TamaRISC-CS: An Ultra-Low-Power Application-Specific Processor for Compressed Sensing

Jeremy Constantin*, Ahmed Dogan*, Oskar Andersson†, Pascal Meinerzhagen*,
Joachim Neves Rodrigues†, David Atienza*, and Andreas Burg*
* Institute of Electrical Engineering, EPFL, Lausanne, VD, 1015 Switzerland
Email: {jeremy.constantin,ahmed.dogan,pascal.meinerzhagen,david.atienza,andreas.burg}@epfl.ch
†Department of Electrical and Information Technology, Lund University, Lund, 22100 Sweden
Email: {oskar.andersson,joachim.rodrigues}@eit.lth.se

***Abstract*—Compressed sensing (CS) is a universal technique for the compression of sparse signals. CS has been widely used in sensing platforms where portable, autonomous devices have to operate for long periods of time with limited energy resources. Therefore, an ultra-low-power (ULP) CS implementation is vital for these kind of energy-limited systems. Sub-threshold (sub-V_T) operation is commonly used for ULP computing, and can also be combined with CS. However, most established CS implementations can achieve either no or very limited benefit from sub-V_T operation. Therefore, we propose a sub-V_T application-specific instruction-set processor (ASIP), exploiting the specific operations of CS. Our results show that the proposed ASIP accomplishes 62x speed-up and 11.6x power savings with respect to an established CS implementation running on the baseline low-power processor.**

I. Introduction

Digital signal processing traditionally relies on the Nyquist sampling theorem, stating that a faithful reconstruction of a signal, limited to a bandwidth B in the frequency spectrum, can be ensured with a sampling rate of $f_s \geq 2 * B$. However, this bandwidth requirement can limit operational time of a portable device due to high transmission costs. In fact, sparse signals include a considerable amount of redundant samples when sampled at the Nyquist sampling rate. Compressed sensing (CS) [1] is a universal data compression technique to compress sparse signals. CS has been widely applied to sensing environment systems and wireless body sensor networks (WBSNs) [2], where portable and autonomous devices are expected to operate for long periods of time with limited energy resources. Hence, an ultra-low-power (ULP) CS implementation is crucial for these energy-limited autonomous systems.

Supply voltage scaling, potentially all the way to the sub-threshold (sub-V_T) regime, can reduce both dynamic and leakage power consumption. To this end, many sensing platforms exploit sub-V_T computing. The state-of-the-art processors for sensing platforms have been reported to consume as little as a few pJ/cycle while operating in the sub-V_T regime [3–5]. Sub-V_T computing can also be applied to the CS. However, most established CS implementations either require a large memory footprint or high computational effort, which can limit the benefit of sub-V_T computing. Leakage power consumption becomes a very important challenge in the sub-V_T regime with reduced active power. A considerable amount of leakage is due to memories in sensing platforms [6]. Moreover, many sensing platforms cannot be power gated completely, to retain their memory content [4], and hence leakage power is always dissipated. Therefore, implementations with large memory requirements are not desirable in the sub-V_T regime. On the other hand, high computational effort requirements can limit the degree of voltage scaling because of performance degradation issues in the sub-V_T regime [7–9].

Application-specific instruction-set processors (ASIPs) can compensate for this performance degradation issue, since they are optimized for a specific application domain, providing increased efficiency and performance for the core algorithms of the domain's target applications. For instance, an ASIP optimized for stereo image processing can achieve up to 130x speed-up compared to a conventional processor [10]. These performance optimizations also lead to energy saving as in [11], where a processing core with few accelerators dedicated to biomedical applications, can achieve up to 11.5x energy saving compared the processing core-only implementation. Despite of their efficiency in some specialized application domains, to the authors knowledge no ASIP core has been reported for low-power CS.

Contributions: In this paper, we propose to synergistically exploit sub-V_T computing in conjunction with an ASIP core for CS to provide an ultra-low-power (ULP) processor for sensing environment applications. To this end, we extend the instruction set of a low-power processor to exploit the specific operations of the CS compression algorithm. Our ASIP core, neither requires high clock frequencies, therefore enabling sub-V_T computing, nor excessive memory requirement, hence major leakage power cut. For a typical case study of electrocardiogram (ECG) signal compression in WBSNs, it consumes only 30.6 nW for an ECG sampling rate of 125 Hz. Moreover, we have shown that the proposed processing platform achieves 62x speed-up and 11.6x power saving with respect to the established computation-based CS implementation running on the baseline low-power processor.

