors, through FIFOs, which are sized to meet the real-time constraints of the intended workloads. Nonetheless, due to an error, it is possible that a FIFO overflows, either because a processor is not able to read from it (e.g., it got stuck in an infinite loop), or because an incorrect and higher amount of data is being written into the FIFO. Overflows can therefore be used as symptoms indicating the occurrence of an unrecoverable or a SDC error.

c) Memory protection: Some portions of the memory, typically the application read-only section of the Data Memory (DM), should never be written to. These memory sections typically store constant parameters that must not be modified during execution. A write request to these memory sections signals a forbidden memory access, generated by an error in the addressing mechanism (either in the instruction or in the processor stack that resides in the DM).

d) Detection of Instruction Memory (IM) access violations: Failures can cause a incorrect jump to an un-initialized memory location. Accesses to addresses outside of the IM portion of an application can be monitored to detect the occurrence of an unrecoverable error.

e) Receiver-side detection of erroneous results: After wireless transmission, further error detection strategies can be performed by the receiver. For example, an effective approach, adopted in this work, consists in comparing the frequency spectrum of the received data against the expected one, having an a-priori knowledge of the application admissible values. Such data integrity analysis must be performed without any intervening error, and cannot therefore be executed on the inexact WBSN.

This article was presented in the International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS) 2017 and appears as part of the ESWEEK-TECS special issue, Vol. 0, No. 0, Article 00. Publication date: 0000.

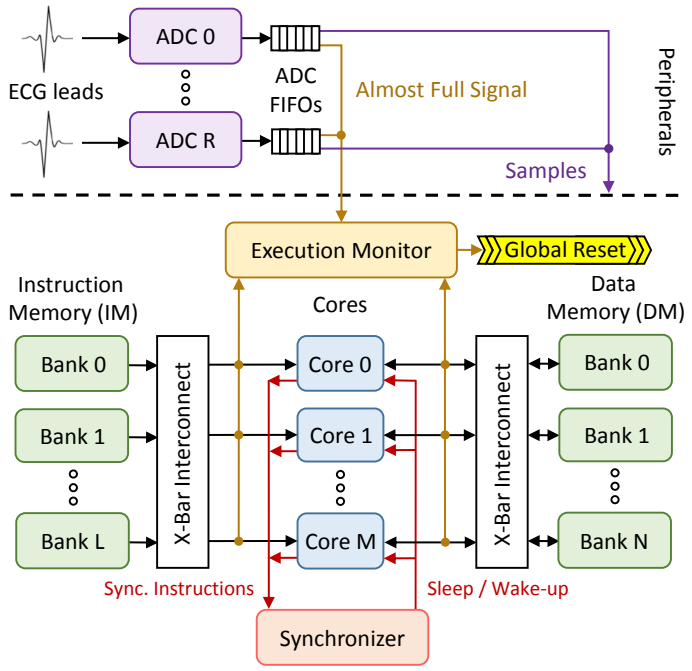

Fig. 5. High-level block scheme of the inexact WBSN architecture.

4.2 Implementation

The WBSN-side error management strategies (from (a) to (d) in Section 4.1) can be implemented with little hardware overhead by monitoring synchronization events, the state of FIFOs, and IM / DM accesses. Herein, we detail a concrete example applied to the multi-core system described in [6]. The hardware components devoted to failure management are encapsulated in an *Execution Monitor* module, which interfaces with the processors, as depicted in Figure 5. The monitor detects synchronization instructions issued by the processing cores. In addition, it gets notified when software FIFOs are being accessed. Moreover, it is connected to the "Almost Full" signals of the ADC hardware FIFOs, which are typically available in these peripherals. Finally, the monitor also surveils the addresses requested by the processing cores from both IM and DM.

Upon detection of an error, the Execution Monitor can activate a global reset to restart execution from a safe initial state. Crucially, the monitor does not embed any SRAM blocks, relying instead on registers that can be reliably operated at low voltages. The implementation details are described in the following paragraphs:

a) Runtime coherence: Synchronization instructions, managed by a hardware synchronizer component, are employed in the multi-core platform in [6] to manage Single Instruction Multiple Data (SIMD) executions and producer/consumer relationships between cores. In an error-free execution, synchronization events are processed in a Last-In-First-Out order, as a processor can only resume its execution from the last-entered synchronization point. In order to check coherence at runtime, a small hardware stack (of 32 words in our implementation) is provided in the Execution Monitor for each processor, storing the synchronization point IDs. A pop request with a different

ID with respect to the one at the top of the stack is therefore an indication of an error, and causes a system reset.

b) FIFO surveillance: ADCs typically embed FIFOs where samples are queued when they are acquired until a processor consumes them. Software FIFOs are also employed to store the intermediate results generated by the producer cores, before the consumer cores use them. In our approach both memory structures are monitored at runtime by two slightly different mechanisms:

