An Ultra-Low-Power Application-Specific Processor with Sub-$\mathrm{V_T}$ Memories for Compressed Sensing

Jeremy Constantin[1], Ahmed Dogan[1], Oskar Andersson[2], Pascal Meinerzhagen[1], Joachim Rodrigues[2], David Atienza[1], and Andreas Burg[1,⋆]

[1] École polytechnique fédérale de Lausanne, VD-1015 Lausanne, Switzerland, Institute of Electrical Engineering
`jeremy.constantin@epfl.ch`
[2] Lund University, 22100 Lund, Sweden, Department of Electrical and Information Technology

Abstract. Compressed sensing (CS) is a universal low-complexity data compression technique for signals that have a sparse representation in some domain. While CS data compression can be done both in the analog- and digital domain, digital implementations are often used on low-power sensor nodes, where an ultra-low-power (ULP) processor carries out the algorithm on Nyquist-rate sampled data. In such systems an energy-efficient implementation of the CS compression kernel is a vital ingredient to maximize battery lifetime. In this paper, we propose an application-specific instruction-set processor (ASIP) processor that has been optimized for CS data compression and for operation in the sub-threshold (sub-$\mathrm{V_T}$) regime. The design is equipped with specific sub-$\mathrm{V_T}$ capable standard-cell based memories, to enable low-voltage operation with low leakage. Our results show that the proposed ASIP accomplishes 62× speed-up and 11.6× power savings with respect to a straightforward CS implementation running on the baseline low-power processor without instruction set extensions.

Keywords: Ultra-Low-Power Processor, Application-Specific Instruction Set Processor, Instruction Set Extensions, Sub-$\mathrm{V_T}$ Operation, Sub-$\mathrm{V_T}$ Embedded Memories, Compressed Sensing.

1 Introduction

Digital signal processing traditionally relies on the Nyquist sampling theorem which states that a faithful reconstruction of a signal, limited to a bandwidth B in the frequency spectrum, can be ensured with a sampling rate of $f_s \geq 2 * B$. Unfortunately, when sampled data needs to be stored or needs to be transmitted over a wireless link, the storage or transmission costs of the raw samples can often

⋆ This chapter extends the work published earlier in [1].

limit the energy-autonomous lifetime of the system. In this case, it is advisable to first compress the data. However, in this case the power consumption of the compression process must also be kept very low to ensure an overall energy-efficiency advantage.

Compressed sensing (CS) [2] is a universal, low-complexity data compression technique to compress sparse signals. CS has been widely used in environmental monitoring systems and in wireless body sensor networks (WBSNs) [3], 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 systems.

On the architectural level, supply voltage scaling, potentially all the way to the subthreshold (sub-V_T) regime, can reduce both dynamic and leakage power consumption. Therefore, 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 [4–6]. Sub-V_T computing can also be used to perform CS data compression (in the digital domain). However, most established CS implementations either require a large memory footprint or still require considerable computational effort (despite the inherent complexity advantage of CS). Leakage power consumption becomes a very important challenge in the sub-V_T regime with reduced active power. A considerable amount of leakage in sensing platforms is due to the integrated memories [7]. Moreover, many sensing platforms cannot be power gated completely, to retain their memory content [5], 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 [8–10]. These issues ultimately limit currently the benefits of CS based data compression in ULP sensor nodes.

Application-specific instruction-set processors (ASIPs) can compensate for the 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 130$\times$ speed-up compared to a conventional processor [11]. These performance optimizations also lead to energy saving as in [12], where a processing core with few accelerators dedicated to biomedical applications, can achieve up to 11.5$\times$ 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 ultra-low-power CS compression.

Contributions. We propose to synergistically exploit sub-V_T computing in conjunction with an ASIP core for CS compression to provide an ultra-low-power solution for compression of sparse signals for sensing 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 does not require high clock frequencies, and therefore enables more aggressive sub-V_T voltage scaling for a given throughput requirement. The very low memory requirements additionally allow for a major reduction in leakage power. For a typical case study of electrocardiogram (ECG) signal compression in WBSNs, the processor consumes only 30.6 nW for an ECG sampling rate of 125 Hz. Moreover, we show that the proposed processing platform achieves 62× speed-up and 11.6× power saving with respect to the established computation-based CS implementation running on the baseline low-power processor.

2 Compressed Sensing

Signal compression based on compressed sensing (CS) [2] is performed by computing the matrix-vector multiplication:

$$y = \mathbf{\Phi} x \tag{1}$$

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, for 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 choosing the entries of $\mathbf{\Phi}$ by random iid sampling from a uniform distribution [2].

2.1 Reduced Complexity Compression Algorithm

The structure and values of the entries of $\mathbf{\Phi}$ determine the computational complexity of the matrix-vector multiplication. Mamaghanian et al. [3] show (for WBSNs) that in fact choosing $\mathbf{\Phi}$ as a sparse matrix that contains only a few non-zero entries per column at random positions is a valid approach which significantly reduces complexity and still provides good integrity of the compressed sparse signals. The non-zero elements can furthermore be chosen as 1, and the number of ones per column (namely I) can be fixed. These constraints on $\mathbf{\Phi}$ lead to a very efficient algorithm (Algorithm 1) for performing CS data compression. As a result, the computational complexity of the CS algorithm is reduced from $n{\times}k$ multiplications and $(n-1){\times}k$ additions for a dense sensing matrix of random values, to only $I \times n$ additions. The sensing matrix can therefore be represented in a compact form by a sequence of $I{\times}n$ random indices $\in \{1, 2, ..., k\}$ describing for each column in $\mathbf{\Phi}$ the rows with non-zero entries[1].