II. Compressed Sensing

Signal compression based on compressed sensing (CS) [1] is performed by computing:

$$y = \mathbf{\Phi} x$$

where the random sensing matrix $\mathbf{\Phi} \in \mathbb{R}^{k \times n}$ with $k < n$ maps an input data vector $x \in \mathbb{R}^n$ holding n samples to a compressed data vector $y \in \mathbb{R}^k$ with k entries, at a compression ratio of $\frac{k}{n}$.

There are multiple approaches of how to choose a random sensing matrix $\mathbf{\Phi}$ with k rows and n columns. Sensing matrices with near optimal properties can for example be constructed by random sampling from a uniform distribution on the columns of $\mathbf{\Phi}$ [1].

A. Reduced Complexity Algorithm

The computational complexity of the matrix-vector multiplication is determined by the structure and values of $\mathbf{\Phi}$. Mamaghanian et al. [2] show (for WBSNs) that in fact choosing $\mathbf{\Phi}$ as a sparse matrix containing only a few non-zero entries per column at random positions is a valid low-complexity approach, that still provides good integrity of the compressed sparse signal. The non-zero elements can furthermore be chosen as 1, and the number of ones per column can be fixed. These constraints on $\mathbf{\Phi}$ lead to a very efficient algorithm (Algorithm 1) for performing CS. Note that the sensing matrix can be represented in a compact form by a sequence of random indices $\in \{1, 2, ..., k\}$.

Algorithm 1 Pseudocode of Compressed Sensing Algorithm

```
1: for i := 1 → n do
2:     sample := getSample()
3:     for j := 1 → #ones do
4:         index := getRandomIndex(1..k)
5:         buffer[index] := buffer[index] + sample
6:     end for
7: end for
```

The key challenge of Algorithm 1 on a resource constrained system is the generation of the random indices. The optimized reference implementation [2] uses a sensing matrix realized as a fixed sequence of buffer indices stored in memory (for a specific value of k). Since large memory footprints are undesirable, especially in the context of ULP sensor nodes, we discuss the generation of the required random indices at runtime.

B. Pseudo Random Number Generation

The generation of random indices can be realized through the use of a pseudo random number generator (RNG). A common implementation of such an RNG is a linear feedback shift register (LFSR). The random sequence generated by an LSFR is defined by the sequence of its internal states. The initial state of an LFSR is defined as the seed. For each state transition (LFSR step) the current internal state bits are combined with the binary coefficients of a polynomial, which defines the pseudo random sequence of the LFSR. The bits selected by the polynomial are summed (parity bit) to produce one new random bit. The next state of an LFSR is calculated by shifting out the least significant bit of the state, and shifting the generated bit in, as the new most significant bit.

For maximum-length LFSRs the cycle length of the generated random number sequence is equal to the number of maximum possible states (excluding zero). Note that although maximum-length LFSRs can provide good sequences of random numbers, the correlation between two subsequent LFSR states, i.e. indices, i_1 and i_2 is high, since $i_2 = i_1/2$ or $i_2 = i_1/2 + k/2$. When the state is used directly, this correlation of the generated indices has a negative effect on the reconstruction quality of the compressed samples (i.e. producing PRD values significantly higher than 9 for mid-low compression ratios, cf. Section IV-C). Hence, we propose to use an LFSR that advances multiple steps per generated index. The number of steps is equal to the number of used index bits. For example, for $k = 256$ the LFSR has to advance 8 states to generate the next index, which yields a small correlation to its predecessor. The quality of our generated random indices for CS is assessed in the case study presented in Section IV-C. The drawback of this approach is the increased computational effort for the RNG, which can be compensated for by custom hardware support.

The proposed generation of the sensing matrix $\mathbf{\Phi}$ can be characterized with four main parameters: the LFSR polynomial, the LFSR seed, the number of index bits (depending on k), and the number of non-zero elements per column (*#ones*). These four configuration parameters enable the generation of a large set of different sensing matrices. Choosing from a pre-constructed pool of feasible values for the RNG configuration, it is hence possible to achieve a good sensing performance for a variety of different signal conditions, potentially even by dynamically changing the RNG configuration at runtime.

