A 4μW Low-Power Audio Processor System for Real-Time Jaw Movements Recognition in Grazing Cattle

Luciano S. Martinez-Rau[1*†], Moritz Weißbrich[2†] and Guillermo Payá-Vayá[2†]

[1*]Research Institute for Signals, Systems and Computational Intelligence, sinc(i), FICH-UNL/CONICET, Ciudad Universitaria UNL, 4to piso FICH, Santa Fe, S3000, Santa Fe, Argentina.
[2]Chair for Chip Design for Embedded Computing, Technische Universität Braunschweig, Mühlenpfordtstraße 23, 38106 Braunschweig, Germany.

*Corresponding author(s). E-mail(s): lmrau@sinc.unl.edu.ar;
Contributing authors: m.weissbrich@tu-braunschweig.de;
g.paya-vaya@tu-braunschweig.de;
†These authors contributed equally to this work.

Abstract

Precision livestock farming consists of technological tools and techniques to improve the livestock management. Proper detection and classification of jaw movement (JM) events are indispensable for the estimation of dry matter intake, detection of health problems, and flag the onset of estrus, among other information. The analysis of acoustic signals is one of the most accepted ways to monitor the feeding behavior of free-grazing cattle. Different acoustic methods have been developed for recognizing JM-events in recent years. However, their operation is limited to off-line analysis on a personal computer. The lack of on-line acoustic monitoring systems is associated with the challenging operation requirements (low-power consumption, autonomy, portability, robustness and non-intrusive on the animal). In this paper, a fixed-point variant of the chew-bite energy-based algorithm is presented. This algorithm is implemented on a new low-power audio processor system for real-time recognition of JM-events. The system includes a Nanocontroller processor, which is

always-on and detects JM-events; and a second transport-triggered architecture (TTA) based processor, which is mainly in power-down and classifies JM-events. The results demonstrate that the proposed fixed-point JM-events recognizer achieves a recognition rate of 91.4% and 90.2% in noiseless and noisy conditions, respectively. The recognition rate increases by 6.1% regarding a previous reference on-line system. Moreover, the proposed audio processor system chip consumes 4 µW on average, i.e., only 2.3% of the power of an always-on TTA-based processor system for the same audio sequence. An exemplary implementation of the proposed system in a 65 nm low-leakage CMOS technology is given.

Keywords: Acoustic monitoring, machine learning, transport-triggered architecture, nanocontroller, pattern recognition, processor architecture

1 Introduction

In recent decades, the world of the dairy industry has undergone profound changes associated with an increasing demand, and at the same time, seeking to achieve a good quality, sustainable, safe, and environmentally friendly production. This has led to a trend in dairy farms towards process automation to stay competitive in an increasingly challenging market, using precision livestock farming tools.

Precision livestock farming is based on the use and development of information and communication technologies to achieve automatic, precise, and real-time animal monitoring capabilities [1]. These tools are used to monitor animal behaviors, in order to help farmers to improve the livestock management by changing the observation scale from herds to individual animals. Individualized behavior monitoring provides important and unique information about collective activities related to heat, health, nutrition and welfare of animals [2, 3]. Thus, these technologies make it possible to optimize and control the resources of livestock production systems.

Monitoring of animal behavior has seen an increase in the emergence of new technology approaches, using video cameras based on computer and deep-learning [4, 5]. However, these solutions are only applicable on confined livestock systems. Wearable sensors are the most appropriate and adopted technology for free-ranging conditions. One of the key technological factors of wearable sensors is their ability to provide useful information in an accurate, effective, and interpretable way, allowing farmers to make the necessary decisions. These sensors have to be small, portable and non-invasive, so as not to interfere with the natural behavior of the animal. Furthermore, they have to be robust to function properly in different operating conditions, and work autonomously and continuously for long periods of time [6].

The livestock sector is increasingly presenting commercial wearable sensor solutions to monitor the feeding behavior of grazing cattle [7]. The most important feeding behaviors recognized are rumination and grazing [8]. Sensors are

based on global positioning systems [9], accelerometers [10], microphones [11], inertial measurement units [12], pressure sensors [13] or in a combination of the above [14]. However, and to the best of our knowledge, there is no wearable sensor used in commercial farming that is able to report masticatory jaw movements (JM) events associated to short-term feeding activities [7]. This information is necessary to estimate the dry matter ingested by the animal [15, 16], to monitor its diet [17] and to detect changes in the rumen pH [18] among others. Besides, the recognition of JM-events leads to estimate long-term foraging activities: rumination and grazing bouts, and rumination and grazing times, respectively [19]. In this sense, the livestock sector presents a lack of sensor solutions for monitoring the short-term feeding behavior of ruminants.

The objectives of this paper are twofold: (*i*) to design, evaluate, and develop a dedicated audio processor system to recognize masticatory JM-events in various operating conditions, implementing the most robust algorithm of the state-of-the-art, called *chew-bite energy based algorithm* (CBEBA) [20]; and (*ii*) to minimize the power consumption of the processor system by using an optimized architecture for processing the CBEBA in real-time. The main contributions of this paper are:

- Design of a dedicated audio processor system for the recognition of JM-events, which can be used in a wearable acoustic sensor.
- Design of a fixed-point variant of the CBEBA and comparison against both a floating-point reference and an acoustic sensor benchmark [21, 22].
- Analysis of the proposed audio processing system in terms of processing performance, silicon area and power consumption, using a 65 nm low-leakage CMOS technology and real audio sequences.

The rest of this paper is organized as follows: Section 2 reviews commercial sensors used for the recognition of JM-events. A brief explanation and proposed simplifications of the CBEBA algorithm to be embedded in a processor are presented in Section 3. A brief description and characteristics of different customized processor architectures chosen in this work are described in Section 4. Section 5 introduces the proposed audio processor system and describes three possible single-core processor systems used for reference. All the presented processor systems are optimized for the processing of the fixed-point variant of the CBEBA. The description of audio datasets, performance metrics, and the experimental setup used to evaluate the embedded algorithm performance, hardware performance and power consumption of the processors is conducted in Section 6. Section 7 presents and analyzes the results of the embedded algorithm and of the four different implemented processors. The main conclusions follow in Section 8.

2 Related Work

The correct recognition of JM-events is a difficult task and presents a lack of automatic solutions to perform it. This section gives an overview and current limitations of sensors and devices used to recognize JM:

- *Accelerometer-based sensors*: Accelerometers are widely used to monitor long-term feeding behaviors, but are rarely applied to short-term JM-events [23]. The algorithm of [24] recognized one class of JM-events using an accelerometer and reaching up to 24 h of autonomy. Then, another algorithm improved the recognition to three different classes of JM-events [25]. Both algorithms were used to estimate the herbage intake, but their validation in different pastures (species and height) is still required. None of these algorithms has been implemented yet in an autonomous wearable sensor. Moreover, the correct orientation, subjection and verification of the accelerometer could affect the correct operation of the sensor in practical applications [26].
- *Multiple sensors*: The commercial RumiWatchSystem (Itin and Hoch GmbH, Liestal, Switzerland) is a precise research tool for automatically measuring and gathering valid information of the feeding behavior of cows. The system includes a nose-band pressure sensor linked with a three-axial accelerometer in a halter to measure JM. An optional pedometer can be added to measure other behaviors. The raw data are saved on an integrated SD card for further processing by the specific software RumiWatch Converter (Itin and Hoch GmbH, Liestal, Switzerland), where generic algorithms are applied to analyze long- and short-term feeding behaviors in detail [27], and thus to be able to estimate the dry matter intake [28]. Since the nature of the RumiWatchSystem is a data logger rather than a wearable sensor, it is not compatible and useful in commercial farms [23].
- *Acoustic-based sensors*: The commercial HR Tag rumination monitor (SCR Engineers Ltd., Netanya, Israel) records long- and short-term rumination activities by measuring the sound with a microphone. The rumination statistical information is computed on-line for an algorithm. This information is automatically downloaded when the animal is close to a linked tag [29]. Further development of the HR Tag has to be incorporated to provide more useful and detailed information, such as the recognition of JM-events and the delimitation of feeding activities bouts. A wearable acoustic sensor to recognize JM-events of cattle was designed and implemented in [21]. The sensor was built in an embedded device with a microcontroller. The firmware has the JM-events recognizer developed by [30] (designated FP-CBRTA by its name Fixed-Point version of the Chew-Bite Real-Time Algorithm), achieving a detection and classification rate of 92% and 78% respectively, under laboratory experimental conditions. The device achieves an autonomy of more than 5 days, under controlled conditions [22]. However, this sensor presents two limitations. First, the FP-CBRTA does not show good robustness to noise and time-varying operating conditions [31].

Table 1: Description of the JM-events associated with the feeding activities

Feeding activity (long-term behavior)	Masticatory JM-events (short-term behavior)	Definition
Grazing	Bite (B)	Apprehension and severance of herbage.
	Grazing-chew (GC)	Crushing, grinding and processing of herbage during ingestion.
	Chew-bite (CB)	Combination of chewing and biting in the same JM during ingestion of grass.
Rumination	Rumination-chew (RC)	Chewing of a regurgitated ruminating bolus.

Second, since energy consumption is not optimized, large battery capacity is needed. A higher number of batteries increases the weight and size of the sensor, thus reducing the comfort of the animal. Moreover, the sensor autonomy still needs to be evaluated in real field conditions, where the animal movements, weather patterns and climate may affect the energy harvesting mechanism [32].

In this paper, the acoustic sensor of [21] is used as reference. The first limitation, presented in this sensor, is here solved using a better JM-events recognizer. The proposal to deal with the second limitation is to replace the general-purpose microcontroller with a custom processor designed to recognize JM-events.

3 JM-events Recognizer

In this section, a brief description of the recognition of JM-events by using the CBEBA algorithm and its proposed implementation for embedded systems is given, considering hardware limitations and optimization of resources.