- *ADC FIFO monitoring:* A signal is raised from each of the 8 ADCs in the system to the Execution Monitor module to inform the latter that the corresponding ADC FIFO is "Almost Full". Upon its reception, the Execution Monitor, in turn, issues the reset flag.
- *Internal FIFO monitoring:* A similar strategy employing an "Almost Full" and empty signals is also used for monitoring the internal FIFOs situated between the producer and the consumer cores. The Execution Monitor stores in a register the value corresponding to the number of words contained in each FIFO. A write request to an almost full FIFO or a read request to an empty one result in a system reset.

c) Lightweight memory protection: The goal of this error detection mechanism is to ensure that the read-only section part of the DM is never written to. For each core, 128 words are allocated for storing the read-only data, starting from the beginning of the private memory section. Programmers can specify a variable to be read-only during the linking phase of the source code compilation. At run-time, the Execution Monitor module checks write requests to the read-only section of the DM. Such a mechanism ensures that constant application parameters are never modified, which is crucial to ensure that errors are never propagated after a reset event.

d) Detection of IM footprint violations: Some unrecoverable errors are related to changes in the execution flow that make the cores jump to an incorrect IM address corresponding to an un-initialized memory location. To tackle this issue, all the IM words are pre-initialized to an instruction which forces the processor to branch unconditionally to a predefined error handler instruction. This instruction is then used to notify the Execution Monitor to immediately trigger a system reset.

The detection of errors can extend beyond the WBSN architecture, and include correctness checks in the receiver, in order to counter SDC errors (as mentioned in Section 4.1). While these checks are implemented in software (and not at the architectural level), they form an integral part of a system relying on inexact computing. Details of the implemented receiver-side error detection strategy are provided below. This strategy is applicable to ECG signals, but, with different frequency thresholds, it can be adapted to other bio-signal modalities as well.

e) Receiver-side detection of erroneous results: The impact of SDC errors can be further minimized at the receiver side by analyzing the power-spectrum of the received data, and filtering out windows of signals that do not correspond to the characteristics of an ECG acquisition. To this end, we first compare the power residing in the frequency range of 0-20 Hz (named Pf_{20}) and the power in the full signal (Pf_{full}), discarding signals with $(Pf_{20}/Pf_{full}) < 0.9$. Then, a second level of verification is carried out, where the power in the frequency range of 0-10 Hz (Pf_{10}) is compared with Pf_{20}, discarding the signals if the ratio $(Pf_{10}/Pf_{20}) < 0.1$. We validated this technique by processing 1080 random windows of 1024 samples from the MIT-BIH Normal Sinus database [39], achieving a rate of false positives (correct windows classified as corrupted by SDC errors) of less than 1%.

5 EXPERIMENTAL SETUP

We investigate the efficiency / QoS trade-off exposed by the relaxed reliability mechanism by considering two bio-signal analysis applications, written in C language, namely:

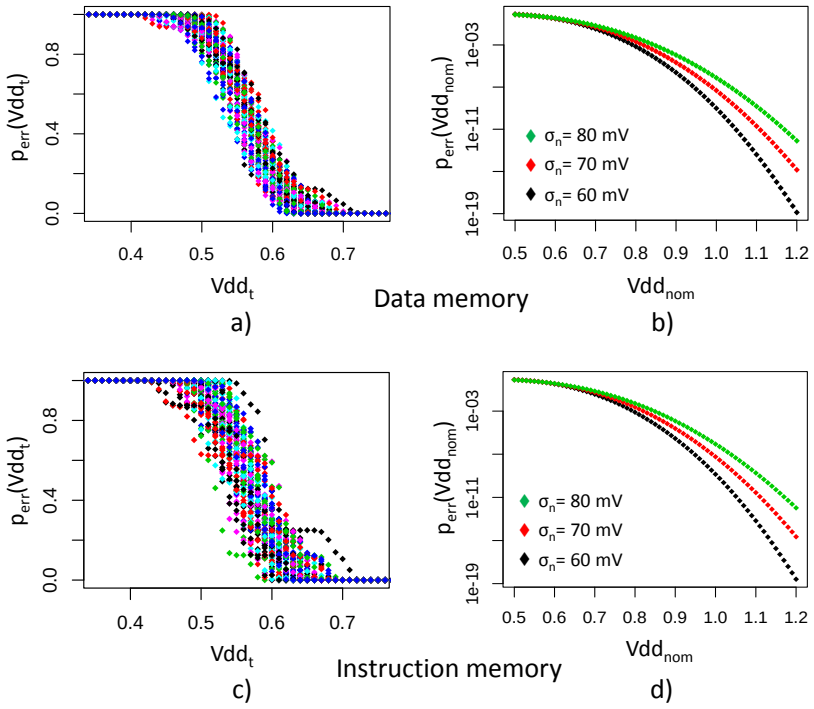

Fig. 6. Failure probabilities per bit in the instruction and data memory banks for 100 variation maps considering a fixed supply voltage value (a, c), and in the presence of supply voltage droops (b, d).