III. Sub-V_T CS Processor

The challenge for implementing the data compression algorithm (Algorithm 1) in a resource constrained environment lies in the realization of the random number sequences needed to address the elements in the sensing buffer. Hence, the goal of our custom designed ASIP architecture is to provide support for an efficient random number sequence generation, enabling energy efficient operation in the sub-V_T regime.

A. Processor Baseline Architecture

The baseline microprocessor used in this study is a custom 16-bit reduced instruction set computing (RISC) architecture (TamaRISC [12]), as shown in Figure 1. TamaRISC provides a complete RISC instruction set, a C-Compiler, as well as interrupt capability for basic embedded real-time operating system support.

1) Core Architecture: The architecture focuses on minimizing the instruction set complexity in a true RISC fashion, while still providing enough hardware support, especially regarding addressing modes, for efficient execution of signal processing applications. The processor has a 3-stage pipeline, comprised of a fetch, decode and execute stage. The core operates on a data word width of 16-bit, comprises 16 general

purpose working registers and 3 external memory ports, one for instruction fetch, one for data read and one for data write. The register file has 3 read ports and 4 write ports, and provides 32-bit double word writeback support. Instruction words are 24 bit wide, with every instruction using only a single word. All instructions generally execute in one cycle, which is guaranteed by the use of complete data bypassing inside the core for register as well as memory writeback data.

2) Instruction Set: The instruction set architecture (ISA) comprises a total of 14 unique instructions, with 8 arithmetic logic unit (ALU) instructions, 2 general data move instructions, 2 program flow instructions, a sleep mode instruction, and an instruction to provide basic hardware loops. The ALU supports addition, subtraction (each with optional carry/borrow), logical AND, OR and XOR, right (arithmetic or not) and left shift, as well as full 16-bit by 16-bit multiplication (32 bit-result) on unsigned and signed data.

All ALU instructions work on 3 operands, using the exact same addressing mode options for each instruction, which helps to reduce complexity of the architecture, since the operand fetch logic and the arithmetic operation are completely decoupled. The supported addressing modes are register direct, register indirect (with pre- or post-increment and decrement) as well as register indirect with offset. The second operand also supports the use of 4-bit literals. Regarding program flow instructions, branching is possible in direct and register indirect mode, as well as by offset with 15 different condition modes. The ISA also includes instructions for interrupt and sleep mode support of the core. The sleep mode allows external clock-gating of the entire core, until a wakeup event occurs (e.g. an IRQ for a new ADC sample).