3.1 CBEBA Algorithm Description

The CBEBA is a real-time pattern recognition algorithm to detect and classify masticatory sounds produced during feeding activities by ruminants in four JM-events described in Table 1. The algorithm is internally divided into six stages [20]:

1. *Pre-processing stage*: the input audio sampled at a frequency $f_i \geq 1800\,\text{Hz}$ is bandwidth limited with a second-order band-pass filter and multiplied by itself to obtain the instantaneous power signal. Furthermore, if the gain level applied to the microphone is available, this could be used to normalize the input audio.
2. *Buffering stage*: the instantaneous power signal is used to compute the envelope signal and the energy signal calculated by frames (frame-energy

signal). Each signal is down-sampled to $f_s = 150\,\text{Hz}$ and saved in a circular buffer which contains data corresponding to one second of audio.

3. *Detection stage*: the presence of peaks in the envelope buffer, located between the middle position and the last position, greater than an adaptive threshold value indicates the detection of a candidate JM-event. When a detection occurs, it is waited until the middle position between the peaks matches with the mean position of the buffer. Then, the start and end position of the candidate JM-event inside the buffers are delimited by comparing the frame-energy buffer with another adaptive threshold.
4. *Feature extraction stage*: the data stored in both buffers between the start and end position are used to extract a set of heuristic features.
5. *Classification stage*: the extracted features are used to decide whether the candidate event should be classified or discarded. In case of a positive decision, a multi-layer perceptron (MLP) classifies the JM-event.
6. *Threshold and parameters tuning stage*: the signals stored in both buffers are used to update the values of both adaptive thresholds. To do this, different values taken from and around the JM-event are used. This allows the thresholds to be adapted to the range of JM-events, taking into account the time-varying signal level of the noise floor.

A complete description of the CBEBA is given in [20].

3.2 CBEBA Fixed-Point Implementation

The CBEBA was coded in floating-point using Matlab R2019b (MathWorks, Natick, MA, USA). As this arithmetic requires dedicated hardware in the processor, which increases its area and its energy consumption, a fixed-point representation of the CBEBA was implemented, hereafter called FP-CBEBA. By using fixed-point format analysis in Matlab, each variable was adjusted individually to store more than 95% of the range obtained by CBEBA in floating-point, without losing sight of the operating principle and the performance of the JM-events recognizer. The stage modified are:

1. *Pre-processing stage*: The second-order band-pass filter can be seen as a first-order high-pass filter cascaded with a first-order low-pass filter. The first-order low-pass filter was avoided in the FP-CBEBA with the aim to achieve reduced computation, but at the cost of not filtering low frequency noise and trend signals. However, this should negligibly affect the JM-events detection and classification due to the adaptive property of both threshold algorithms.
2. *Buffering stage*: The sample rate of the envelope and energy signals was modified from $f_s = 150\,\text{Hz}$ to $f_s = 98\,\text{Hz}$, reducing one third the operating frequency of a great part of the algorithm at the cost of reduced recognition rate.
3. *Detection stage*: Instead of looking for the envelope peaks at a particular position in the envelope buffer, the detection task is performed in real-time, i.e., the envelope peaks are searched at the same time that the data

are stored in the memory. This allows a hardware implementation of the envelope buffer in a non-addressable memory, like a first-in-first-out buffer (FIFO).

4. *Classification stage*: The MLP classifier is an artificial neural network that internally performs multiplications, divisions and exponential operations, among others. The MLP is replaced with a decision tree (DT) classifier in order to reduce hardware requirements and computation. A DT performs only comparison operations and divides the decision boundary linearly, unlike the MLP, which can deal with non-linear data [33]. The DT was trained in the floating-point CBEBA and adjusted to fixed-point format, as previously described.

4 Processor Architecture Organizations

The performance and hardware efficiency of a processor designed for a specific application can be increased by applying the application-specific instruction-set processor (ASIP) concept. Thus, two fully customized processor architectures are considered. These are a minimal processor for ultra-low-power programmable system state controllers, called Nanocontroller [34, 35], and a transport-triggered architecture (TTA) [36]. Both architectures allow a complete customization of the data flow and operations required by a target application.

4.1 Nanocontroller Architecture

The Nanocontroller processor architecture [35] is based on the concept of one-operand accumulator architectures. It has been designed for the purpose of programmable system state and power management control in the always-on power domain of an ASIC or System-on-Chip (SoC), which contains multiple processor cores in multiple power domains. Therefore, the primary design goals of the Nanocontroller architecture are minimum silicon area and leakage power consumption in order to reduce the overall always-on power consumption of a system. While Nanocontroller processes frequent tasks of low complexity, like audio sample pre-processing and JM-events detection in the FP-CBEBA, and continuously controls the system state and power management, other system components are supposed to be power-gated to the off-state by default. In case of required processing of infrequent, more complex tasks, like JM-events classification in FP-CBEBA, a separate processor will be temporarily switched on. The intelligent power gating of the system domains is under full program control of the Nanocontroller.

The Nanocontroller instruction-set architecture (ISA) uses only 16 compact, control-oriented instructions encoded in 4-bit opcodes. This minimal ISA consists of basic arithmetic, data memory load/store, single-instruction increment/decrement, comparison and conditional branch operations, which have been selected to support memory-efficient software implementation of finite

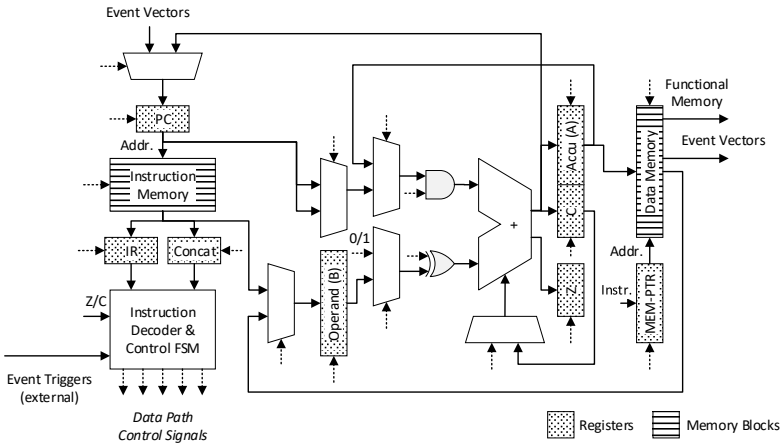

Fig. 1: Schematic block diagram of the Nanocontroller architecture [35].

state machines (FSMs) for system state control and power management strategies. Variable-length encoding of literal values is applied in order to reduce the size of immediate values and memory addresses in the code, i.e., the architecture uses *variable-length instruction encoding*. Due to these techniques and compared to general-purpose microcontrollers, both the number of required instructions and the code size are reduced, e.g., 61% smaller code size compared to the PIC12 ISA is achieved in the Nanocontroller-based system described in [34], while still maintaining a flexible and fully programmable controller for FSMs. Consequently, the silicon area and the power consumption of the instruction memory are significantly reduced as well.

The schematic block diagram of the Nanocontroller architecture is depicted in Fig. 1. In order to execute the instructions of the minimal ISA, only an adder unit is required as an arithmetic circuit. A single adder is used in multi-cycle execution to update both accumulator `A` and the program counter `PC` in order to reduce the amount of logic components (resource sharing). Several multiplexers, `AND` and `XOR` masking extend the adder with subtraction, increment, decrement and clearing functionality. Zero and carry flag registers `Z` and `C` are updated implicitly for conditional branches. For reduced code size, parts of the data memory are used as functional memory and for event vectors. Here, the term *functional memory* describes a memory location that performs implicit actions when updated, e.g., toggles output pins or generates enable signals according to a certain system state. By that, dedicated I/O instructions can be eliminated. *Event vectors* directly point to a code location, which is executed when an internal or external event trigger is received, e.g., when a new audio sample is ready to be processed. This way, code overhead of software-based interrupt handlers is eliminated. For further reference, an extensive and detailed description and the evaluation of the Nanocontroller architecture and ISA are presented in [35].

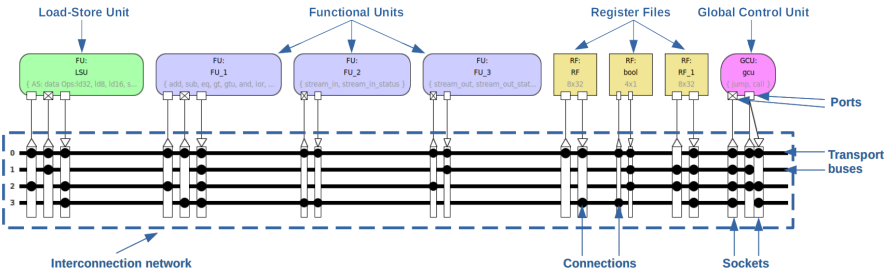

Fig. 2: Internal parts of a TTA processor.

As a final remark, it should be emphasized that the architectural concept of Nanocontroller is not restricted to control tasks by design, but is prepared for specialization with additional processing units. In this work, a parameterizable and programmable filter co-processor for digital audio pre-processing is added to the always-on domain in order to establish a tiny ASIP for the processing requirements of the FP-CBEBA. The proposed processing platform and monitoring system will be described in Section 5.

4.2 Transport-Triggered Architecture

The TTA, can be described as a very long instruction word (VLIW) architecture, in which internal data paths are exposed to the instruction set [37]. A TTA processor consists of different register files (RFs) and functional units (FUs). Each FU can be customized, implementing one or more computational operations internally. The FUs and RFs are connected through a programmable interconnection of transport buses, creating programmable data paths. While in most processor architectures the instruction word specifies the computational operation to be performed, in a TTA it indicates how the data should be moved between the FUs through the different buses. The operands are transferred to a FU through programmable sockets, linking a FU to the buses using connections. The computational operation of a FU is triggered as a side effect of writing data on it. A special FU that requires access to data memory or communicates outside the processor is called load-store unit (LSU). The FU that controls the execution of the program flow and performs jumps and function calls by accessing the program memory is called the global control unit (GCU). The basic FUs required to form a TTA are the GCU, the LSU, a 32-bit wide RF, a 1-bit wide Boolean RF and the arithmetic-logic unit (ALU). An example of TTA marking its internal parts is shown in Fig. 2.