On a resource constrained system, the key challenge of Algorithm 1 is the generation of the random indices. The optimized reference implementation [3] uses a sensing matrix realized as a fixed sequence of indices stored in memory (for a specific value of k). Since large memory footprints are undesirable, especially in the context of ULP sensor nodes and sub-$\mathrm{V_T}$ operation, we discuss the generation of the required random indices at runtime.

[1] Note that strictly speaking such a representation requires unique row-indices per column. However, this requirement can often be relaxed without a significant impact.

Algorithm 1. Pseudocode of Compressed Sensing Algorithm

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

2.2 Pseudo Random Number Generation

A pseudo random number generator (RNG) can be used for the generation of the random indices. 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 referred to as its 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 to produce one new bit (parity 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.

Maximum-length LFSRs provide a cycle length of the generated random number sequence that 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., subsequent 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. 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 only a small correlation to its predecessor. The quality of our generated random indices for CS is assessed in the case study presented in Section 4.4. 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 hence be described 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 (I). These four configuration parameters enable the generation of a large set of different sensing matrices. At the same time, the compact representation of the RNG configuration keeps the memory overhead small compared to the case where all indices for multiple matrices would need to be stored.

Hence, by choosing from a preconstructed pool of feasible values for the RNG configuration, it is hence possible to achieve good sensing performance for a variety of different signal conditions, potentially even by dynamically changing the RNG configuration at runtime. This capability supports one of the strength of the compressed sensing method which lies in the fact that even a randomly chosen sensing matrix $\mathbf{\Phi}$ ensures a good mapping for the sample data of a signal source which has sparsity in a specific (potentially) unknown base with high probability. On the contrary, any CS implementation using only a single or a very small number of pre-stored sensing matrices loses its generality to perform well independent of the signal source. To alleviate this issue, our approach therefore tries to minimize all related storage and memory costs to support a large number of different random sensing matrices, which can be dynamically generated at runtime.

3 Sub-V_T CS Processor

Resource constrained environments, such as ULP processing nodes, pose significant challenges for the implementation of the presented data compression algorithm (Algorithm 1). The key performance issue 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.

3.1 Processor Baseline Architecture

In this study we use a custom 16-bit reduced instruction set computing (RISC) architecture (TamaRISC [13]), as shown in Figure 1, as the baseline microprocessor. TamaRISC provides a complete RISC instruction set, a C-Compiler, as well as interrupt capability for basic embedded real-time operating system support.

Core Architecture. The main focuses of the architecture lies 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 microarchitecture 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.

Moreover, the TamaRISC architecture supports memory-to-memory arithmetic instructions with advanced operand addressing modes, and is hence not a typical load/store RISC architecture, but rather inspired by typical microcontroller architectures.