- Eight-lead Compressed Sensing (*CS*), which executes the simultaneous lossy encoding of eight ECG input signals [32], with each input stream being processed by a core.
- Morphological Filtering coupled with CS (*MF-CS*), which processes four ECG signals in parallel, filtering out high- and low-frequency noise via morphological operators [43], before performing their compression. For this application, four cores perform the filtering phase, producing data that is consumed by the other four cores executing the CS phase.

For both applications, we consider an acquisition frequency of the input signals of 500 samples / second. The target digital platform comprises eight processing cores, whose hardware structure and dedicated compiler are generated with Synopsys ASIP designer [44]. It operates at 5 MHz, a frequency that is sufficient to meet the real-time constraints for the considered benchmarks. Each core, featuring a three-stage pipeline, is interfaced with multi-banked instruction and data memories (implemented as 8T SRAMs) via logarithmic crossbars. The DM is arranged in 16 banks of 2 KWords each, with a word-size of 16 bits, while the IM is organized in 8 banks of 4 KWords, with a word size of 24 bits.

Embedded in the platform, a hardware synchronizer allows execution of code in SIMD mode whenever multiple cores access the same IM location in the same clock cycle [6]. Moreover, it pauses (clock gates) and resumes execution of cores in order to re-establish SIMD execution after divergences due to data-dependent branches and to avoid active waiting when no input data is available (either from the ADC or from another core). Finally, the Execution Monitor circuitry described in Section 4 overlooks the execution of the processors, restarting the system whenever an application-visible error is detected.

This article was presented in the International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS) 2017 and appears as part of the ESWEEK-TECS special issue, Vol. 0, No. 0, Article 00. Publication date: 0000.

The system has been synthesized in 65*nm* low-leakage technology to retrieve its area and characterize the energy profile of its components, relying on small synthetic benchmarks. The retrieved parameters are then employed to annotate a cycle-accurate SystemC model of the platform. This strategy enables lengthy simulations of real-world DSP applications, which would be impractically long if performed at the RTL level.

The failure rates in the memory subsystem are modeled with VARIUS-NTV [23], according to the methodology outlined in Section 3.1. SRAM technology parameters (e.g.: the gate capacitance, the channel length of transistors, etc.) are set according to the ITRS road-map [22]. Results, shown in Figure 6, consider 100 different variation maps and a Gaussian distribution of the voltage supply with a standard deviation of 60, 70 and 80mV.

For each error-injected simulation, a bit-flip has been introduced in the memory system to model memory failures (either one during an IM read operation or one while a DM address is read/written to). The output generated by the faulty executions of the benchmark applications were then compared to a failure-free execution, retrieving the incurred error type as described in Section 3.2.

6 EXPERIMENTAL RESULTS

In this section we analyze the effectiveness of our inexact computing approach from various standpoints. First, we show the application-visible outcomes of bit-flips in memory, while executing the two considered benchmarks. Then, we analyze the effectiveness and area overhead of the implemented error-recovery strategy. Finally, we provide evidence on the trade-off between energy efficiency and the drop in QoS, exhibited by the inexact architecture.

6.1 Error Analysis

To investigate the link between memory failures and application errors, we performed a series of simulated executions on the target system disabling error recovery. Each simulation run processed one ECG window (1024 samples per lead, corresponding to 2 seconds of simulated time), in the presence of a single bit-flip, randomly located in IM or DM. 3000 random tests were processed for the CS application and 500 for the MF-CS one, as the latter requires a considerably larger simulation time.

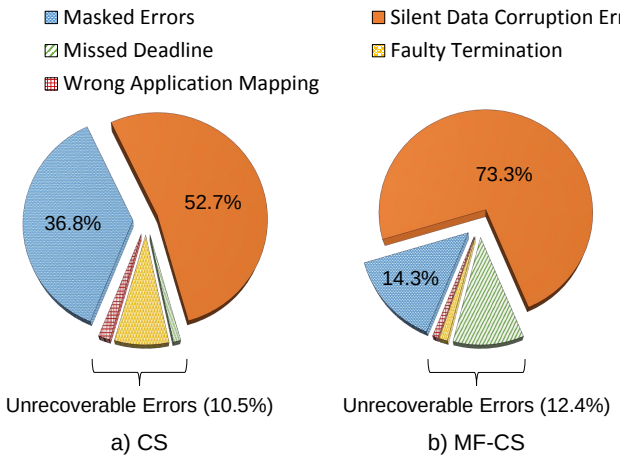

Fig. 7. Application-visible error distribution due to randomly distributed memory failures.

This article was presented in the International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS) 2017 and appears as part of the ESWEEK-TECS special issue, Vol. 0, No. 0, Article 00. Publication date: 0000.