The configurable architecture allows adding new FUs with customized functionality and computational operations, modifying the number of buses to allow parallel data transfer, and selectively configuring the interconnection of the sockets with the buses, considering a compromise between programming flexibility and complexity of interconnection. The configuration, customization and design space exploration process during the design of a specific TTA can

be realized using an open-source TTA HW/SW co-design environment (TCE) tools including an LLVM-based C compiler [36].

Register bypassing allows data to be routed from one unit to another without going through the RF. Register bypassing can be done by the software programmer (or the compiler) modifying the programmer-visible interconnection network. FUs may include additional internal state registers, which cannot be directly accessed via the TTA programming model, to implement complex computational operations. Due to the registered FUs and register bypassing, a RF is not mandatory in a TTA processor, but it is commonly incorporated into the architecture to temporarily store frequently accessed variables and results.

Using techniques like either register bypassing or multiple FU ports receiving common operands simultaneously from a single source (operand sharing), the RF port pressure is reduced. Therefore, this flexible configurable architecture makes it possible to reduce circuit complexity and to improve the utilization efficiency of resources, by merging hardware resources, such as buses and RF ports, that are rarely or never used in parallel [38].

The encoding of immediate operands on each bus is shortened to reduce the instruction word width, which depends on the total number of buses. If it is required, an immediate unit (IU) can be used to combine multiple short immediate values from several buses to form a long immediate value, e.g., a 32-bit immediate value [39].

5 Proposed Monitoring System

In the following section, the selected processor architectures that are programmed with the FP-CBEBA are presented. A particular processor takes as inputs the digital audio in parallel format and, if it is available, the gain of the acquired system. The digital audio samples come from either a MEMS or electret microphone. The output of the processor is serial data containing the JM-event classified and a corresponding timestamp.

5.1 Proposed Audio Processor System Architecture

As mentioned above, the FP-CBEBA is internally divided into six stages that perform different tasks. These six stage could be grouped into two parts, in which the first part operates continuously, while the second part operates only when a candidate JM-event is detected. With this in mind, it is possible to implement the first part of the FP-CBEBA using an always-on Nanocontroller processor, while the second part is implemented in a TTA processor that will be powered down by default. In this way, using these two processor cores together with different memories to pass data between them, it is possible to implement a system that exploits the advantages of each processor architecture in terms of processing performance and energy consumption.

Fig. 3 shows a block diagram of the proposed audio processor system. The functional blocks of the system have been designed following the properties of the FP-CBEBA, which can be partitioned into continuous *always-on processing*

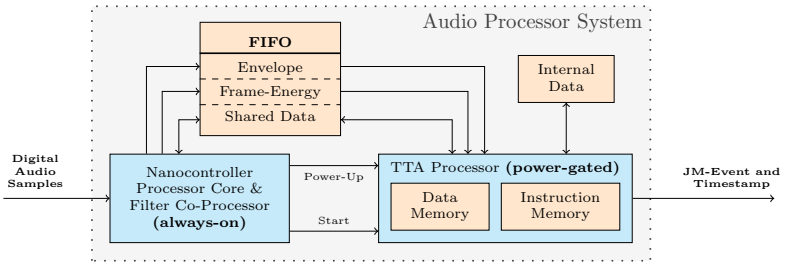

Fig. 3: Simplified block diagram of the audio processor system consisting of a Nanocontroller (core & filter co-processor) and a TTA. Blue blocks indicate processors and orange blocks represent memories.

and intermittent *JM-event-triggered processing*. The Nanocontroller processor core in the always-on domain works as master and the power-gated TTA as slave, and a FIFO memory serves to communicate data between them. Whereas at the beginning the TTA is powered down, the Nanocontroller reads the digital audio samples, computes the pre-processing and buffering stages with the support of a programmable filter co-processor, stores both downsampled envelope and frame-energy values in the FIFO memory and detects a candidate JM-event. Once a candidate JM-event has been detected, the Nanocontroller core sends, at the right time, a *power-up* signal and a *start* signal to the TTA. The *power-up* signal serves to turn the TTA on and must be sent before receiving the *start* signal by considering the necessary time to boot the TTA program from the instruction memory and to read the current state of its internal parameters from the internal data memory. Then, when the JM-event is centered on the FIFO, the Nanocontroller writes into the FIFO the current state of some internal data shared by the two processors and sends the *start* signal to enable the TTA execution.

When this occurs, the TTA reads the FIFO and executes the rest of the stages of the FP-CBEBA, that is, delimiting the start and end of the candidate JM-event, extracting its features, determining if it really is considered a JM-event, and if so, classifying it and writing its class together with a timestamp on the output of the TTA processor. Subsequently, the shared data received from the Nanocontroller through the FIFO are used to tune values of the thresholds and associated with internal parameters. These tuned values are partially stored in the internal data memory and communicated to the Nanocontroller via the FIFO. For all shared data values, a width of 16 bit is sufficient for the operation of the FP-CBEBA and is used for both the FIFO and internal data memory width. The updated shared data values are used for the Nanocontroller to detect new candidate JM-events. Finally, the TTA is powered down.

The core of the TTA is similar to a minimum configuration and is shown in Fig. 4. It uses one transport bus, one ALU, one 16 x 32 bit RF, one 2 x 1 bit RF, one IU, one LSU, one GCU and three customized FU required

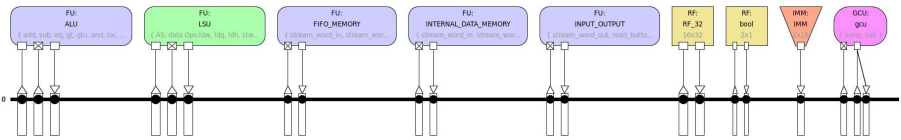

Fig. 4: Internal functional units and interconnection network of the *Slave-TTA* used in combination with the Nanocontroller core & filter co-processor.

Table 2: Comparative characteristics of different TTA core configuration.

TTA core	Instruction bitwidth [bits]	Instruction number	Instruction memory [bits]	Data memory [bits]	Minimum clock frequency [MHz]
Single-TTA-1	17	4352	73984	1024	51.68
Single-TTA-2	33	2560	84480	1024	29.43
Single-TTA-3	30	2816	84480	1024	33.24
Slave-TTA	17	4352	73984	1024	1.79

for external communications. Internally, the TTA uses a data width of 32 bit for the transport bus, ALU, RF, IU and LSU due to C compiler requirements. However, for the 3 customized FUs, a data width of only 16 bit is required to interface the FIFO and internal data memory described above. The first customized FU (*INPUT_OUTPUT*) reads the *start* signal and writes the classified JM-event with the associated timestamp. The second customized FU (*FIFO_MEMORY*) reads and writes the FIFO. The third customized FU (*INTERNAL_DATA_MEMORY*) reads and writes the internal data memory. This TTA is called *Slave-TTA* and the principal characteristics indicating the required memory sizes and the minimum operating frequency are presented in Table 2.

In Fig. 5, the sub-system in the always-on domain is depicted. The Nanocontroller processor core, which has been described in Section 4, is extended with a filter co-processor in order to support the continuous audio sample pre-processing of FP-CBEBA. Due to its design-time parameterizable hardware description and its run-time programmability, the filter co-processor can be generally applied to any FIR and IIR filtering schemes. All necessary multiplication and accumulation (MAC) steps are performed sequentially, with one MAC operation each cycle. Co-processor operation is conducted by a programmable, microcoded control unit. The number of filter taps and processable signals is only limited by the amount of memory resources and the implementation bit width of the filter data path, which are generic design parameters and have been selected according to the FP-CBEBA fixed-point analysis in this work. New input samples enter a shift register bank, where current and previous signal values, as well as previous filtering results, are stored and used to compute new filtering results. For the first-order high-pass filter and the

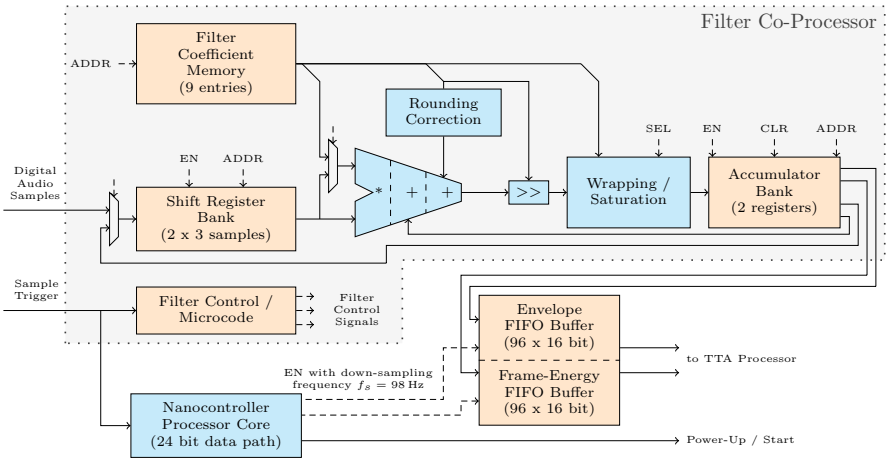

Fig. 5: Simplified block diagram of the always-on domain (filter co-processor & Nanocontroller processor core). Blue blocks indicate processing resources and orange blocks represent memories. Numeric descriptions for memory depths and data widths denote the selected parameterization of the generic data path descriptions, adapted to the requirements of the FP-CBEBA implementation in this paper.

low-pass envelope filter of the FP-CBEBA, 2 shift registers are implemented with a depth of 3 samples each and data widths of 16 and 25 bit, respectively. A MAC unit performs multiplication of the samples with the 20 bit wide filter coefficients, which have been stored by the user in a programmable memory. After the MAC operation, fixed-point format correction is performed by a shift-right unit. To reduce errors in the filter computation, a rounding correction term is added before bit truncation. Furthermore, depending on the desired characteristics, either wrapping or saturation is programmed in case of numeric range overflow. The final result of any filter operation is stored in one of the registers of the accumulator bank, which has been configured to 2 registers for FP-CBEBA with data widths of 43 and 27 bit. Via enable signals generated by the Nanocontroller application at the frequency f_s, the 16 bit wide envelope and frame-energy FIFO buffers are updated in order to be evaluated by the classification running on the powered-up TTA in case of detected JM-events.