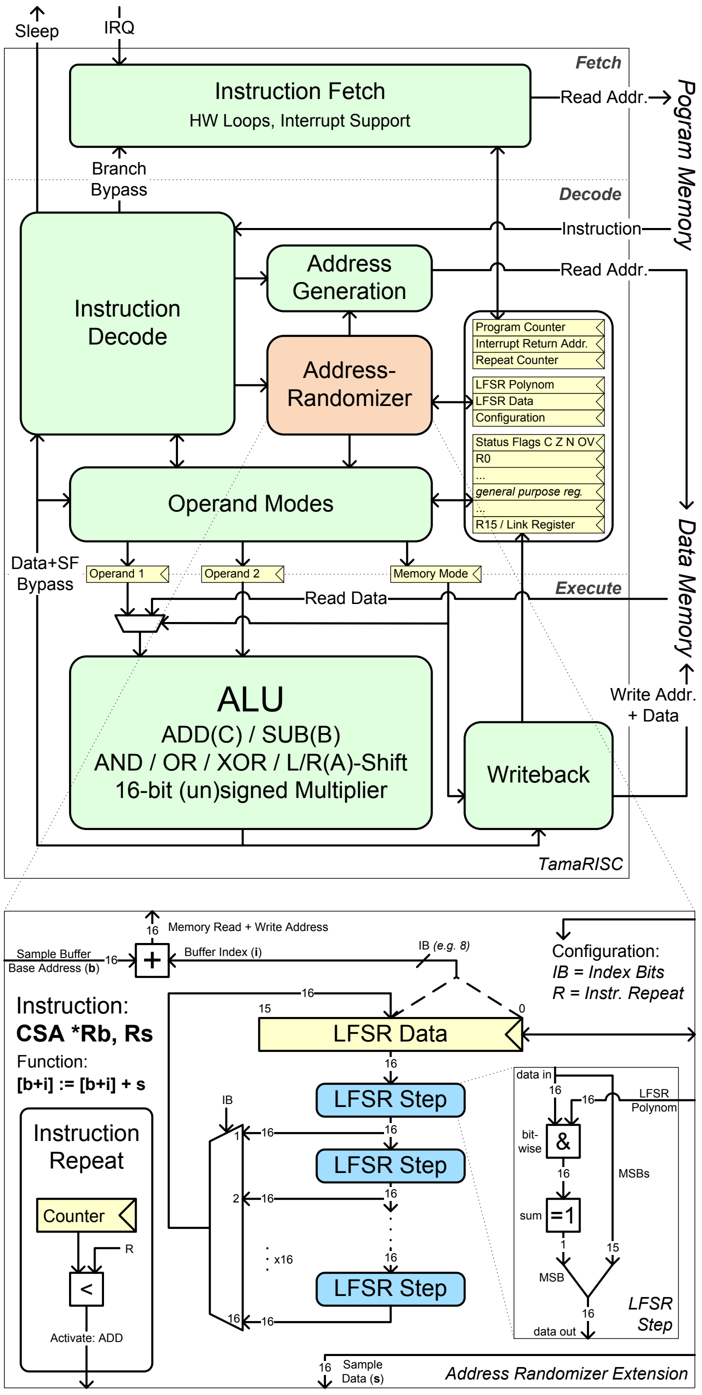

Fig. 1. TamaRISC sub-V_T microprocessor architecture including address-randomizer extension for CS

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 two source and one destination 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 interrupt request triggered by a new ADC sample).

3.2 Sub-V_T Memories

While the core logic of the sub-V_T CS processor works reliably at low supply voltages in the sub-V_T regime, conventional data and instruction memories based on 6-transistor (6T) static random-access memory (SRAM) bitcells fail to operate reliably at low voltages [14]. Therefore, such conventional, embedded 6T SRAM macrocells prohibitively limit the overall reliability and the manufacturing yield of the proposed sub-V_T CS processor. More precisely, under gradual supply voltage down-scaling, read and write access failures start to appear first, before the occurrence of data retention failures at even lower voltages [15]. Specially designed SRAM macrocells based on 8- or 10-transistor (8-10T) bitcells are typically used to enable reliable data storage in the sub-V_T regime. For example, a typical 8T SRAM cell contains a read buffer to avoid the direct access of the bit lines to the internal storage nodes and consequently to avoid the risk of switching the bitcell during a read access [16], thereby improving read-ability. Moreover, a popular 10T SRAM bitcell contains, in addition to the read buffer, a tri-state inverter in the cross-coupled latch; this tri-state inverter is disabled during a write access in order to avoid write contention [17], thereby improving write-ability. All these 8T or 10T SRAM macrocells, specifically optimized for robust sub-V_T operation, need to be custom-designed due to the lack of good, commercially available low-voltage memory compilers. Such custom design is associated with a high engineering effort and bares high risk, unless each macrocell is first manufactured and silicon-proven independently, before its integration into a larger VLSI system.

As opposed to such custom-designed sub-V_T 8T/10T SRAM macrocells, we employ a fully automated standard-cell based memory (SCM) compilation flow [18]. The use of SCMs considerably simplifies the design process, and the

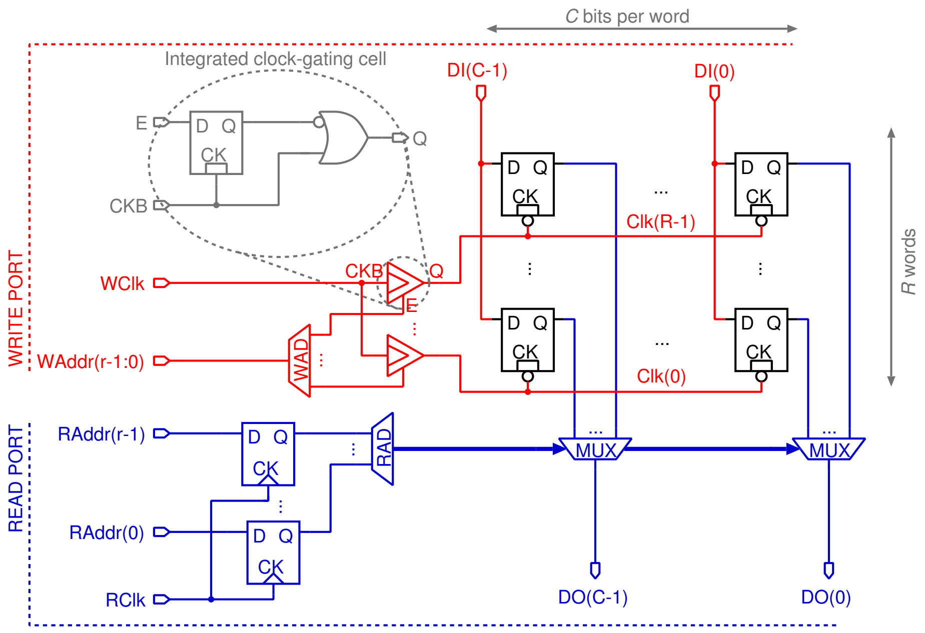

Fig. 2. Schematic of the latch array with clock-gates for the generation of write select signals and static CMOS readout multiplexers. The write port is highlighted in red, while the read port is highlighted in blue.

resulting latch or flip-flop arrays directly avoid the aforementioned reliability concerns of conventional 6T SRAM. In particular, standard-cell latches already contain a read buffer to avoid read failures and a cell-internal keeper which is disabled during write, i.e., during the transparent phase of the latch, to avoid the risk of write contention and write failures. Consequently, the proposed SCMs work reliably in the sub-V_T regime without the need for any extra engineering effort, and allow the complete system to operate at aggressively scaled voltages. Among many architectural variants summarized in [18], this work adopts the latch array architecture shown in Fig. 2. This architecture consists of a write address decoder (WAD) and clock gates for the generation of the one-hot encoded write select pulses (row-wise gated clock signals). Moreover, static CMOS multiplexers are used to read out the desired address (word). While a read logic based on tri-state buffers exhibits lower leakage current, the chosen CMOS multiplexers are faster and more robust for sub-V_T operation. This latch array architecture can be synthesized from commercially available standard-cell libraries. However, note that it is possible to customize one or several standard-cells to meet a specific design goal, such as ultra-low leakage power. For example, the leakage power and access energy can be reduced by approximately 50% by using a single custom-designed standard-cell, namely an ultra-low leakage latch using stack forcing and channel length stretching, as well as a tri-state enabled output buffer to implement the read logic [19].