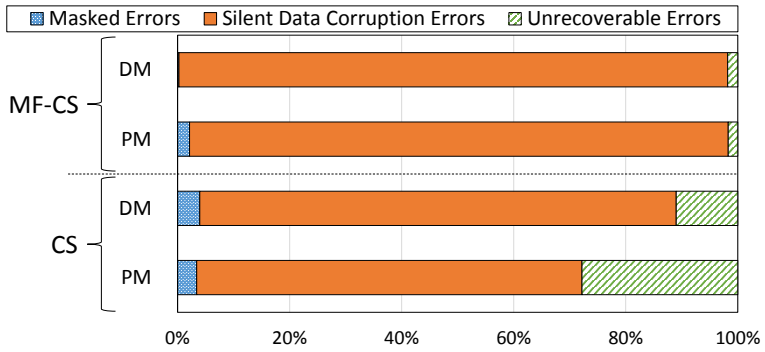

Fig. 8. Run-time error profile, considering the access frequency of memory words.

The observed error outcomes are depicted in Figure 7. In the CS application (Figure 7.a), about a third of the simulations resulted in masked errors and hence have no application-visible effect. A little more than half of them produced SDC errors, while the rest of the error injection tests created unrecoverable errors, preventing the system from recovering to a correct execution state. Among these unrecoverable errors, a majority of memory failures lead to a jump instruction to an unused IM location. The rest are either failures that make the processor run into an infinite (or extremely long) loop, ultimately missing the execution deadline, or those which cause a wrong application mapping (i.e.: when a task is not assigned to the correct core). We observed a similar error breakdown for the MF-CS application (Figure 7.b). For this benchmark, SDC errors dominate, while masked errors are substantially reduced compared to the CS case. The contribution of unrecoverable errors remain largely unaltered, among which errors that trap the cores in a hanged state dominate, while other unrecoverable error sources are less prominent than in CS. In both benchmarks, the error profiles offer vast opportunities for the inexact error management scheme, dependent on the failure criticality, as opposed to state-of-the-art error correction strategies that equally protect each memory location.

Since different memory locations have different access frequencies, as a second step in our failure characterization we collected statistics on the access count of each memory word, weighting the error profile of the DM and IM accordingly. In this way, we computed statistics on the likely result of a bit-flip at run-time, which are summarized in Figure 8. The experimental evidence shows a high prevalence of unrecoverable and (especially) SDC errors with respect to masked ones. In particular, for CS, while the ratio of unrecoverable errors is non-negligible, the impact of bit-flips causing masked errors are reduced when the number of memory accesses are considered. MF-CS, instead, presents a diminished ratio for both masked and unrecoverable errors, while the impact of SDC errors are even more dominant than in the unweighted distribution.

6.2 Effectiveness of The Error Recovery Strategy

In a further round of experiments, we inspected the impact of errors on the availability of the system, and the overall impact on the acquired signal integrity, when the error mitigation strategies described in Section 4 are applied. As expected, all unrecoverable errors are countered by the Execution Monitor, resulting in orderly resets with no side effects after the resumption of execution. In addition, the platform is able to detect 96.4% and 91.8% of SDC errors for CS and MF-CS applications, respectively, by checking the state of a few critical system elements (synchronization events, FIFOs state, IM and DM accesses).

This article was presented in the International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS) 2017 and appears as part of the ESWEEK-TECS special issue, Vol. 0, No. 0, Article 00. Publication date: 0000.

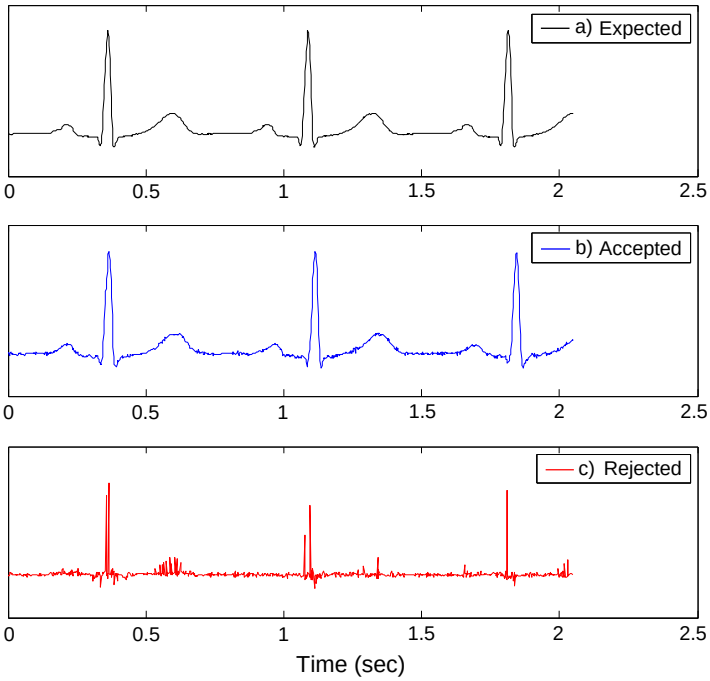

Fig. 9. Example of signals impacted by SDC errors relative to an error-free ECG.