As soon as the FIFO buffers have been updated with a new value, JM-event detection is performed on the Nanocontroller processor. For this, the buffer update control, the detection stage of the FP-CBEBA algorithm and possible power-up of the TTA processor have been manually implemented as application code in Nanocontroller assembly language. Compared to the description in previous work [35], the data path width of the Nanocontroller core has been set to 24-bit in order to adapt to the required resolution. Furthermore, following the ASIP concept, the instruction set has been slightly

Table 3: Nanocontroller instruction set for the proposed monitoring system. Modifications compared to [35] are in bold font.

Class	*Mnemonic*	*Description*
Arith-metic	`SUB`	**SUBtract data memory from accumulator**
	`CMP`	**CoMPare data memory to accumulator, set flags**
	`CMPI`	CoMPare accumulator with Immediate, set flags
Load/ Store	`LDI`	LoaD Immediate into accumulator
	`LD`	LoaD from data memory into accumulator
	`ST`	STore accumulator to data memory
	`STL`	- at Last used address
Single Instr. Memory Modify	`LIS`	Load-Increment-Store data memory
	`LISL`	- at Last used address
	`LDS`	Load-Decrement-Store data memory
	`LSRAS`	**Load, Shift Right by one bit and Accumulate, Store**
	`CST`	Clear accumulator, STore to data memory
	`CSTL`	- at Last used address
Branch	`BNE`	read flags, Branch if Not Equal zero
	`BNC`	**read flags, Branch if No Carry set**
Sleep	`SLEEP`	SLEEP until next wake-up trigger

customized for the detection stage of FP-CBEBA. In Table 3, the performed modifications are highlighted. Subtraction (`SUB`) and comparison (`CMP`) of direct addressed values in functional/data memory are used for the necessary operations of detection threshold adaptation. An alternative branch operation on no carry set condition (`BNC`) is applied in the state transitions of the envelope peak identification. These 3 instructions can be performed with the existing resources and require no new data path components. The combined `LSRAS` instruction performs the operation `MEM[addr]` ← `ACCU + (MEM[addr] >> 1)`, which is used in post-classification correction of the envelope detection threshold. Moreover, this instruction is also used in the application as an ordinary single-bit right-shift with the special condition `ACCU = 0`. The single-bit right-shift operation is enabled by adding one additional multiplexer following the operand B (`OPB`) register to the data path. Utilizing these instruction-set modifications, the Nanocontroller program consists of 177 instructions in total, which occupy 219.5 Bytes of instruction memory when translated to binary code (∅ 9.92 bit/instruction, where 4 bit account for the opcode and ∅ 5.92 bit/instruction account for variable-length address or immediate value encoding).

5.2 Reference Always-on Single-TTA-based Processor Systems

As previously described, the proposed audio processor system uses multiple specialized architectures, i.e., Nanocontroller, filter co-processor and power-gated *Slave-TTA*, to exploit the execution properties of the CBEBA algorithm. An energy efficiency comparison between the proposed audio processor system and the acoustic sensor based on microcontroller used for reference is not possible, because the microcontroller performs additional tasks, such as handling communications with other modules. In order to compare the energy efficiency of the proposed audio processor system to a conventional system implementation using a single processor core, such reference processor systems are defined in the following. The single processor needs to provide enough performance to compute the feature-based JM-event classification in real-time between two input audio samples. Therefore, from the selected set of processor architectures, single-TTA-based processors are considered. It must be emphasized that the whole FP-CBEBA will then be implemented and executed on a single TTA, which includes the continuous pre-processing, buffering and JM-event detection, as well as the intermittent JM-event classification. Due to continuous operation, the processor has to be always-on and no power gating is used. The clock gating technique is applied to reduce the dynamic power dissipation between consecutive inputs audio samples (once the TTA is just waiting for the next audio sample) [40].

There are three reference single-core TTA processors proposed, called *Single-TTA-1*, *Single-TTA-2*, and *Single-TTA-3*, and shown in Fig. 6. *Single-TTA-1* is the equivalent TTA core to *Slave-TTA* and serves to evaluate the increase in the instruction length program memory using similar TTA hardware. The FU *AUDIO_IO* reads the audio samples and writes the classified JM-events with their timestamp. Originating from this configuration, the profiling features of the TCE framework [36] have been used to identify possible instruction-level parallelism in the computational intense parts of the FP-CBEBA. Based on that, FU and bus resources have been added to the other two TTA configurations in order to reduce the execution cycle count and scheduled code length, which is expected to implicitly result in reduced power consumption for a constant workload, due to less computation cycles and instruction memory access cycles. *Single-TTA-2* has two fully-connected buses with an additional specific ALU with reduced operation instruction set (`ADD` and `SUB`) that works in parallel with the principal ALU to reduce the instruction length program memory at the cost of adding additional hardware resources. Additionally, the single 16 x 32 bit RF is separated in two 8 x 32 bit RF. *Single-TTA-3* has two partially-connected buses, reducing the instruction bit width of *Single-TTA-2* by connecting the least used sockets to a single bus. The main characteristics of the processors are given in Table 2.

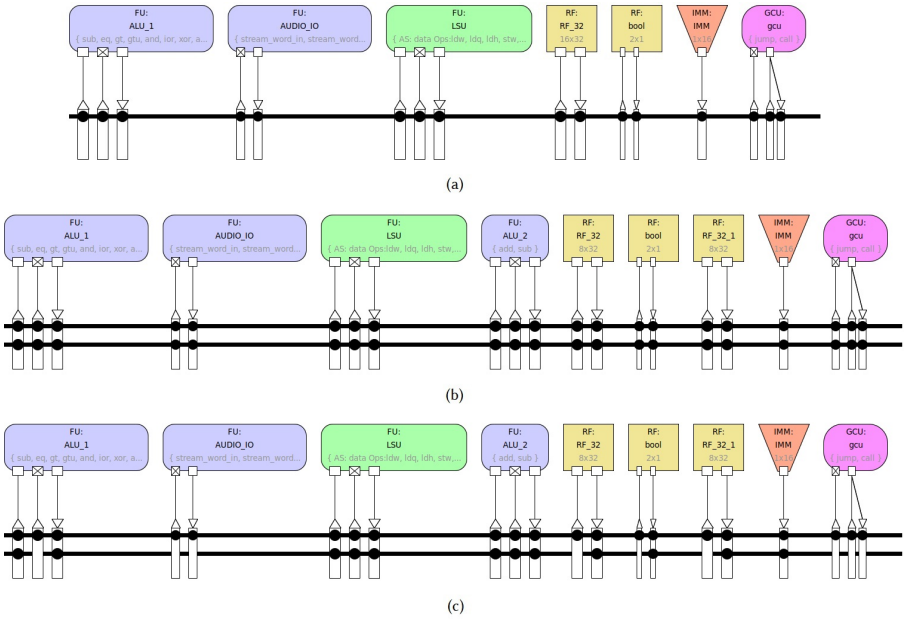

Fig. 6: Internal functional units and interconnection network of the TTA used in (a) *Single-TTA-1*, (b) *Single-TTA-2*, and (c) *Single-TTA-3*.

6 Materials and Methods

The proposed FP-CBEBA algorithm is evaluated in three independent datasets with recordings of dairy cows acquired in different free ranging operating conditions. The first two datasets (DS1 and DS2) are the same ones described and used in the paper that presents the CBEBA [20] in order to evaluate the recognition performance in noiseless and noise conditions. DS1 has more than 4200 JM-events and is composed of 36 segments (short-term record of 75 - 150 s) captured in very low or negligible environmental noise conditions. DS2 is made up of 32 segments (short-term record of 75 - 150 s) with more than 5200 JM-events captured in free ranging conditions typically affected by environmental noises. In this work, the same segments (24 in DS1 and 20 in DS2) used in [20] are considered to conduct a direct comparison between FP-CBEBA and CBEBA. The third dataset (DS3) is the same used and described in [21] with the objective of comparing the JM-events recognition performance between FP-CBEBA and the state-of-art FP-CBRTA. DS3 has more than 3700 JM-events and is formed by 5 segments (long-term record of 10 min).

Two experts in cow foraging behavior with prior experience to identify and classify individual JM-events labeled aurally all segments of the three datasets. The same procedure was previously applied successfully in [19–21, 30, 31,

41]. This supervised labeling was marked as gold-standard for the purpose of evaluating the performance.

A deep description for each dataset about the protocols for animal handling and care, animal information (species and weight), field information (place, date, dimensions, description, and pen size), grass information (species, and height level), data information (observation time per animal, number of observation points per day, total collection days, etc.), and gold-standard JM-events labels is given in [20, 21].

JM-events recognition performance experiment has been performed using a combined scheme of simulation and emulation. The TTA is emulated in a FPGA Artix-7 XC7A35T-L1CSG324I (Xilinx Inc., San José, CA, USA) and is connected to a desktop computer that simulates the works of the Nanocontroller (including the filter co-processor). For this, cycle- and bit-accurate architectural simulation models have been described in SystemC language and functionally verified against the Matlab reference implementation of the FP-CBEBA for the given datasets. For the proposed processor architecture, the Nanocontroller simulation model reads the digital input audio and writes the output results in a binary file. Every time that the Nanocontroller wakes up the TTA, the binary file is transferred via UART through the USB port to the FPGA. Bi-directional USB communication supports the necessary closed loop between Nanocontroller (including the filter co-processor) and TTA.