Even though these latch-based memories are optimized for low-voltage and low-power operation, they still consume considerable leakage power. In our system example, memories account for 70–95% of the architecture's total power consumption, depending on the mode of operation. Furthermore, the sub-V_T memories consume a considerable area share: our implementation with moderate memory sizes of 256 instructions (6 kBit) and 512 data words (8 kBit) results in the processing core only consuming 16% of the total area.

3.3 Index Sequence Implementations

As discussed before, the generation of random numbers used as the buffer indices in Algorithm 1 is commonly performed using one of the following two approaches. The first approach employs precomputation and storage of all required indices in form of a large array in data memory, while the second approach performs the computation of the index sequence at runtime based on a pseudo RNG.

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.

Computation at Runtime. The generation of suitable random indices can also be performed by sequence computation based on pseudo RNGs, such as the algorithm proposed in Section 2.2. This approach only requires the data memory to comprise the sensing buffer, which for compression of a set of 512 samples by 50% equals 256 data words (e.g., 512 bytes with a sample precision of 12 (up to 16) bit). As shown in Section 2.2, 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 12×8 steps per sample, i.e., a computational requirement of about 1 kOp per sample, dedicated to the task of random number generation alone. This requirement becomes problematic, since downscaling of the supply voltage considerably limits the maximum core clock frequency (cf. Figure 3). 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 task of pseudo-random index generation efficiently in hardware.

3.4 Instruction Set Extension for CS

Analysis of the CS kernel loop shows that the extension of memory operand addressing with efficient randomization can result in significant performance gains. We hence introduce an extension to the TamaRISC instruction set architecture, adding a new instruction that 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 number of iterations, 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, independently of the CS specific memory addressing and accumulation.

Moreover, the CSA instruction is generally used in a small loop in conjunction with the processor's sleep mode, which puts the core in a dormant state (clock-gated) to significantly reduce power consumption until new sample data is available. On wakeup by an interrupt request, the sample data is fetched from the ADC and the CSA instruction is executed, after which the core can immediately be put to sleep again.

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, the 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 added to pseudo random locations of the sensing buffer. This configurability amounts to storage requirements of at most three 16-bit values per sensing matrix.

Hardware Implementation. The internal hardware structure of the *address randomizer* extension to the TamaRISC micro-architecture is presented in Fig. 1. The custom instruction employs for the sample accumulation the existing 16-bit adder unit in the ALU and does not introduce any new units to the data path of 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 instruction set extension (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 (I) per matrix column.

4 Power and Performance Results

Due to the need to retain their memory content, many sensing platforms can not be power gated completely [5], and hence, leakage power is always dissipated. Therefore, our sub-$\mathrm{V_T}$ CS processor always operates at a clock frequency that barely accomplishes the task on time while lowering the supply voltage to the corresponding minimum possible level that avoids timing violations. Note that the objective is to minimize power for a given workload in contrast to the operation at the energy-minimum-voltage (EMV), where maximizing energy efficiency often requires a higher operating voltage to balance leakage and active energy.

4.1 Synthesis and Energy Profiling

The design is synthesized above threshold at nominal supply voltage of 1.0 V with a low-power high threshold-voltage 65-nm CMOS technology, which has a threshold voltage $\mathrm{V_T} \leq 700\,\mathrm{mV}$. 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-$\mathrm{V_T}$ energy characterization model that has been derived in [20] and that is briefly introduced in the next subsection. With this model, parameters retrieved from critical path information as well as a traditional *value change dump* based power simulation are used to compute maximum operational speed, energy and power dissipation in the sub-$\mathrm{V_T}$ region.

In our implementation, the post-layout critical path delay at nominal supply voltage is 5.2 ns, according to the gate-level static timing analysis. Optimization for maximum frequency and thus a larger slack on the critical path allows for a more aggressive voltage scaling. However, leakage and active power increase considerably with hard timing-constrained designs. Tight constraints will force the tool to infer nets with high fan-out as well as stronger buffers, which increases capacitance on the critical path and consequently yield a slower operation in the sub-$\mathrm{V_T}$ region. Following the strategy proposed in [21], we relax the timing constraint to achieve a design with low area and leakage cost. Simulation results show that a relaxed timing constraint of 9 ns, at nominal supply voltage, gives good power results, while it still enables for aggressive voltage scaling for our target applications.