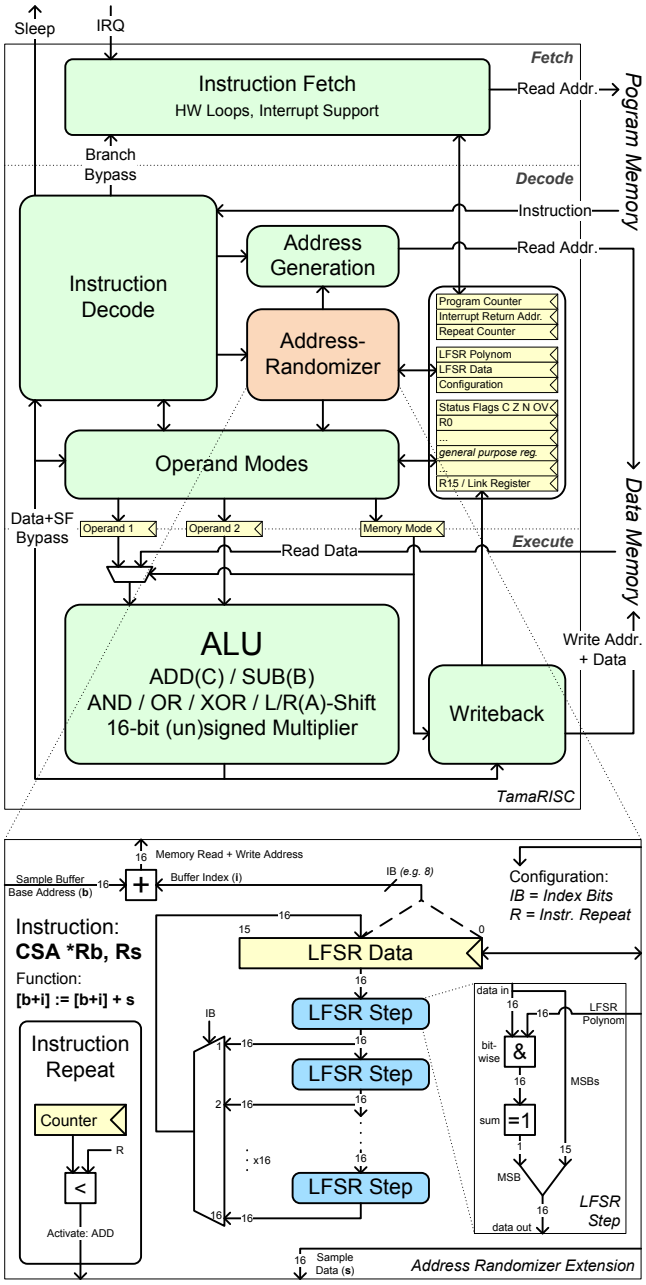

Fig. 1. TamaRISC Sub-V_T Microprocessor Architecture including Address-Randomizer Extension for CS

B. Sub-V_T Memories

In this work, we employ a sub-V_T latch-based memory design [13] as alternative to 8T/10T SRAM hardmacros, which simplifies the design flow, ensures reliable data storage at no extra design effort, and allows to perform supply voltage scaling for the complete system. Even though these latch-based memories are optimized for low-voltage/low-power operation, they still consume a considerable leakage power. In our system example, memories account for 70–95% of the system's total power consumption, depending on the mode of operation. Furthermore, the sub-V_T memories consume a lot of area: our implementation with moderate memory sizes of 256 instructions and 512 data words results in the processing core only consuming 16% of the total area.

C. Index Sequence Implementations

There are two main approaches to generate the random numbers, used as buffer indices in Algorithm 1: precomputation and storage of all required indices in form of a large array in data memory, or computation of the index sequence at runtime based on a pseudo RNG.

1) Precomputation: The storage of a preconstructed sequence effectively trades computational effort for memory consumption. For example, the requirement for a single sensing matrix (with 12 non-zero entries per column), used for the compression of a set of 512 samples by 50%, is 6 Kbyte of memory. However, a relatively large memory footprint is especially undesirable in an ULP embedded system, for reasons of die area and power consumption. Since sub-V_T memories are large and consume most of the total power through leakage for low voltages, the storage of tens of Kbyte of data for sensing matrices is not a feasible option, especially when different matrices are to be supported.

2) Computation at Runtime: The second approach for the generation of random sequences of indices applies computation based on pseudo RNGs, such as the algorithm proposed

in Section II-B. The only required data memory is hence the sensing buffer, which for compression of a set of 512 samples by 50% requires 256 data words (e.g. 512 byte with a sample precision of 12 bit). As shown in Section II-B, for each generated index, the RNG has to perform the same number of LFSR steps as the number of bits per index. A typical implementation (on a RISC ISA) in software can perform one 16-bit LFSR step in about 10 operations. For the example of a sensing buffer size of 256 and 12 ones per column in the sensing matrix, this results in a computational requirement of about 1k operations per sample, dedicated to the task of random number generation alone. Downscaling of the supply voltage also considerably limits the maximum core clock frequency (cf. Figure 2). Due to the relatively large computational overhead, achievable sampling rates for sub-V_T operation are therefore reduced to the range of tens of Hz, which is undesirable.

To combine the benefits of instant random number access of the storage approach, with the memory savings of the computational approach, we propose an instruction set extension for TamaRISC, which performs the computational task of index generation efficiently in hardware.

D. Instruction Set Extension for CS

Our instruction set extension (ISE) performs an accumulation of sample data on randomized memory addresses within a defined buffer. Essentially, lines 3-6 of Algorithm 1 are combined into a single instruction, named Compressed Sensing Accumulation (CSA). The assembler semantic of CSA is: *CSA *Rb, Rs*. As shown in Fig. 1, the CSA instruction takes two general purpose registers as arguments: the first (*Rb*) holding the data memory base address (*b*) of the sensing buffer, the second (*Rs*) containing the sample data (*s*). The CSA instruction addresses a random element (*i*) within the referenced buffer and adds the provided sample onto the existing value in the memory. This operation is repeated for a configured amount of times, by the use of a counter register dedicated to the instruction. With each repetition a new pseudo random element of the buffer is addressed. Since the LFSR state of the *address randomizer* can be directly accessed through the register file, the LFSR hardware can also be used for efficient pseudo random number generation, independent of the CS specific memory addressing and accumulation.