The FP-CBEBA software implementation for the Nanocontroller and TTA processor cores of the proposed monitoring system has been partitioned manually. All TTA code has been written in C and is compiled and verified using the TCE framework [36]. The Nanocontroller application parts have been manually implemented via hand-written assembly code with a total length of 177 instructions. A custom table assembler tool is applied to translate the assembly code to binary code. For verification and debugging, behavioral simulation using cycle-accurate SystemC models of the Nanocontroller and filter co-processor architectures has been applied. Further details on Nanocontroller assembly language and application development can be found in [35].

After recognition performance and functionality of the proposed monitoring system have been validated with the FPGA-based rapid prototype of the TTA and the simulation models of the Nanocontroller and filter co-processor, an ASIC implementation flow has been set up and completed in order to assess pre-silicon metrics of required silicon area and energy consumption for the selected exemplary target hardware platform, i.e., a semi-custom standard cell ASIC using a 65 nm low-leakage CMOS technology. Hardware descriptions of Nanocontroller and the filter co-processor are provided as hand-written VHDL code at register-transfer level (RTL), VHDL code for the TTA processor configurations is generated from the TCE framework [36]. The RTL description is synthesized to the gate-level netlist representation using the Cadence GENUS 21.1 synthesis solution. Physical implementation, placement and routing are performed with Cadence INNOVUS 21.1 in order to obtain post-place-and-route gate-level netlist, extracted parasitics and timing annotations. In order to

obtain the metrics for energy consumption, the application-specific switching activity of FP-CBEBA for the ASIC implementation is gathered via gate-level simulation with Mentor/Siemens EDA QuestaSim 2021.3. Because the simulation run-time was found to be approximately a factor of 2000 higher than real-time, the simulation of all available recording datasets was not feasible. Instead, a representative portion of an audio segment was selected, so that the simulation workload can be computed within several hours of server time. The representative audio depicts the worst case energy consumption scenario and lasts 10 seconds (24500 audio samples) and contains 14 JM-events, i.e., 1.4 JM-events per second, which is higher than the typical JM frequency range of 0.75 - 1.20 [42]. Continuous operation is assumed for the always-on domain of Nanocontroller and filter co-processor, while the TTA domain is assumed to be powered up only in the 14 classification intervals of correct detected JM-events. The simulated clock of the always-on Nanocontroller domain is set to the minimum required frequency for real-time operation at $f_i = 2450\,\text{Hz}$ and $f_s = 98\,\text{Hz}$, i.e., 500 kHz. The TTA clock is set to the maximum frequency of 66 MHz defined by the common synthesis constraints for all configurations stated in Table 2, in order to minimize the power-up intervals during JM-event classification and, thus, the leakage energy consumption. After simulation, the energy consumption of the implemented chip is extracted from the ASIC library data and switching activity using Synopsys PrimePower S-2021.06-SP1.

For a dense integration, the instruction and data memory of the TTA domain are implemented by SRAM macro block IP for the 65 nm low-leakage technology with sizes of 4352 x 17 bit and 256 x 32 bit, respectively (10272 Bytes in total). In the always-on Nanocontroller and filter co-processor domain, however, all memories are implemented as standard cell memories (SCM, flip-flop arrays) [43]. Particularly, these are 256 Bytes of instruction memory, 48 Bytes of data memory, 576 Bytes of envelope/frame-energy buffers and 74.5 Bytes for storing filter coefficients and filter microcode (954.5 Bytes in total). On the one hand, SCM is selected due to reduced area and power inefficiency of SRAM macro block IP at small memory capacities. On the other hand, SCM is selected because it tolerates a significantly wider range of near-/sub-threshold voltage operation [43], which may be interesting to facilitate in future work in order to further improve the energy efficiency of the always-on system domain.

7 Experimental Results

7.1 Performance Assessment

Proper temporal synchronization between JM-events detected and classified by the algorithms and their corresponding gold-standard reference label is necessary to make valid comparisons. This can be done using the HTK performance analysis tool HResults (HTK 3.4.1, Cambridge University, UK), which is based on a dynamic programming-based string alignment procedure [44]. Outputs

Table 4: Comparative results (mean ± SD; %) of FP-CBEBA and CBEBA.

		Dataset DS1		Dataset DS2	
		Floating point (CBEBA)	Fixed point (FP-CBEBA)	Floating point (CBEBA)	Fixed point (FP-CBEBA)
Detection	*Recall*	**98.4 ± 1.0**	97.4 ± 1.1	**98.0 ± 0.9**	95.0 ± 1.3
	Precision	99.2 ± 0.5	**99.3 ± 0.8**	**95.8 ± 2.1**	92.8 ± 2.4
	$F1-score$	**98.8 ± 0.6**	98.3 ± 0.6	**96.9 ± 1.4**	93.9 ± 1.7
Classification	$Recall_M$	**92.0 ± 2.6**	91.3 ± 3.1	89.8 ± 4.3	**90.0 ± 4.6**
	$Precision_M$	**92.2 ± 2.8**	91.8 ± 3.4	87.0 ± 5.5	**87.9 ± 5.6**
	$F1-score_M$	**92.1 ± 2.4**	91.5 ± 3.1	88.4 ± 4.7	**88.9 ± 4.9**
	$Recall_G$	**88.9 ± 3.7**	88.1 ± 4.5	**87.9 ± 4.8**	87.2 ± 6.3
	$Precision_G$	**89.3 ± 4.3**	89.3 ± 4.6	81.5 ± 8.6	**83.3 ± 8.8**

recognition statistics were used to establish the detection and classification performance metrics.

The JM-events detection supports whether JM-events exist or not, without matter of their proper classes. The performances are reported by *precision*, *recall* and *F1-score* metrics, which depend on the number of correct JM-events detected (true positives), the number of incorrect JM-events detected (false positives) and the number of undetected JM-events (false negatives).

The JM-events classification considers the class of JM-events detected. Classification performances were computed over the correct JM-events detected, averaged by JM-event class (macro-averaging) and reported as F1-score ($F1-score_M$), arithmetic precision ($precision_M$) and arithmetic recall ($recall_M$) [45], and their equivalent geometric precision ($precision_G$) and geometric recall ($recall_G$) [46] according to the number of true positives (tp_i), the number of true negatives (tn_i), the number of false positives (fp_i), and the number of false negatives (fn_i) JM-events for each JM-event class i, respectively.

The comparative performance results of detection and classification of JM-events between FP-CBEBA and the reference CBEBA are shown in Table 4. Performance metrics discrepancies have been evaluated to be statically significant ($p < 0.05$) according to the Wilcoxon signed-rank test [47]. Two adjacent performance values with green background within the same rows and dataset combinations differ significantly, whereas two adjacent performance values with pink background within the same rows and dataset combinations show no significant difference. When comparing FP-CBEBA versus CBEBA, it is observed a slight decline of 0.5% in *F1-score* in the noiseless dataset DS1. However, this decline is greater (3.0%) in the noisy dataset DS2. Furthermore, the *recall* also decreases by 3.0% in DS2, indicating an enhancement of undetected JM-events. On the other hand, it is not possible to state a decline in the JM-events classification rate in DS1 and DS2 datasets, since the results are not statistically significant.

The recognition rate results obtained by each class in the DS1 and DS2 datasets are shown in the confusion matrices of Fig. 7. The confusion matrix of CBEBA in DS1 (Fig. 7 (a)) shows a significant confusion between GC and B events given that 6.9% GC is classified as B and 9.7% B is classified as GC. For

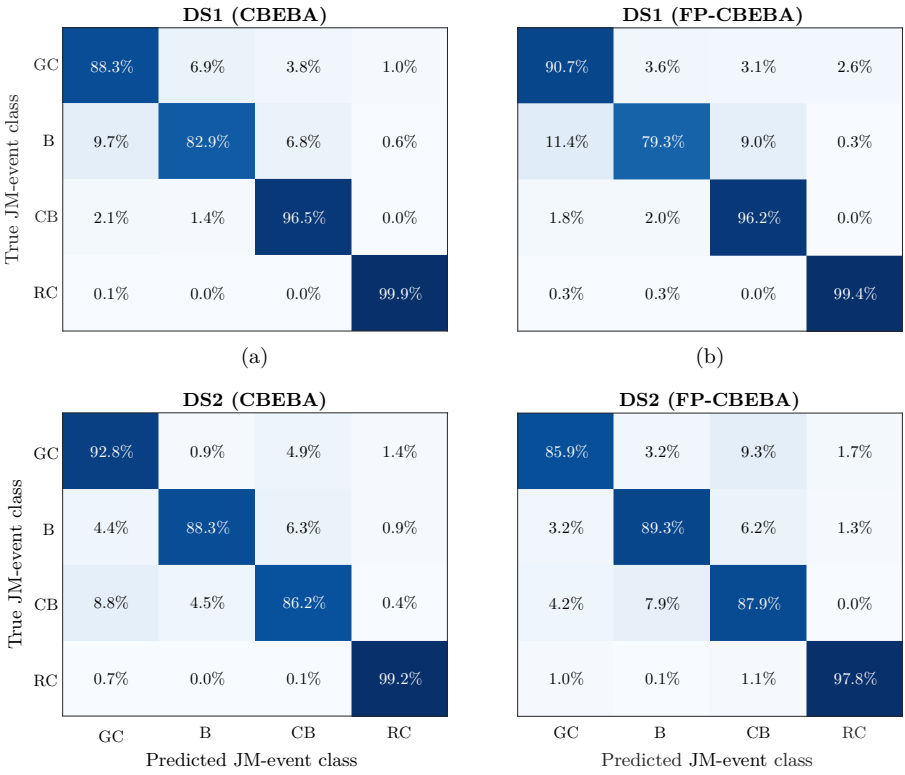

Fig. 7: Confusion matrices of the CBEBA (panels a and c) and FP-CBEBA (panels b and d) algorithms showing recognition rates for chew (C), bite (B), chew-bite (CB) an rumination-chew (RC) events classified respectively in the DS1 and DS2 dataset.

FP-CBEBA in DS1 (Fig. 7 (b)), this misclassification problem is particularly more remarkable for B events, with 11.4% B classified as GC and 9.0% B classified as CB due partly to the linear decision boundary generated by the DT classifier. However, this is coupled with an increment in the recognition rate of GC for FP-CBEBA (90.7% vs. 88.3%) regarding to CBEBA for the same dataset.