4.2 Sub-$\mathrm{V_T}$ Energy Profiling

The total energy dissipation E_T of static CMOS circuits operated in the sub-$\mathrm{V_T}$ regime is modelled as

$$E_\mathrm{T} = \underbrace{\alpha C_\mathrm{tot} {V_\mathrm{DD}}^2}_{E_\mathrm{dyn}} + \underbrace{I_\mathrm{leak} V_\mathrm{DD} T_\mathrm{clk}}_{E_\mathrm{leak}} + \underbrace{I_\mathrm{peak} t_\mathrm{sc} V_\mathrm{DD}}_{E_\mathrm{sc}}, \quad (2)$$

where $\mathrm{E_{dyn}}$, $\mathrm{E_{leak}}$, and $\mathrm{E_{sc}}$ are the dynamic, leakage, and short-circuit energy, respectively. The energy dissipation $\mathrm{E_{sc}}$ has been shown to be negligible in the sub-$\mathrm{V_T}$ regime [22]. The switching current causing the energy dissipation $\mathrm{E_{dyn}}$ results from subthreshold currents [23], i.e., from the drain currents of MOS transistors whose gate-to-source voltage $\mathrm{V_{GS}}$ is equal to or lower than the threshold voltage $\mathrm{V_T}$ ($\mathrm{V_{GS}} \leq \mathrm{V_T}$). Whenever the subthreshold current is not used to switch a circuit node, it contributes to $\mathrm{E_{leak}}$. For a given clock period $\mathrm{T_{clk}}$, (2) may be rewritten as

$$E_\mathrm{T} = \mu_\mathrm{e} C_\mathrm{inv} k_\mathrm{cap} {V_\mathrm{DD}}^2 + k_\mathrm{leak} I_0 V_\mathrm{DD} T_\mathrm{clk}, \tag{3}$$

where $\mathrm{I_0}$ and $\mathrm{C_{inv}}$ represent average leakage current and input capacitance of a single inverter, respectively. Furthermore, k_leak and k_cap are the average leakage and capacitance of the circuit, respectively, both normalized to a single inverter. Moreover, μ_e represents the circuit's average switching activity.

In the sub-$\mathrm{V_T}$ domain, it is beneficial to operate at the maximum achievable frequency to reach minimum energy dissipation per operation. The critical path delay of the circuit is given by

$$T_\mathrm{clk} = k_\mathrm{crit} \frac{C_\mathrm{inv} V_\mathrm{DD}}{I_0 \mathrm{e}^{V_\mathrm{DD}/(n V_\mathrm{T})}}, \tag{4}$$

where k_crit is the critical path delay of the circuit normalized to an inverter delay, $V_\mathrm{T} = kT/q$ is the thermal voltage and n is the subthreshold factor. By assuming that operation is performed at the maximum frequency, the total energy dissipation $\mathrm{E_T}$ is found by introducing (4) into (3), which gives

$$E_\mathrm{T} = C_\mathrm{inv} {V_\mathrm{DD}}^2 \left[\mu_\mathrm{e} k_\mathrm{cap} + k_\mathrm{crit} k_\mathrm{leak} \mathrm{e}^{-V_\mathrm{DD}/(n V_\mathrm{T})} \right]. \tag{5}$$

Additionally, for a system that operates at a fixed frequency with a given clock period $\mathrm{T_{clk}}$, at a given $\mathrm{V_{DD}}$, the power is derived from (3) as

$$P = \frac{\mu_\mathrm{e} C_\mathrm{inv} k_\mathrm{cap} {V_\mathrm{DD}}^2}{T_\mathrm{clk}} + k_\mathrm{leak} I_0 V_\mathrm{DD}, \tag{6}$$

which shows that the static power consumption is directly proportional to $\mathrm{V_{DD}}$.

The application of the model provides the sub-$\mathrm{V_T}$ profile of a design, i.e., energy/operation, power consumption, and critical path speed. The model was validated by measurements and accuracy is within a 10% error rate at the measured temperatures 0, 27 and 37 °C. In Table 1 the design properties of the sub-$\mathrm{V_T}$ processor are shown in terms of leakage, capacitance and critical path normalized to a single inverter, as well as switching activity. For more details, the reader is referred to [20].

4.3 Simulation Results

Fig. 3 shows the power consumption and the corresponding supply voltage of the sub-$\mathrm{V_T}$ CS processor for various clock frequencies in the sub-$\mathrm{V_T}$ domain,

Table 1. Architectural properties for sub-V_T modeling

Design properties	Value
k_{cap}	254 000
k_{crit}	434
k_{leak}	194 000
μ_e	0.0675

computed using (6) and (4). More specifically, a clock frequency of 100 kHz for the CS processor is achieved at a supply voltage of 0.37 V. As a result, a total power of 288 nW is dissipated, where 27% of the power consumption is due to 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 now has a share of 98%. As demonstrated in Fig. 3, the leakage power dominates the overall power for clock frequencies lower than 1.5 kHz, corresponding to 0.2 V supply voltage. In this particular technology, operation below 0.25 V is not recommended due to higher rates of functional failures from larger process variations according to [20].

The energy profile of the sub-V_T CS processor is shown in Fig. 4. Energy dissipation at maximum operational frequency (5) is shown together with fixed clock frequencies (3) of 2.1 kHz, 16.5 kHz and 100 kHz. It is observed that operating at lower frequency than dictated by supply voltage results in higher