Finally, the majority of incorrect outputs caused by SDC errors, which are not identified by the Execution Monitor on the WBSN, can be effectively filtered using receiver-side integrity checks (Section 4.2-*e*). In particular, highly corrupted signals (Figure 9.c), are rejected by the classification strategy while, on the other hand, the signals which are accepted despite SDC errors (e.g. Figure 9.b) show a high correlation with the expected ECG 9.a). Data integrity checks were able to identify 82.4% of the SDC errors which escaped detection by the Execution Monitor.

6.3 Area Overhead

The Execution Monitor comprises different components: a single register bank tracks when a processor enters or exits a synchronization point, while a stack for each core checks the legality of the synchronization instructions being issued. Moreover, hardware counters monitor FIFOs for

Component	**Area/element (μm^2)**	**# of elements**
Processor	16 969	8
DM bank	69 366	16
IM bank	98 726	8
Crossbar	17 430	3
Synchronizer	6 302	1
Execution Monitor	*25 684*	*1*
Total area (μm^2)	**2 119 711**	-

Table 1. Area of the multi-core platform and of its architectural components.

This article was presented in the International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS) 2017 and appears as part of the ESWEEK-TECS special issue, Vol. 0, No. 0, Article 00. Publication date: 0000.

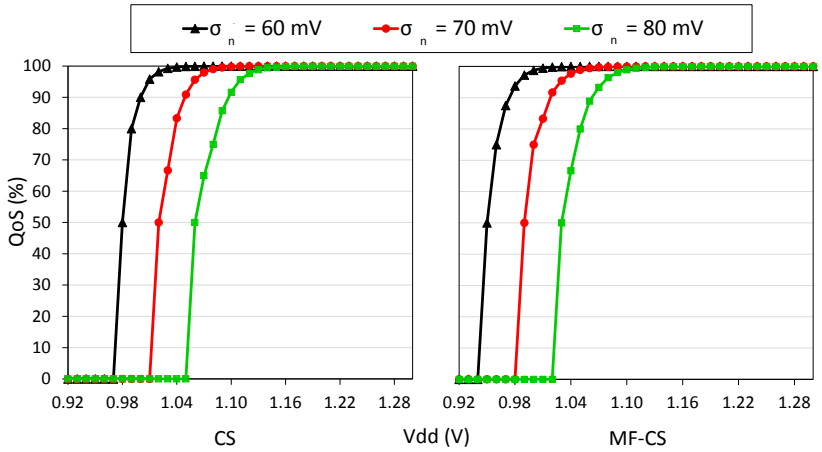

Fig. 10. Evolution of the Quality-of-Service for different voltage supply values and droop variations.

possible overflows, and comparators ensure that instruction and data addresses fall within valid IM/DM ranges.

Post-synthesis data (shown in Table 1) highlights that the silicon real estate required to implement these architectural elements (indicated in the penultimate row) results in a really limited overhead of 1.2% of the total system area, which is comparable to the one required by a single crossbar. For comparison, an alternative strategy in which each program and data memory word is protected by single Error Correction Code (ECC), would incur a 12x bigger area overhead, even when just the extra storage to accommodate the ECC bits is considered. Moreover, an ECC-based strategy would also require non-negligible energy costs to perform parity calculations on every memory access, which are not required in our implementation.

6.4 Efficiency/Quality of Service Trade-off

As a metric for the expected Quality-of-Service (QoS) of the inexact system, we measured the ratio between correct and discarded data. Figure 10 shows the QoS degradation depending on the employed supply voltage. We have pessimistically assumed that any detected error, either by the WBSN or by the receiver, results in the rejection of the full window of samples being processed (i.e. a block of 1024 samples representing 2 seconds of the ECG signal). As it can be observed, the presence of bit-flips does not translate into tangible reductions of the QoS until the voltage is reduced close to 1.05V (for the case of $\sigma_n = 70mV$).

For comparison, we set the supply voltage that guarantees an exact execution to be 1.27V, which is sufficient to suffer less than one bit-flip per year. A QoS degradation of 10% therefore allows a supply reduction to 1.02V (-20.8%) for MF-CS. At this supply level, the platform is able to handle more than 1700 bit-flips per hour and the QoS is guaranteed to be over 90%.

To assess the optimal point at which the voltage supply of the platform should be set-up, in Figure 11, we show the average energy consumption per valid sample (i.e. samples of windows that have not been discarded either on-board or on the receiver side). As it can be observed, while considering $\sigma_n = 70mV$, the optimal point corresponds to a Vdd of 1.07V and 1.04V for CS and MF-CS, bringing power savings of 28.6% and 32.3%, respectively. In both cases the QoS degradation is below 3%, as shown in Figure 10, and the expected error occurrence rate due to the voltage under-supply is 240 bit-flips per hour for CS and 900 bit-flips per hour for MF-CS.