The average recognition rates for CBEBA and FP-CBEBA achieve 91.9% and 91.4% in noiseless condition respectively, and 91.6% and 90.2% in noisy condition respectively. The noises affect more negatively the JM-events recognition in FP-CBEBA than in CBEBA due to the different classifier models implemented in each algorithm, among other things. Thereby regarding CBEBA, FP-CBEBA presents a decline of 6.9% and 1.4% for GC and RC respectively, but an increase of 1.0% and 1.7% for B and CB respectively.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (sinc.unl.edu.ar)
L. Rau, M. Weißbrich & G. Payá-Vayá; "A 4W Low-Power Audio Processor System for Real-Time Jaw Movements Recognition in Grazing Cattle"
Journal of Signal Processing Systems, Vol. 95, No. 4, pp. 407-424, 2023.

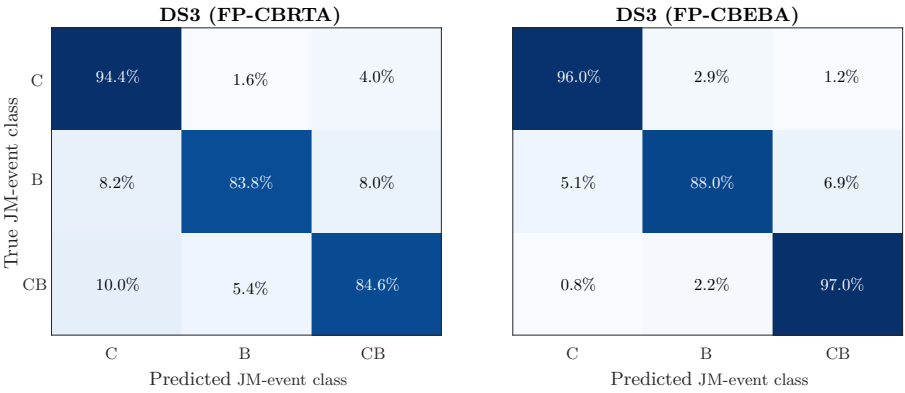

Fig. 8: Recognition rates in terms of chew (C), bite (B) and chew-bite (CB) events using the FP-CBRTA and the FP-CBEBA algorithms in the DS3 dataset.

On the other hand, a direct comparison between FP-CBEBA and FP-CBRTA cannot be established since both algorithms determine different JM-events in terms of classes. One way to compare them is by grouping the rumination-chew (RC) and grazing-chew (GC) classes into a new general chew (C) class, in order to obtain the same JM-events classes recognized by the FP-CBRTA (chew, bite and chew-bite). The confusion matrices in Fig. 8 show the JM-events recognition rates obtained from both algorithms. The results achieve an improvement in favor of FP-CBEBA for all individual classes, with an average recognition rate of 93.7% for FP-CBEBA versus 87.6% for FP-CBRTA. Detection and classification performance metrics have not been assessed, since the 5 audio segments of the DS3 dataset are less than the minimum necessary to perform the Wilcoxon signed-rank test [48].

7.2 Evaluation of the Audio Processor System Chip

The chip implementation of the proposed combined monitoring system (*Nano + Slave-TTA*) is analyzed in terms of required silicon area and compared to the reference single-TTA-based processor systems. From the results presented in Fig. 9, the following observations and conclusions can be made:

- The silicon area of the TTA processor implementations is dominated by the instruction memory (IMEM). Between 63% and 66% are required for the IMEM, while the core area only accounts for 19% to 25% of the TTA domain. Data memory (DMEM) requires between 12% and 14%.
- For the proposed combined *Nano + Slave-TTA* architecture, the TTA part is only used for JM-events classification and can be reduced in area (86% of *Single-TTA-2*). In total, however, larger silicon area is required due to the added Nanocontroller and filter co-processor (170% of *Single-TTA-2*).

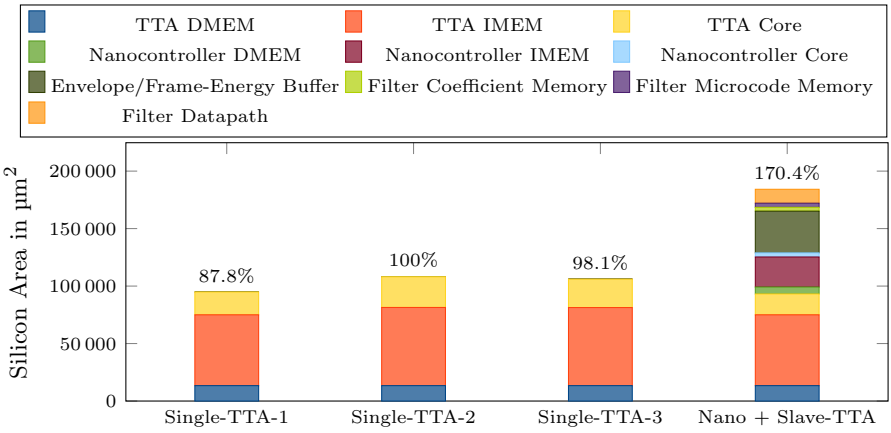

Fig. 9: Comparison of silicon area of the different processor architectures (65 nm low-leakage standard cell ASIC implementation). The relative ratios above the bars are normalized to *Single-TTA-2*, the single-processor configuration with the lowest energy consumption.

- For the *Nano + Slave-TTA* processer system, the Nanocontroller and filter co-processor use 954.5 Bytes of memory, while the TTA domain incorporates 10.8x more (10272 Bytes of IMEM and DMEM). Despite the order of magnitude in capacity, both domains have approximately the same size in terms of area. This is due to less integration density of the standard-cell memory (SCM) in the Nanocontroller domain compared to the SRAM macro IP in the TTA domain. However, as already mentioned above, SCM is expected to have advantages in energy consumption and sub-/near-threshold behavior, which is desired for the always-on Nanocontroller domain.
- Like the TTA domain, the Nanocontroller domain, including the filter co-processor, is dominated by memories with 40% for the envelope/frame-energy buffers and 29% for the Nanocontroller IMEM.
- Due to the memory-dominated domains in terms of silicon area, it is expected that energy consumption will also be dominated by the required memories. Further data path area optimizations, e.g., folding of functional units, will only have limited impact and have therefore not been considered in this work.

Fig. 10 contains the evaluation results for leakage and dynamic energy consumption of the chip while processing the evaluation sequence of 24500 audio samples (10 seconds with 14 correct detected JM-events). Following observations and conclusions can be made:

- As expected, the energy consumption of the 3 Single-TTA configurations is dominated by the dynamic energy consumption of the memories (52% to 54% of the total energy consumption).

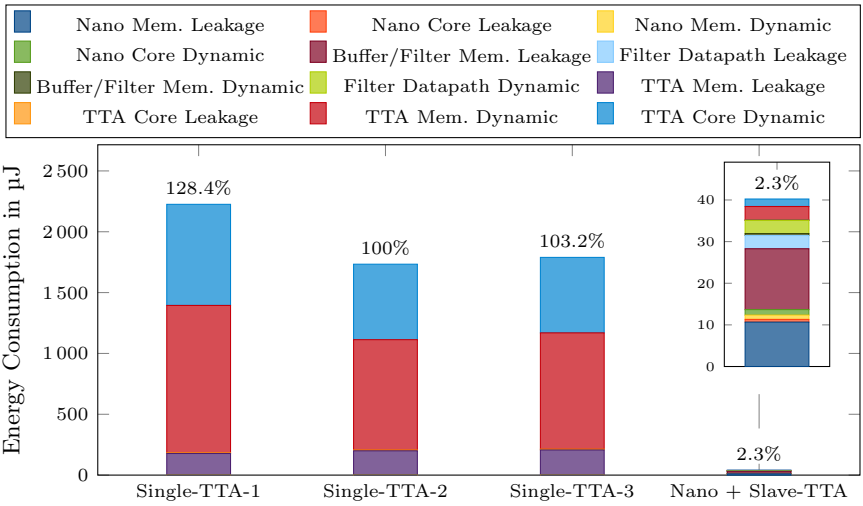

Fig. 10: Comparison of leakage and dynamic energy consumption of the different processor architectures for processing the evaluation data set (65 nm low-leakage standard cell ASIC implementation). The relative ratios above the bars are normalized to *Single-TTA-2*, the single-processor configuration with the lowest energy consumption.

- Although the Single-TTA configurations cannot be powered-down and must be always active for audio sample pre-processing, filtering and JM-events detection stages, leakage energy is not the major issue and accounts only for 8% to 12% of the total energy consumption. Also this part is fully dominated by the memories, the TTA core leakage component is negligible and disappears in the plot due to its tiny quantity.
- All architectures apply clock gating to reduce dynamic energy consumption in idle phases. Because the energy consumption of the Single-TTA configurations is defined by dynamic energy consumption, more performant architectures, which require less clock cycles, consume less energy despite their increased processing resources and silicon area. Together with the information from Table 2 and Fig. 9, it can be seen that *Single-TTA-2* requires 57% of the clock cycles and clock frequency of *Single-TTA-1*, but has only 14% increased area. Therefore, clock cycle reduction overcompensates the area increase and *Single-TTA-2* has the lowest energy consumption of the 3 single TTA-based processors, i.e., 1.73 mJ are required for the 10 seconds sequence (173 µW on average).
- In the proposed combined *Nano + Slave-TTA* architecture, the TTA domain is powered down by default and only powered up for JM-events classification. For the evaluation sequence, this reduces the energy consumption of the TTA to only 0.3% of *Single-TTA-2*. Together with the specialized Nanocontroller and filter co-processor, the *Nano + Slave-TTA* chip implementation requires