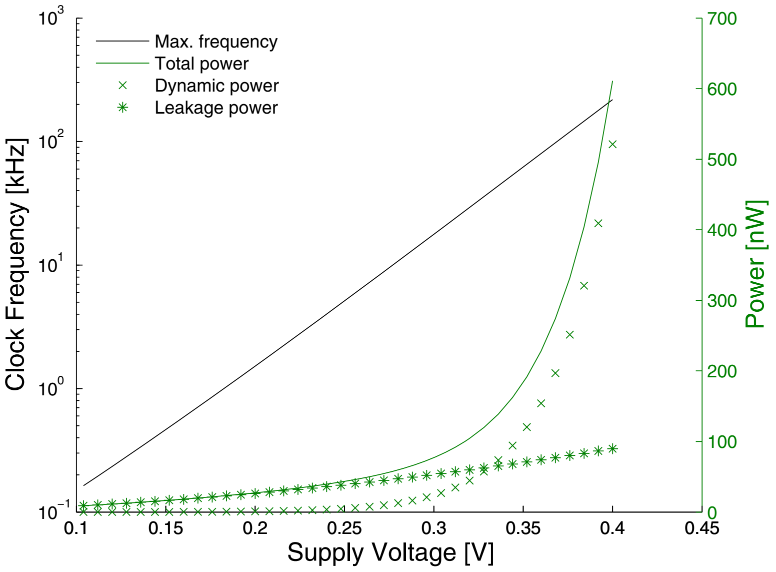

Fig. 3. Power and performance exploration of the sub-V_T CS processor

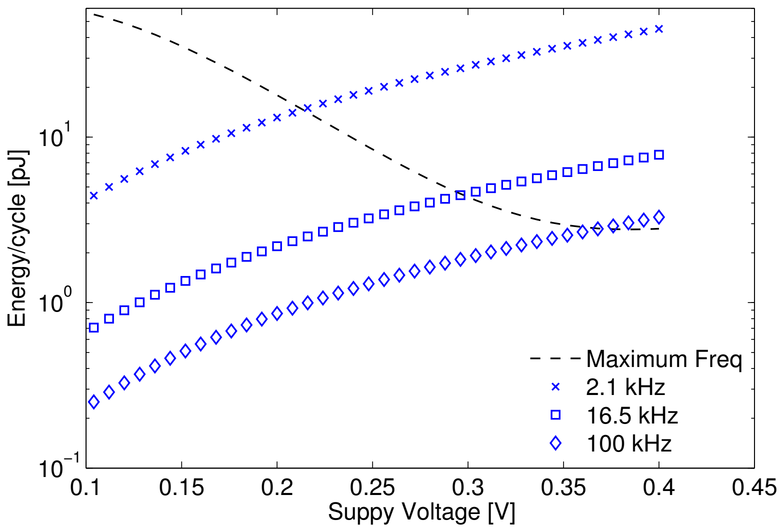

Fig. 4. Energy profile for various operational frequencies of the sub-V$_T$ CS processor

energy dissipation. Thus the implementation goal is to use a supply voltage that is just barely sufficient to support the necessary clock frequency.

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 instruction set extension for CS accounts for less than 3% of the overall area.

4.4 Case Study: CS-Based ECG Signal Compression

As a case study, we apply the CS algorithm for the compression of ECG signals [3]. The test case performs data compression on blocks of 512 samples, recorded at different sampling rates.

Quality of Produced Sensing Matrices. Mamaghanian et al. [3] have shown that 12 non-zero elements in each column of the sensing matrix are sufficient to maintain satisfactory quality of reconstructed ECG signals for diagnostic purposes. Based on the study in [3], 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. 5 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 the percentage root-mean-square difference (PRD) for different compression ratios [3]. PRD quantifies the percent error between the original and the reconstructed signal where a PRD value less than 9 is classified as "very good" or "good" quality for ECG diagnosis. 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. 5 shows as an example the PRD values with respect to various compression ratios for one of the constructed sensing matrices 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 Fig. 5, a PRD value below 10 is retained for compression ratios up to almost 60%. Moreover, 50% compression is achieved with a PRD of 7.7. Similar to the state-of-the-art CS sensing matrices [3], our sensing matrices that are generated by our multi-step LFSR mechanism, accomplish a "good" or "very good" quality of the reconstructed signals for compression ratios less than 53%.

Power vs. Performance Analysis. We consider the example of 50% data compression of ECG signals, using the ECG database in [24] for stimuli generation, to analyze the power and performance of our sub-$\mathrm{V_T}$ CS processor. 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-$\mathrm{V_T}$ 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.

The presented sub-$\mathrm{V_T}$ 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 ($I = 12$). 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-$\mathrm{V_T}$ 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. 6 shows the power consumption of the sub-$\mathrm{V_T}$ CS processor for various ECG sampling rates. More specifically, for 125 Hz sampling rate the sub-$\mathrm{V_T}$ CS processor 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.

To compare our ISE-enhanced CS processor with the baseline processor, we consider the construction of the CS sensing matrix by computing random sequences of indices based on a pseudo RNG algorithm (c.f. Section 2) running on the baseline ISA. Our results show that the optimized implementation for the baseline core requires a significantly higher computational effort. Specifically,