1) Configurability: To enable the construction of many different sensing matrices, the custom instruction is based on four parameters, accessible through dedicated configuration registers. The custom instruction supports software reconfigurability regarding the employed 16-bit LFSR polynomial, LFSR seed, and the required index width used for memory addressing. Additionally the number of non-zero entries per matrix column can be configured, which equals the number of times a sample is applied to the sensing buffer. This configurability amounts to storage requirements of at most 3x 16 bit per sensing matrix.

2) Hardware Implementation: The hardware structure of the *address randomizer* extension is presented in Fig. 1. The custom instruction uses the existing 16-bit adder unit in the ALU and does not introduce any new units in the execution stage of the processor. The decode stage holds the extended address generation logic, which enables addressing of a random word inside the sensing buffer by combining a buffer base address (*b*) with index bits (*i*) taken from the least significant bits of the current LFSR state. The number of index bits depends on the value set in the configuration register. In one cycle, the LFSR state is updated by the same number of LFSR steps as index bits used (1-16). Additionally, the ISE is realized as a multi-cycle instruction, which allows handling of one sample in a number of cycles equal to the configured number of non-zero entries per matrix column.

IV. Power and Performance Results

A. Sub-V_T Synthesis Flow

As aforementioned, many sensing platforms cannot be power gated completely [4], to retain their memory content. Hence, leakage power is always dissipated. Therefore, our sub-V_T CS processor always operates at a clock frequency that barely accomplishes the task on time while lowering supply voltage to the minimum possible level.

The design is synthesized above threshold at a nominal voltage of 1.0 V with a low-power high threshold-voltage 65-nm CMOS technology, which has a threshold voltage $V_T < 700$ mV. Toggling information is obtained by simulating a fully routed design (including clock tree) with back-annotated timing information. The design is characterized by employing the sub-V_T energy characterization model given in [14] which uses parameters retrieved from critical path information as well as a traditional *value change dump* based power simulation. This model provides an accurate energy profile, verified by silicon measurements of previously fabricated sub-V_T circuits.

The achieved minimum for the critical path delay is 5.2 ns. Optimization for maximum frequency in general allows for more voltage scaling of the design. However, leakage and active power increase considerably with hard constrained designs, due to their larger implementation size. Following the strategy proposed in [15], we relax the timing constraint to achieve a design consuming less area, leakage, and active power. Simulation results show that for our design a relaxed constrained of 9 ns gives good power results, while still allowing for considerable voltage scaling.

B. Simulation Results

Fig. 2 shows the power consumption and the corresponding supply voltage of the sub-V_T CS processor for various clock frequencies in the sub-V_T domain. More specifically, the CS processor operates at 0.37 V for a required clock frequency of 100 kHz. As a result, a total power of 288 nW is dissipated, where 27% of the power consumption is due to the leakage power. When the required clock frequency is reduced to 1 kHz through voltage scaling, the sub-V_T CS processor consumes 22.5 nW in total, where the leakage dissipation is now responsible for 98% of the consumption. Indeed, as can be seen from

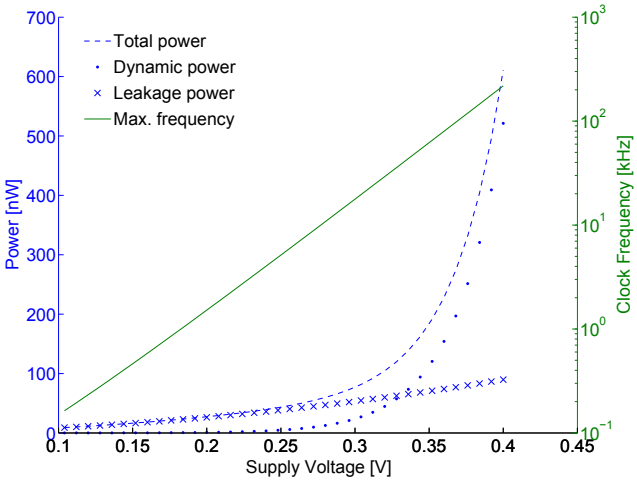

Fig. 2. Power and performance exploration of the sub-V_T CS processor.