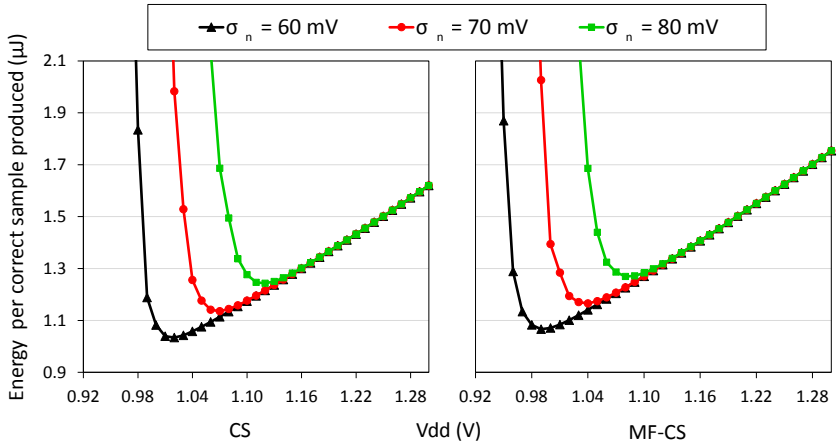

Fig. 11. Evolution of the energy consumption per correct output (compressed sample) produced, for different voltage supply values and droop variations.

Finally, Figure 12 shows the overall power consumption of the multi-core system supplied with a voltage level that ensures error-free execution (1.27V) and the one incorporating the proposed Execution Monitor supplied with the optimal Vdd (1.13V) that guarantees a QoS of more than 99%, which is the standard for medical devices, for the considered applications. This figure shows that, while the consumption of the Execution Monitor is not negligible in relative terms (up to 5.7%), the absolute reduction of the system consumption reaches more than 20% for CS and up to 24% for the MF-CS application.

7 CONCLUSIONS

Traditionally, best practices in digital system design are based on a stack of abstraction layers, separating concerns at the lower (technology) and higher (application) levels. Abstraction is crucial when approaching complex endeavors such as the design of modern Systems-on-Chips. On the

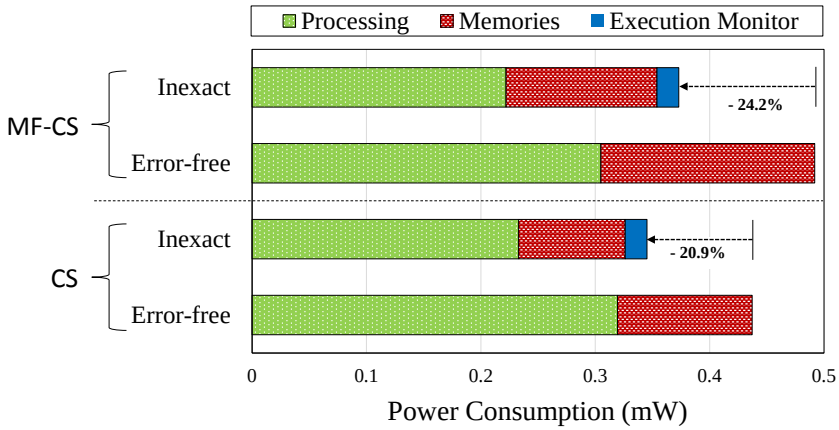

Fig. 12. Power consumption comparison of the error-free multi-core system (operating at 1.27V) with respect to the proposed inexact system (operating at 1.13V), which guarantees a QoS of over 99% for both applications.

This article was presented in the International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS) 2017 and appears as part of the ESWEEK-TECS special issue, Vol. 0, No. 0, Article 00. Publication date: 0000.

other hand, the interface between layers should also include some degrees of flexibility, as a poor choice may hide important optimization opportunities that only emerge from cross-layer interactions. We advocate that the exploration of the trade-offs between efficiency and reliability for ultra-low power systems requires such holistic view.

Following this intuition, we jointly investigated the error tolerance of applications and the power/reliability trade-off exposed by digital circuits. The outcome of our analysis was then leveraged to devise a multi-core architecture supporting inexact execution, and its associated run-time model. We showed that a 100% correctness guarantee is not required to avoid permanent system-level failures. Moreover, targeting the concrete scenario of embedded bio-signal analysis applications, we highlighted that high failure rates can be tolerated, while minimally degrading the application-level QoS. Finally, we observed energy efficiency gains, due to voltage over-scaling, of more than 24% with respect to an equivalent, but failure-free alternative.

Our findings establish the benefits of the inexact computing paradigm for the design of ultra-low power WBSN architectures. They also open further research avenues, ranging from the applicability of our methodology to workloads beyond bio-signal processing, to the design of only "good enough" power delivery networks devoted to inexact systems.