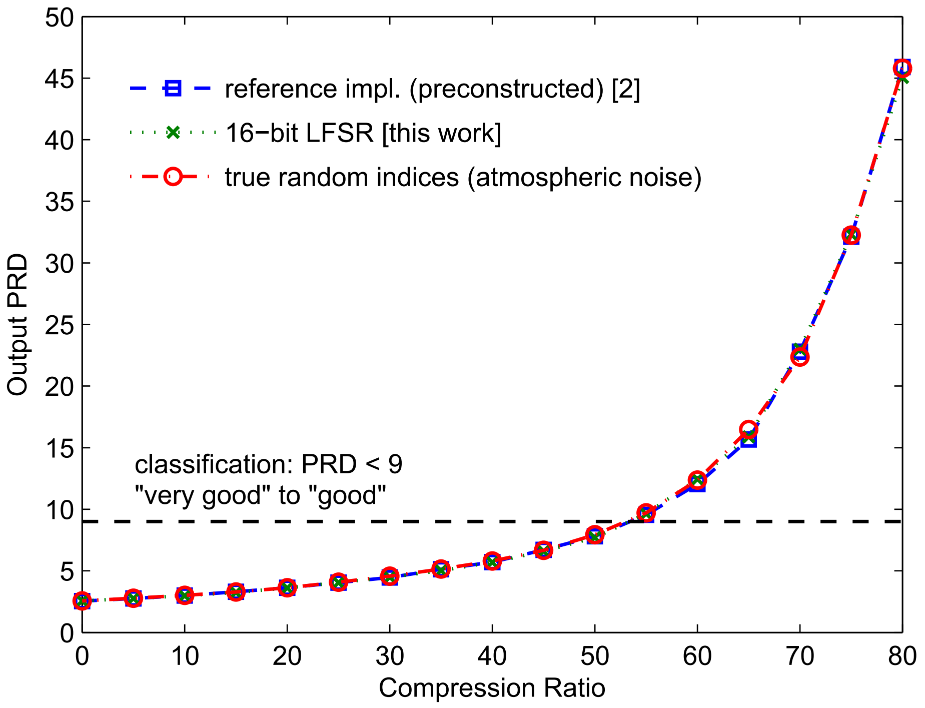

Fig. 5. PRD values at various compression ratios for three index sequences (sensing matrices Φ), each using different methods of construction

the increased computational effort per sample in terms of cycles amounts to $(10log_2(k) + 5)I + 5$, compared to our implementation based on the 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, which translates into a speed-up of $62\times$ for our ISE-supported implementation. Therefore, a sampling rate of $f_s = 125\,\text{Hz}$ requires a clock frequency of 128 kHz, using the pure software approach. This results in a total power consumption for the design of 355 nW (cf. Fig. 3), which is $11.6\times$ 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. [3] 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 $23.6\times$ higher performance requirement than our CS implementation, in terms of cycle count alone.

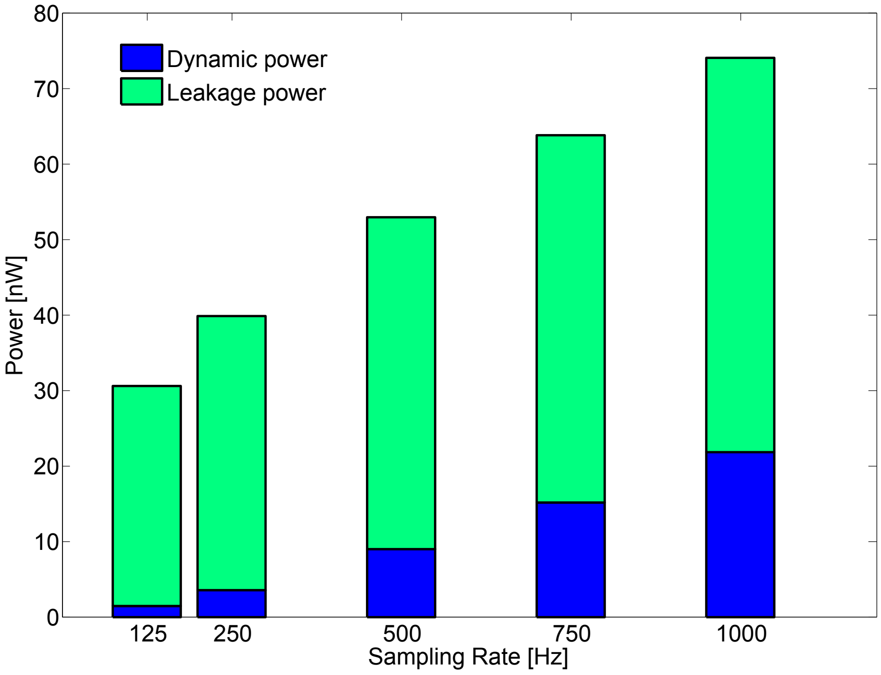

Fig. 6. Power consumption for various ECG sampling rates

5 Conclusion

Compressed sensing (CS) is a well-known universal data compression technique applied to sparse signals, used widely for sensing environment applications. Autonomous and portable devices, such as sensing platforms, however enforce ultra-low-power CS implementations, due to their limited energy resources. Therefore, we have proposed a subthreshold processing platform specifically optimized for CS, while still maintaining the flexibility and configurability of a processor based system. To this end, we have customized the instruction set architecture of a low-power baseline processor to exploit the specific operations of the CS algorithm. Specifically, we propose a Compressed Sensing Accumulation (CSA) instruction that efficiently performs accumulation of sample data on randomized memory addresses within a defined sampling buffer. Moreover, our processing platform embeds the required data and instruction memories in the form of sub-V_T-capable standard-cell memories, which are essential for ULP operation. We show that our processing platform requires neither high computational effort nor excessive memory sizes compared to straight-forward implementations. Therefore, the platform is well suited to exploit subthreshold computing at low voltages and with very low leakage. Our system consumes only 30.6 nW for a case study of CS-based electrocardiogram (ECG) signal compression at an ECG sampling rate of 125 Hz. Our results show that the proposed processing platform achieves 62× speed-up and 11.6× power savings with respect to an established CS implementation running on the baseline low-power processor.