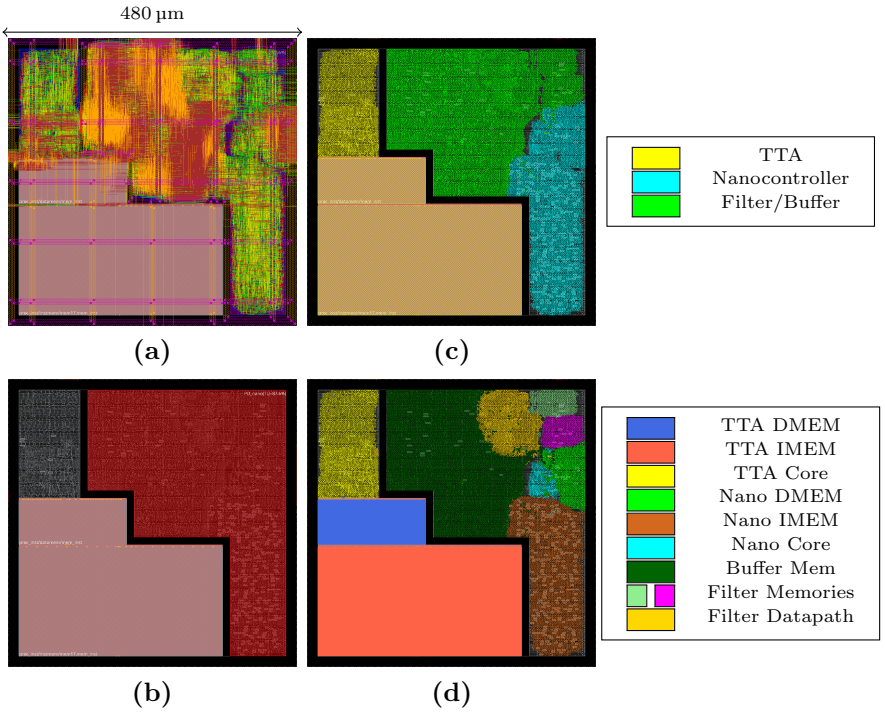

Fig. 11: Layout of the ASIC implementation of the proposed monitoring system: **(a)** Physical view with signal and power routing, **(b)** power domain view with power-gated TTA domain on the left and always-on Nanocontroller domain on the right, **(c)** highlighted core placement, **(d)** highlighted detailed module placement.

2.3% of the energy consumption of *Single-TTA-2*. In absolute values, 40 µJ are required for the 10 seconds sequence, i.e., very low 4 µW on average.

- The energy consumption of the combined *Nano + Slave-TTA* chip implementation is dominated by the always-on domain, consisting of Nanocontroller and filter co-processor (87% of the total energy consumption). As expected, power consumption of memories has the largest fractions, i.e., 36% account for buffer and filter memory leakage energy, and 26% account for the Nanocontroller IMEM and DMEM leakage energy. The strong occurrence of leakage compared to dynamic energy can be explained with the low required clock frequency of 500 kHz, which is sufficient for audio sample pre-processing, filtering and JM-events detection in the always-on domain.

The considered physical chip implementation of the proposed monitoring system (*Nano + Slave-TTA*) using 4 views from the Cadence INNOVUS 21.1 implementation solution is depicted in Fig. 11. Due to the consistent color

highlighting in Fig. 11 (d), the placement and dimensions of the system modules can be directly compared to the area results in Fig. 9. Particularly, the approximate silicon area equality between the TTA domain (on the left side, including the 2 memory macro blocks for IMEM and DMEM) and the always-on Nanocontroller and filter co-processor domain (on the right side) can be observed.

8 Conclusion

In this work, a real-time JM-events recognizer implemented on a novel low-power audio processor system has been proposed. The JM-events recognizer is a pattern recognition algorithm that uses a combination of classical signal processing and machine learning techniques. The proposed JM-events recognizer works on fixed-point arithmetic format, thus reducing the power consumption. Experimental performances using noiseless and noisy audio records on free ranging conditions is compared against the floating-point reference implemented on a computer. No evidence of decrease in the classification of JM-events is found, and only a small decrease of 3.0% for the detection of JM-events. The recognition in real-time is desired to ensure the animal wellness and to improve the livestock management systems, among others. However, there is a lack of previous solutions since it is a difficult task. The proposed FP-CBEBA increases by 6.1% the recognition rate regarding a previous reference on-line embedded device, and is able to be used in future embedded fixed-point processor systems. Moreover, a second significant property of event-based real-time processing algorithms like the FP-CBEBA is that these can be partitioned into *always-on processing* and *event-triggered processing*. Always-on processing consists of frequent, continuously repeating tasks of limited computational complexity, i.e., for FP-CBEBA, audio sample pre-processing and threshold-based JM-events detection. In case of detected JM-events, which appear less frequently and intermittently, algorithmic parts of higher computational complexity are triggered, i.e., feature-based JM-events classification.

As a further contribution of this work, an audio processor system has been proposed, which exploits the event-based property in order to significantly increase the energy efficiency of processing FP-CBEBA. The system consists of two specialized processor cores in separate power domains. The always-on domain contains a very small Nanocontroller core with a minimal 4-bit ISA, as well as a programmable filter co-processor, which are used to perform energy-efficient, continuous audio sample pre-processing and JM-event detection at a low clock frequency of 500 kHz. A VLIW-style processor core, in this case a transport-triggered architecture (TTA), provides higher performance in order to process the feature-based classification of JM-events in real-time. The latter core resides in a second power domain, which is controlled by the Nanocontroller and can be shut down completely in order to eliminate any energy consumption when no event has been triggered. The proposed

system has been implemented and evaluated as a chip design, using a low-leakage 65 nm standard cell ASIC technology. In comparison to a conventional always-on single-processor reference implementation using the same technology, the average power consumption of the chip is significantly reduced from 173 µW to 4 µW, i.e., to 2.3 % of the reference value for processing the same audio sequence, with 70 % higher silicon area requirements due to the added Nanocontroller and programmable filter co-processor.

Acknowledgments. The authors would like to thank KBS Robotic Dairy Farm staff and Campo Experimental J. Villarino Dairy Farm staff involved in the data gather, curation and labelling. A special thanks to Julio Galli from Facultad de Ciencias Agrarias, Universidad Nacional de Rosario and Santiago Utsumi from the Department of Animal and Range Science, New Mexico State University, United States for their constant support, help and assistance during the completation of this work. This work has been granted by the Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET) and the Deutscher Akademischer Austauschdienst (DAAD) (ID: 57507441).

Declarations

Availability of data and materials

The authors declare that all relevant evaluation data are available in the figures and tables within the article. The generated and analyzed datasets are available from the corresponding author on reasonable request.

References

[1] Berckmans, D. General introduction to precision livestock farming. *Animal Frontiers* **7** (1), 6–11 (2017) .

[2] Michie, C., Andonovic, I., Davison, C., Hamilton, A., Tachtatzis, C., Jonsson, N., Duthie, C.-A., Bowen, J. & Gilroy, M. The internet of things enhancing animal welfare and farm operational efficiency. *Journal of Dairy Research* **87** (S1), 20–27 (2020) .

[3] González, L., Tolkamp, B., Coffey, M., Ferret, A. & Kyriazakis, I. Changes in feeding behavior as possible indicators for the automatic monitoring of health disorders in dairy cows. *Journal of Dairy Science* **91** (3), 1017–1028 (2008) .

[4] Mahmud, M. S., Zahid, A., Das, A. K., Muzammil, M. & Khan, M. U. A systematic literature review on deep learning applications for precision cattle farming. *Computers and Electronics in Agriculture* **187**, 106313 (2021). https://doi.org/10.1016/j.compag.2021.106313 .

[5] Chen, C., Zhu, W. & Norton, T. Behaviour recognition of pigs and cattle: Journey from computer vision to deep learning. *Computers and Electronics in Agriculture* **187**, 106255 (2021) .

[6] Becker, T., Kluge, M., Schalk, J., Tiplady, K., Paget, C., Hilleringmann, U. & Otterpohl, T. Autonomous sensor nodes for aircraft structural health monitoring. *IEEE Sensors Journal* **9** (11), 1589–1595 (2009) .

[7] Aquilani, C., Confessore, A., Bozzi, R., Sirtori, F. & Pugliese, C. Review: Precision livestock farming technologies in pasture-based livestock systems. *Animal* **16** (1), 100429 (2022). https://doi.org/10.1016/j.animal.2021.100429 .

[8] Kilgour, R. J. In pursuit of "normal": A review of the behaviour of cattle at pasture. *Applied Animal Behaviour Science* **138** (1-2), 1–11 (2012) .

[9] Augustine, D. J. & Derner, J. D. Assessing herbivore foraging behavior with GPS collars in a semiarid grassland. *Sensors* **13** (3), 3711–3723 (2013) .

[10] Pavlovic, D., Czerkawski, M., Davison, C., Marko, O., Michie, C., Atkinson, R., Crnojevic, V., Andonovic, I., Rajovic, V., Kvascev, G. & Tachtatzis, C. Behavioural classification of cattle using neck-mounted accelerometer-equipped collars. *Sensors* **22** (6), 2323 (2022) .

[11] Rodrigues, J. P. P., Pereira, L. G. R., do Carmo Diniz Neto, H., Lombardi, M. C., de Assis Lage, C. F., Coelho, S. G., Sacramento, J. P., Machado, F. S., Tomich, T. R., Maurício, R. M. & Campos, M. M. Technical note: Evaluation of an automatic system for monitoring rumination time in weaning calves. *Livestock Science* **219**, 86–90 (2019). https://doi.org/10.1016/j.livsci.2018.11.017 .

[12] Sakai, K., Oishi, K., Miwa, M., Kumagai, H. & Hirooka, H. Behavior classification of goats using 9-axis multi sensors: The effect of imbalanced datasets on classification performance. *Computers and Electronics in Agriculture* **166**, 105027 (2019) .

[13] Chen, G., Li, C., Guo, Y., Shu, H., Cao, Z. & Xu, B. Recognition of cattle's feeding behaviors using noseband pressure sensor with machine learning. *Frontiers in Veterinary Science* **9** (2022) .