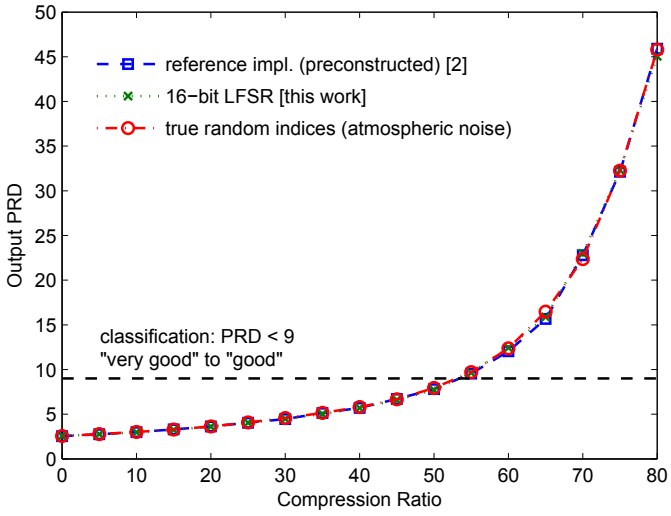

Fig. 3. PRD values at various compression ratios for three index sequences (sensing matrices Φ), each using different methods of construction.

Fig. 2, the leakage power consumption dominates the overall power for clock frequencies lower than 1.5 kHz, corresponding to 0.2 V supply voltage.

The total area of the sub-V_T CS processor is 84.7 kGEs, where 1 GE corresponds to the area of a NAND-2 minimum drive strength gate. The instruction and data memory in the processor have a size of 768 Bytes and 1 kByte, respectively. The memories occupy 84% of the overall area, whereas the core occupies the rest. The area overhead of the CS-extension is less than 3% of the overall area.

C. Case Study: CS-based ECG Signal Compression

As a case study, we apply the CS algorithm for the compression of ECG signals [2]. The test case performs data compression on blocks of 512 samples, recorded at different sampling rates.

1) Quality of Produced Sensing Matrices : Mamaghanian et al. [2] have recently shown that 12 non-zero elements in each column of the sensing matrix are sufficient to maintain satisfactory quality of reconstructed ECG signals for diagnosis purposes. Based on the study in [2], we group random indices into groups of 12, where each group determines the non-zero elements of the corresponding column in the sensing matrix. Assuming that there are no repeated indices in a group, the corresponding column of the sensing matrix will have only ones and zeros. However, in case of repetition the repeated indices will accumulate, which, according to our experiments, does not lead to any quality degradation in the reconstructed signal as shown in Fig. 3 for an example sensing matrix.

To ensure a good quality of diagnostic analysis on the reconstructed ECG signal, the compression performance is quantified according to percentage root-mean-square difference (PRD) for different compression ratios [2]. PRD quantifies the percent error between the original and the reconstructed signal where $PRD < 9$ is classified as "very good" or "good" quality for ECG diagnosis purposes. Thanks to our configurable CS-extension, many sensing matrices with different combination of primitive polynomials and seeds can be constructed. These sensing matrices are analyzed by quantifying their corresponding PRD values for various compression ratios. More specifically, Fig. 3 shows as an example the PRD values with respect to various compression ratios for one of the constructed sensing matrix with a polynomial $p = x^{13} + x^{12} + x^{11} + x^{10} + x^9 + x^7 + x^3 + x^2 + x^1 + 1$ and the seed "0x6218" in hexadecimal combination. As seen from the figure, a PRD value below 10 is retained for compression ratios up to 60%. Moreover, 50% compression is achieved with a PRD of 7.7. Similar to the state-of-the-art CS sensing matrices [2], our produced sensing matrices accomplish a "good" or "very good" quality of the reconstructed signals for a compression ratio less than 60%.

2) Power vs. Performance Analysis: To analyze the power and performance of our sub-V_T CS processor, we consider the example of 50% data compression of ECG signals using the ECG database in [16] for stimuli generation. The required operating frequency to support a given sampling rate to compress ECG signals in real-time is given by: $f \geq N * f_s$ where f_s and N stand for the given sampling rate and the required average number of clock cycles to process a sample. The clock frequency of the sub-V_T CS processor is always adjusted, to have the minimum required clock frequency, according to the given ECG sampling rate. The supply voltage of the processor is then lowered accordingly.