Acknowledgment. This work has been partially funded by the BodyPowered-SenSE Nano-Tera.ch RTD project (ref. number: 20NA21_143069), which is evaluated by the Swiss NSF and funded by Nano-Tera.ch with Swiss Confederation financing.

References

1. Constantin, J., Dogan, A., Andersson, O., Meinerzhagen, P., Rodrigues, J., Atienza, D., Burg, A.: TamaRISC-CS: An ultra-low-power application-specific processor for compressed sensing. In: 2012 IEEE/IFIP 20th International Conference on VLSI and System-on-Chip (VLSI-SoC), pp. 159–164 (2012)
2. Donoho, D.L.: Compressed sensing. IEEE Trans. on Information Theory 52(4), 1289–1306 (2006)
3. Mamaghanian, H., et al.: Compressed sensing for real-time energy-efficient ECG compression on wireless body sensor nodes. IEEE TBME 58(9), 2456–2466 (2011)
4. Jocke, S.C., et al.: A 2.6-uw sub-threshold mixed-signal ecg soc. In: Symposium on VLSI Circuits, pp. 60–61 (2009)
5. Hanson, S., et al.: A low-voltage processor for sensing applications with picowatt standby mode. IEEE J. Solid-State Circuits 44(4), 1145–1155 (2009)
6. Kwong, J., et al.: A 65nm sub-vt microcontroller with integrated sram and switched-capacitor dc-dc converter. In: ISSCC, pp. 318–616 (2008)
7. Dogan, A.Y., Atienza, D., Burg, A., Loi, I., Benini, L.: Power/Performance exploration of single-core and multi-core processor approaches for biomedical signal processing. In: Ayala, J.L., García-Cámara, B., Prieto, M., Ruggiero, M., Sicard, G. (eds.) PATMOS 2011. LNCS, vol. 6951, pp. 102–111. Springer, Heidelberg (2011)
8. Hanson, S., et al.: Exploring variability and performance in a sub-200-mV processor. IEEE J. Solid-State Circuits 43(4), 881–891 (2008)
9. Zhai, B., et al.: A 2.60pJ/Inst subthreshold sensor processor for optimal energy efficiency. In: IEEE VLSI, pp. 154–155 (2006)
10. Dreslinski, R., et al.: Near-threshold computing: Reclaiming Moore's law through energy efficient integrated circuits. Proc. IEEE 98(2), 253–266 (2010)
11. Banz, C., et al.: Instruction set extension for high throughput disparity estimation in stereo image processing. In: ASAP, pp. 169–175 (September 2011)
12. Kwong, J., Chandrakasan, A.: An energy-efficient biomedical signal processing platform. IEEE J. Solid-State Circuits 46(7), 1742–1753 (2011)
13. Dogan, A.Y., et al.: Multi-core architecture design for ultra-low-power wearable health monitoring systems. In: DATE (2012)
14. Qazi, M., Sinangil, M., Chandrakasan, A.: Challenges and directions for low-voltage SRAM. IEEE Design and Test of Computers 28(1), 32–43 (2011)
15. Mukhopadhyay, S., Mahmoodi, H., Roy, K.: Modeling of failure probability and statistical design of SRAM array for yield enhancement in nanoscaled CMOS. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 24(12), 1859–1880

16. Chang, L., Fried, D., Hergenrother, J., Sleight, J., Dennard, R., Montoye, R., Sekaric, L., McNab, S., Topol, A., Adams, C., Guarini, K., Haensch, W.: Stable SRAM cell design for the 32 nm node and beyond. In: 2005 Symposium on VLSI Technology. Digest of Technical Papers pp. 128–129 (June 2005)
17. Jain, S., Khare, S., Yada, S., Ambili, V., Salihundam, P., Ramani, S., Muthukumar, S., Srinivasan, M., Kumar, A., Gb, S., Ramanarayanan, R., Erraguntla, V., Howard, J., Vangal, S., Dighe, S., Ruhl, G., Aseron, P., Wilson, H., Borkar, N., De, V., Borkar, S.: A 280mV-to-1.2V wide-operating-range IA-32 processor in 32nm CMOS. In: 2012 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC), pp. 66–68 (February 2012)
18. Meinerzhagen, P., et al.: Benchmarking of standard-cell based memories in the sub-V_t domain in 65-nm CMOS technology. JETCAS 1(2), 173–182 (2011)
19. Meinerzhagen, P., Andersson, O., Mohammadi, B., Sherazi, Y., Burg, A., Rodrigues, J.: A 500 fW/bit 14 fJ/bit-access 4kb standard-cell based sub-VT memory in 65nm CMOS. In: Proc. IEEE ESSCIRC, pp. 321–324 (September)
20. Akgun, O., et al.: High-level energy estimation in the sub-V_T domain: Simulation and measurement of a cardiac event detector. IEEE TBCAS 6(1), 15–27 (2012)
21. Meinerzhagen, P., et al.: Synthesis strategies for sub-V_t systems. In: ECCTD, pp. 552–555 (2011)
22. Vittoz, E.: Low-Power Electronics Design. CRC Press (2004)
23. Soeleman, H., Roy, K., Paul, B.: Robust subthreshold logic for ultra-low power operation. IEEE T-VLSI Systems 9(1), 90–99 (2001)
24. Harvard-MIT Division of Health Sciences and Technology Biomedical Engineering Center: MIT-BIH arrhythmia database directory, `http://www.physionet.org/physiobank/database/mitdb`