REFERENCES

[1] H. Alemdar et al. 2010. Wireless sensor networks for healthcare: A survey. *Computer Networks* 54, 15 (October 2010), 2688–2710.

[2] M. Ashouei et al. 2011. A voltage-scalable biomedical signal processor running ECG using 13pJ/cycle at 1MHz and 0.4 V. In *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2011 IEEE International*. IEEE, 332–334.

[3] L. Atzori et al. 2010. The internet of things: A survey. *Computer networks* 54, 15 (October 2010), 2787–2805.

[4] S. Borkar et al. 2003. Parameter variations and impact on circuits and microarchitecture. In *Proceedings of the 40th annual Design Automation Conference*. ACM, 338–342.

[5] D. Bortolotti et al. 2014. Approximate compressed sensing: ultra-low power biosignal processing via aggressive voltage scaling on a hybrid memory multi-core processor. In *Proceedings of the 2014 international symposium on Low power electronics and design*. ACM, 45–50.

[6] R. Braojos et al. 2014. Hardware/software approach for code synchronization in low-power multi-core sensor nodes. In *Design, Automation and Test in Europe Conference and Exhibition (DATE), 2014*. IEEE, 1–6.

[7] R. Braojos et al. 2014. Ultra-low power design of wearable cardiac monitoring systems. In *Proceedings of the 51st Annual Design Automation Conference*. ACM, 1–6.

[8] R. Braojos et al. 2016. A synchronization-based hybrid-memory multi-core architecture for energy-efficient biomedical signal processing. *IEEE Trans. Comput.* (September 2016).

[9] J. Constantin et al. 2012. TamaRISC-CS: An ultra-low power application-specific processor for compressed sensing. In *VLSI and System-on-Chip (VLSI-SoC), 2012 IEEE/IFIP 20th International Conference on*. IEEE, 159–164.

[10] R. H. Dennard et al. 1974. Design of ion-implanted MOSFET's with very small physical dimensions. *IEEE Journal of Solid-State Circuits* 9, 5 (January 1974), 256–268.

[11] A. Dogan et al. 2013. Synchronizing code execution on ultra-low power embedded multi-channel signal analysis platforms. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2013*. IEEE, 396–399.

[12] R. G. Dreslinski et al. 2010. Near-threshold computing: Reclaiming moore's law through energy efficient integrated circuits. *Proc. IEEE* 98, 2 (January 2010), 253–266.

[13] H. Esmaeilzadeh et al. 2012. Architecture support for disciplined approximate programming. In *ACM SIGPLAN Notices*, Vol. 47. ACM, 301–312.

[14] EU-MEP. 2015. Cardiovascular diseases facts and figures. www.mepheartgroup.eu/index.php/facts-a-figures. (2015).

[15] P. Ghanta et al. 2005. Stochastic power grid analysis considering process variations. In *Proceedings of the conference on Design, Automation and Test in Europe-Volume 2*. IEEE Computer Society, 964–969.

[16] C. Gomez et al. 2010. Wireless home automation networks: A survey of architectures and technologies. *IEEE Communications Magazine* 48, 6 (May 2010), 92–101.

[17] P. Gupta et al. 2013. Underdesigned and Opportunistic Computing in Presence of Hardware Variability. *Trans. Comp.-Aided Des. Integ. Cir. Sys.* 32, 1 (Jan. 2013), 8–23.

[18] Y. Hao et al. 2008. Wireless body sensor networks for health-monitoring applications. *Physiological measurement* 29, 11 (October 2008), R27.

This article was presented in the International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS) 2017 and appears as part of the ESWEEK-TECS special issue, Vol. 0, No. 0, Article 00. Publication date: 0000.