Our sub-V_T CS processor requires 8460 clock cycles to apply 50% compression on 512 samples of ECG data when the sensing matrix is constructed by 12 random indices per column ($\#ones = I = 12$). This corresponds to only an average of $N = 16.5$ cycles processing time for each sample (16 cycles per sample + setup overhead per sample set). As a result, the sub-V_T CS processor must operate with a clock frequency of 2.1 kHz and 16.5 kHz for 125 Hz and 1 kHz sampling rates, respectively. Fig. 4 shows the power consumption of the sub-V_T CS processor for various ECG sampling rates. More specifically, for 125 Hz sampling rate the sub-V_T CS processor

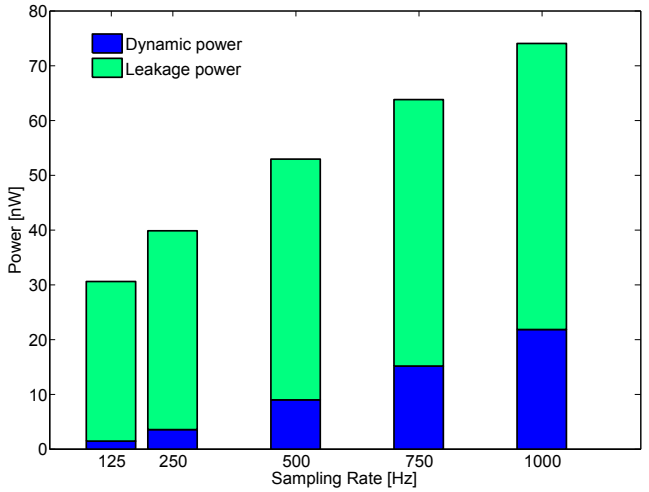

Fig. 4. Power consumption for various ECG sampling rates.

consumes only 30.6 nW in total with 95% of the power due to leakage. Similarly, the total power consumption is only 74 nW for a sampling rate of 1 kHz, where 70.7% is because of leakage dissipation.

For comparison, constructing the CS sensing matrix by computing random sequences of indices based on a pseudo RNG algorithm (c.f. Section II) running on the baseline processor, requires a significantly higher computational effort. The increased computational effort per sample in terms of cycles amounts to $(10log_2(k) + 5)I + 5$, compared to our proposed CSA instruction with an effort of $I + 4$ cycles. In this case of LFSR emulation by software, code optimized to the baseline ISA processes one ECG sample, including the sensing matrix construction, on average in $N = 1025.5$ cycles. Therefore, a sampling rate of $f_s = 125\,Hz$ requires a clock frequency of 128 kHz. Hence the total power consumption of the design would be 355 nW (cf. Fig. 2). This is 11.6x higher than the sub-V_T CS processor with ISE, where the random indices are produced with the help of the embedded LFSR. Moreover, Mamaghanian et al. [2] report a code execution time of 25 ms on a different architecture with a clock frequency of 8 MHz, for applying 51% compression on a set of 512 ECG samples, where pre-computed random indices are stored in the memory. This results in $N = 390.5$ cycles per sample, a 23.6x higher performance requirement than our CS implementation, in terms of cycle count alone.

V. Conclusion

Compressed sensing (CS) is a universal data compression technique applied to sparse signals, used widely for sensing environment applications. Autonomous and portable devices with limited energy efficiency in sensing platforms enforce ultra-low-power CS implementations for these systems. Therefore, we have proposed a sub-threshold processing platform optimized for CS. To this end, we have customized the instruction set of a low-power baseline processor to exploit the specific operations of CS algorithm. Our processing platform requires neither high computational effort nor excessive memory sizes as in straight-forward implementations. Therefore, it is well suited to exploit sub-threshold computing at low voltages and with low leakage. Our processing platform consumes only 30.6 nW for a case study of CS-based electrocardiogram (ECG) signal compression for an ECG sampling rate of 125 Hz. Our results show that the proposed processing platform achieves 62x speed-up and 11.6x power savings with respect to an established CS implementation running on the baseline low-power processor.

References

[1] D. L. Donoho, "Compressed sensing," *IEEE Trans. on Information Theory*, vol. 52, no. 4, pp. 1289–1306, 2006.

[2] H. Mamaghanian, N. Khaled, D. Atienza Alonso, and P. Vandergheynst, "Compressed sensing for real-time energy-efficient ECG compression on wireless body sensor nodes," *IEEE Trans. on Biomedical Engineering*, vol. 58, no. 9, pp. 2456–2466, 2011.