[14] Cabezas, J., Yubero, R., Visitación, B., Navarro-García, J., Algar, M. J., Cano, E. L. & Ortega, F. Analysis of accelerometer and GPS data for cattle behaviour identification and anomalous events detection. *Entropy* **24** (3), 336 (2022) .

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (sinc.unl.edu.ar)
L. Rau, M. Weißbrich & G. Payá-Vayá; "A 4W Low-Power Audio Processor System for Real-Time Jaw Movements Recognition in Grazing Cattle"
Journal of Signal Processing Systems, Vol. 95, No. 4, pp. 407-424, 2023.

[15] Galli, J., Cangiano, C., Pece, M., Larripa, M., Milone, D., Utsumi, S. & Laca, E. Monitoring and assessment of ingestive chewing sounds for prediction of herbage intake rate in grazing cattle. *Animal* **12** (5), 973–982 (2018). https://doi.org/10.1017/S1751731117002415 .

[16] Galli, J. R., Cangiano, C. A., Milone, D. H. & Laca, E. A. Acoustic monitoring of short-term ingestive behavior and intake in grazing sheep. *Livestock Science* **140** (1), 32–41 (2011). https://doi.org/10.1016/j.livsci.2011.02.007 .

[17] Ungar, E. D., Ravid, N., Zada, T., Ben-Moshe, E., Yonatan, R., Baram, H. & Genizi, A. The implications of compound chew–bite jaw movements for bite rate in grazing cattle. *Applied Animal Behaviour Science* **98** (3-4), 183–195 (2006) .

[18] Beauchemin, K. A. Ingestion and mastication of feed by dairy cattle. *Veterinary Clinics of North America: Food Animal Practice* **7** (2), 439–463 (1991). https://doi.org/10.1016/S0749-0720(15)30794-5 .

[19] Chelotti, J. O., Vanrell, S. R., Rau, L. S. M., Galli, J. R., Planisich, A. M., Utsumi, S. A., Milone, D. H., Giovanini, L. L. & Rufiner, H. L. An online method for estimating grazing and rumination bouts using acoustic signals in grazing cattle. *Computers and Electronics in Agriculture* **173**, 105443 (2020) .

[20] Martinez-Rau, L. S., Chelotti, J. O., Vanrell, S. R., Galli, J. R., Utsumi, S. A., Planisich, A. M., Rufiner, H. L. & Giovanini, L. L. A robust computational approach for jaw movement detection and classification in grazing cattle using acoustic signals. *Computers and Electronics in Agriculture* **192**, 106569 (2022). https://doi.org/10.1016/j.compag.2021.106569 .

[21] Deniz, N. N., Chelotti, J. O., Galli, J. R., Planisich, A. M., Larripa, M. J., Rufiner, H. L. & Giovanini, L. L. Embedded system for real-time monitoring of foraging behavior of grazing cattle using acoustic signals. *Computers and Electronics in Agriculture* **138**, 167–174 (2017) .

[22] Martinez Rau, L. S., Deniz, N. N., Chelotti, J. O., Giovanini, L. L. & Kler, P. A. Acoustic real-time sensor for ingestive behaviour of grazing cattle. *IX Congreso de Microelectrónica Aplicada (UEA2018)* (Universidad Nacional de Catamarca, 2018).

[23] Riaboff, L., Shalloo, L., Smeaton, A. F., Couvreur, S., Madouasse, A. & Keane, M. T. Predicting livestock behaviour using accelerometers: A systematic review of processing techniques for ruminant behaviour prediction from raw accelerometer data. *Computers and Electronics in Agriculture* **192**, 106610 (2022) .

[24] Giovanetti, V., Cossu, R., Molle, G., Acciaro, M., Mameli, M., Cabiddu, A., Serra, M., Manca, C., Rassu, S., Decandia, M. & Dimauro, C. Prediction of bite number and herbage intake by an accelerometer-based system in dairy sheep exposed to different forages during short-term grazing tests. *Computers and Electronics in Agriculture* **175**, 105582 (2020). https://doi.org/10.1016/j.compag.2020.105582 .

[25] Ding, L., Lv, Y., Jiang, R., Zhao, W., Li, Q., Yang, B., Yu, L., Ma, W., Gao, R. & Yu, Q. Predicting the feed intake of cattle based on jaw movement using a triaxial accelerometer. *Agriculture* **12** (7), 899 (2022) .

[26] Li, C., Tokgoz, K. K., Fukawa, M., Bartels, J., Ohashi, T., Takeda, K. & Ito, H. Data augmentation for inertial sensor data in cnns for cattle behavior classification. *IEEE Sensors Letters* **5** (11), 1–4 (2021). https://doi.org/10.1109/LSENS.2021.3119056 .

[27] Werner, J., Leso, L., Umstatter, C., Niederhauser, J., Kennedy, E., Geoghegan, A., Shalloo, L., Schick, M. & O'Brien, B. Evaluation of the RumiWatchSystem for measuring grazing behaviour of cows. *Journal of Neuroscience Methods* **300**, 138–146 (2018). https://doi.org/10.1016/j.jneumeth.2017.08.022 .

[28] Rombach, M., Südekum, K.-H., Münger, A. & Schori, F. Herbage dry matter intake estimation of grazing dairy cows based on animal, behavioral, environmental, and feed variables. *Journal of Dairy Science* **102** (4), 2985–2999 (2019). https://doi.org/10.3168/jds.2018-14834 .

[29] Gregorini, P., Dela Rue, B., Pourau, M., Glassey, C. & Jago, J. A note on rumination behavior of dairy cows under intensive grazing systems. *Livestock Science* **158** (1), 151–156 (2013). https://doi.org/10.1016/j.livsci.2013.10.012 .

[30] Chelotti, J. O., Vanrell, S. R., Milone, D. H., Utsumi, S. A., Galli, J. R., Rufiner, H. L. & Giovanini, L. L. A real-time algorithm for acoustic monitoring of ingestive behavior of grazing cattle. *Computers and Electronics in Agriculture* **127**, 64–75 (2016) .

[31] Chelotti, J. O., Vanrell, S. R., Galli, J. R., Giovanini, L. L. & Rufiner, H. L. A pattern recognition approach for detecting and classifying jaw movements in grazing cattle. *Computers and Electronics in Agriculture* **145**, 83–91 (2018) .

[32] Shresthamali, S., Kondo, M. & Nakamura, H. Adaptive power management in solar energy harvesting sensor node using reinforcement learning. *ACM Transactions on Embedded Computing Systems (TECS)* **16** (5s), 1–21 (2017) .

[33] Bishop, C. M. *Pattern Recognition and Machine Learning* (Springer Verlag, 2006).

[34] Weißbrich, M., Blume, H. & Payá-Vayá, G. A silicon-proof controller system for flexible ultra-low-power energy harvesting platforms. *2022 11th International Conference on Modern Circuits and Systems Technologies (MOCAST)*, 1–6 (2022).

[35] Weißbrich, M. & Payá-Vayá, G. Nanocontroller: A minimal and flexible processor architecture for ultra-low-power always-on system state controllers. *Embedded Computer Systems: Architectures, Modeling, and Simulation*, 103–119 (Springer International Publishing, Cham, 2022).

[36] Jääskeläinen, P., Viitanen, T., Takala, J. & Berg, H. *HW/SW Co-design Toolset for Customization of Exposed Datapath Processors*, 147–164 (Springer International Publishing, 2017). URL https://doi.org/10.1007/978-3-319-49679-5_8.

[37] Corporaal, H. *Microprocessor architectures - from VLIW to TTA* (John Wiley & Sons, 1997).

[38] Jääskeläinen, P., Tervo, A., Payá-Vayá, G., Viitanen, T., Behmann, N., Takala, J. & Blume, H. Transport-triggered soft cores. *2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, 83–90 (IEEE, 2018).

[39] Jääskeläinen, P., Kultala, H., Viitanen, T. & Takala, J. Code density and energy efficiency of exposed datapath architectures. *Journal of Signal Processing Systems* **80** (1), 49–64 (2015) .

[40] Panda, P. R., Silpa, B., Shrivastava, A. & Gummidipudi, K. *Power-efficient system design* (Springer Science & Business Media, New York London, 2010).

[41] Vanrell, S. R., Chelotti, J. O., Galli, J. R., Utsumi, S. A., Giovanini, L. L., Rufiner, H. L. & Milone, D. H. A regularity-based algorithm for identifying grazing and rumination bouts from acoustic signals in grazing cattle. *Computers and Electronics in Agriculture* **151**, 392–402 (2018) .

[42] Andriamandroso, A., Bindelle, J., Mercatoris, B. & Lebeau, F. A review on the use of sensors to monitor cattle jaw movements and behavior when grazing. *Biotechnologie, Agronomie, Société et Environnement* **20** (2016) .

[43] Teman, A., Rossi, D., Meinerzhagen, P., Benini, L. & Burg, A. Power, area, and performance optimization of standard cell memory arrays through controlled placement. *ACM Trans. Des. Autom. Electron. Syst.*

21 (4) (2016). https://doi.org/10.1145/2890498 .

[44] Young, S., Evermann, G., Gales, M., Kershaw, D., Moore, G., Odell, J., Ollason, D., Povey, D., Valtchev, V. & Woodland, P. The HTK book version 3.4 manual. *Cambridge University Engineering Department* (2006) .

[45] Sokolova, M. & Lapalme, G. A systematic analysis of performance measures for classification tasks. *Inf. Process. Manag.* **45** (4), 427–437 (2009) .

[46] Ballabio, D., Grisoni, F. & Todeschini, R. Multivariate comparison of classification performance measures. *Chemometrics and Intelligent Laboratory Systems* **174**, 33–44 (2018) .

[47] Wilcoxon, F. Individual comparisons by ranking methods. *Biometrics Bulletin* **1** (6), 80 (1945) .

[48] Mundry, R. & Fischer, J. Use of statistical programs for nonparametric tests of small samples often leads to incorrect p values: examples from animal behaviour. *Animal Behaviour* **56** (1), 256–259 (1998). https://doi.org/10.1006/anbe.1998.0756 .