[19] Y. He et al. 2010. Xetal-pro: an ultra-low energy and high throughput SIMD processor. In *Proceedings of the 47th Design Automation Conference*. ACM, 543–548.
[20] K. J. Heilman et al. 2007. Accuracy of the LifeShirt®(Vivometrics) in the detection of cardiac rhythms. *Biological psychology* 75, 3 (July 2007), 300–305.
[21] Texas Instruments. 2013. 2.4-GHz Bluetooth® low energy System-on-Chip. www.ti.com/lit/ds/symlink/cc2540.pdf. (June 2013).
[22] ITRS. 2016. International Technology Roadmap for Semiconductors. www.itrs2.net/. (2016).
[23] U. R. Karpuzcu et al. 2012. VARIUS-NTV: A microarchitectural model to capture the increased sensitivity of manycores to process variations at near-threshold voltages. In *Dependable Systems and Networks (DSN), 2012 42nd Annual IEEE/IFIP International Conference on*. IEEE, 1–11.
[24] S. Khare et al. 2013. Prospects of near-threshold voltage design for green computing. In *VLSI Design and 2013 12th International Conference on Embedded Systems (VLSID), 2013 26th International Conference on*. IEEE, 120–124.
[25] P. K. Krause et al. 2011. Adaptive voltage over-scaling for resilient applications. In *2011 Design, Automation Test in Europe*. 1–6.
[26] J. Kwong et al. 2011. An energy-efficient biomedical signal processing platform. *IEEE Journal of Solid-State Circuits* 46, 7 (June 2011), 1742–1753.
[27] X. Li et al. 2007. Application-Level Correctness and Its Impact on Fault Tolerance. In *Proceedings of the 2007 IEEE 13th International Symposium on High Performance Computer Architecture (HPCA '07)*. IEEE Computer Society, Washington, DC, USA, 181–192.
[28] J. W. S. Liu et al. 1994. Imprecise computations. *Proc. IEEE* 82, 1 (Jan 1994), 83–94.
[29] S. Lobodzinski. 2013. ECG patch monitors for assessment of cardiac rhythm abnormalities. *Progress in cardiovascular diseases* 56, 2 (September 2013), 224–229.
[30] A. Mainwaring et al. 2002. Wireless sensor networks for habitat monitoring. In *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*. Acm, 88–97.
[31] W. K. Mak et al. 2007. Voltage island generation under performance requirement for SoC designs. In *Proceedings of the 2007 Asia and South Pacific Design Automation Conference*. IEEE Computer Society, 798–803.
[32] H. Mamaghanian et al. 2011. Compressed sensing for real-time energy-efficient ECG compression on wireless body sensor nodes. *IEEE Transactions on Biomedical Engineering* 58, 9 (May 2011), 2456–2466.
[33] S. Mittal. 2016. A survey of architectural techniques for near-threshold computing. *ACM Journal on Emerging Technologies in Computing Systems (JETC)* 12, 4 (July 2016), 46.
[34] S. Mittal. 2016. A survey of techniques for approximate computing. *ACM Computing Surveys (CSUR)* 48, 4 (May 2016), 62.
[35] Y. Morita et al. 2007. An area-conscious low-voltage-oriented 8T-SRAM design under DVS environment. In *VLSI Circuits, 2007 IEEE Symposium on*. IEEE, 256–257.
[36] S. S. Mukherjee et al. 2005. The Soft Error Problem: An Architectural Perspective. In *Proceedings of the 11th International Symposium on High-Performance Computer Architecture (HPCA '05)*. IEEE Computer Society, Washington, DC, USA, 243–247.
[37] S. Mukhopadhyay et al. 2005. Modeling of failure probability and statistical design of SRAM array for yield enhancement in nanoscaled CMOS. *IEEE transactions on computer-aided design of integrated circuits and systems* 24, 12 (November 2005), 1859–1880.
[38] A. Pantelopoulos et al. 2010. A survey on wearable sensor-based systems for health monitoring and prognosis. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 40, 1 (January 2010), 1–12.
[39] PhysioBank. 2012. MIT-BIH Normal Sinus Rhythm Database. https://www.physionet.org/physiobank/database/nsrdb/. (February 2012).
[40] J. Schlachter et al. 2017. Design and Applications of Approximate Circuits by Gate-Level Pruning. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* (February 2017).
[41] M. Seok et al. 2008. The Phoenix Processor: A 30pW platform for sensor applications. In *VLSI Circuits, 2008 IEEE Symposium on*. IEEE, 188–189.
[42] S. R. Sridhara et al. 2011. Microwatt embedded processor platform for medical system-on-chip applications. *IEEE Journal of Solid-State Circuits* 46, 4 (February 2011), 721–730.
[43] Y. Sun et al. 2002. ECG signal conditioning by morphological filtering. *Computers in biology and medicine* 32, 6 (November 2002), 465–479.
[44] Synopsys. 2017. ASIP Designer. www.synopsys.com/dw/ipdir.php?ds=asip-designer. (2017).
[45] G. V. Varatkar et al. 2008. Error-resilient Motion Estimation Architecture. *IEEE Trans. Very Large Scale Integr. Syst.* 16, 10 (Oct. 2008), 1399–1412.
[46] WHO. 2017. The top 10 causes of death (Fact sheet no 310). www.who.int/mediacentre/factsheets/fs310/en/. (January 2017).

This article was presented in the International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS) 2017 and appears as part of the ESWEEK-TECS special issue, Vol. 0, No. 0, Article 00. Publication date: 0000.

[47] F. Zhang et al. 2012. Design of ultra-low power biopotential amplifiers for biosignal acquisition applications. *IEEE transactions on biomedical circuits and systems* 6, 4 (January 2012), 344–355.
[48] M. Zhang et al. 2006. Soft-error-rate-analysis (SERA) methodology. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 25, 10 (August 2006), 2140–2155.
[49] X. Zhang et al. 2013. Characterizing and evaluating voltage noise in multi-core near-threshold processors. In *Proceedings of the 2013 International Symposium on Low Power Electronics and Design*. IEEE Press, 82–87.

Received XXXX; revised XXXX; accepted XXXX