[3] S. C. Jocke, J. F. Bolus, S. N. Wooters, A. D. Jurik, A. C. Weaver, T. N. Blalock, and B. H. Calhoun, "A 2.6-uw sub-threshold mixed-signal ecg soc," in *VLSI Circuits, 2009 Symposium on*, june 2009, pp. 60 –61.

[4] S. Hanson, M. Seok, Y.-S. Lin, Z. Y. Foo, D. Kim, Y. Lee, N. Liu, D. Sylvester, and D. Blaauw, "A low-voltage processor for sensing applications with picowatt standby mode," *Solid-State Circuits, IEEE Journal of*, vol. 44, no. 4, pp. 1145 –1155, april 2009.

[5] J. Kwong, Y. Ramadass, N. Verma, M. Koesler, K. Huber, H. Moormann, and A. Chandrakasan, "A 65nm sub-vt microcontroller with integrated sram and switched-capacitor dc-dc converter," in *Solid-State Circuits Conference, 2008. ISSCC 2008. Digest of Technical Papers. IEEE International*, feb. 2008, pp. 318 –616.

[6] A. Y. Dogan, D. Atienza, A. Burg, I. Loi, and L. Benini, "Power/performance exploration of single-core and multi-core processor approaches for biomedical signal processing," in *Proceedings of the 21st international conference on Integrated circuit and system design: power and timing modeling, optimization, and simulation*, ser. PATMOS'11. Berlin, Heidelberg: Springer-Verlag, 2011, p. 102111. [Online]. Available: http://dl.acm.org/citation.cfm?id=2045364.2045375

[7] S. Hanson, B. Zhai, M. Seok, B. Cline, K. Zhou, M. Singhal, M. Minuth, J. Olson, L. Nazhandali, T. Austin, D. Sylvester, and D. Blaauw, "Exploring variability and performance in a sub-200-mV processor," *IEEE J. Solid-State Circuits*, vol. 43, no. 4, pp. 881–891, April 2008.

[8] B. Zhai, L. Nazhandali, J. Olson, A. Reeves, M. Minuth, R. Helfand, S. Pant, D. Blaauw, and T. Austin, "A 2.60pJ/Inst subthreshold sensor processor for optimal energy efficiency," in *Proc. IEEE Symp. VLSI Circuits*, 2006, pp. 154–155.

[9] R. Dreslinski, M. Wieckowski, D. Blaauw, D. Sylvester, and T. Mudge, "Near-threshold computing: Reclaiming Moore's law through energy efficient integrated circuits," *Proceedings of the IEEE*, vol. 98, no. 2, pp. 253–266, Feb. 2010.

[10] C. Banz, C. Dolar, F. Cholewa, and H. Blume, "Instruction set extension for high throughput disparity estimation in stereo image processing," in *Proc. IEEE Int. Conf. on Application-Specific Systems, Architectures and Processors*, 2011, pp. 169 –175.

[11] J. Kwong and A. Chandrakasan, "An energy-efficient biomedical signal processing platform," *IEEE J. Solid-State Circuits*, vol. 46, no. 7, pp. 1742–1753, July 2011.

[12] A. Y. Dogan, C. J., R. M., B. A., and A. D., "Multi-core architecture design for ultra-low-power wearable health monitoring systems," in *Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2012.

[13] P. Meinerzhagen, S. Sherazi, A. Burg, and J. Rodrigues, "Benchmarking of standard-cell based memories in the sub-V_t domain in 65-nm CMOS technology," *IEEE J. on Emerging and Selected Topics in Circuits and Systems*, vol. 1, no. 2, pp. 173–182, June 2011.

[14] O. Akgun, J. Rodrigues, Y. Leblebici, and V. Öwall, "High-level energy estimation in the sub-V_t domain: Simulation and measurement of a cardiac event detector," *IEEE Trans. Biomedical Circuits and Systems*, vol. 6, no. 1, pp. 15–27, Feb. 2012.

[15] P. Meinerzhagen, O. Andersson, Y. Sherazi, A. Burg, and J. Rodrigues, "Synthesis strategies for sub-V_t systems," in *Proc. IEEE European Conf. on Circuit Theory and Design*, Aug. 2011, pp. 552–555.

[16] H.-M. D. of Health Sciences and T. B. E. Center, "MIT-BIH arrhythmia database directory," http://www.physionet.org/physiobank/database/mitdb.