A Configurable 12–237 kS/s 12.8 mW Sparse-Approximation Engine for Mobile Data Aggregation of Compressively Sampled Physiological Signals

Fengbo Ren, *Member, IEEE*, and Dejan Marković, *Member, IEEE*

***Abstract*—Compressive sensing (CS) is a promising technology for realizing low-power and cost-effective wireless sensor nodes (WSNs) in pervasive health systems for 24/7 health monitoring. Due to the high computational complexity (CC) of the reconstruction algorithms, software solutions cannot fulfill the energy efficiency needs for real-time processing. In this paper, we present a 12—237 kS/s 12.8 mW sparse-approximation (SA) engine chip that enables the energy-efficient data aggregation of compressively sampled physiological signals on mobile platforms. The SA engine chip integrated in 40 nm CMOS can support the simultaneous reconstruction of over 200 channels of physiological signals while consuming $<1\%$ of a smartphone's power budget. Such energy-efficient reconstruction enables two-to-three times energy saving at the sensor nodes in a CS-based health monitoring system as compared to traditional Nyquist-based systems, while providing timely feedback and bringing signal intelligence closer to the user.**

***Index Terms*—Application-specific integrated circuits (ASICs), biomedical signal processing, compressed sensing, digital integrated circuits, energy efficiency, low-power design, minimization methods, parallel architecture, real-time systems, reconfigurable architecture, signal reconstruction.**

I. Introduction

DIGITAL electronic industry today relies on Nyquist sampling theorem, which requires to double the size (sampling rate) of the signal representation on the Fourier basis to avoid information loss. However, most natural signals have much sparser representation on some other, non-Fourier, orthogonal basis. This implies a large amount of redundancy in Nyquist-sampled data, making compression a necessity prior to storage or transmission [1], [2]. Recent advances in compressive sensing (CS) theory suggest an alternative data acquisition framework that can directly access the signal information in its sparse domain [3], [4]. Compared to the conventional Nyquist framework, the CS framework has several intrinsic advantages. First, random encoding is a universal compression method that can effectively apply to all compressible signals regardless of what their sparse domain is. This is a desirable merit for the data fusion across multiple signal sources. Second, sampling and compression can be performed at the same stage in CS, allowing for a sampling rate that is significantly lower than the Nyquist rate. Therefore, CS has a potential to greatly impact the data acquisition devices that are sensitive to cost, energy consumption, and portability, such as wireless sensor nodes (WSNs) in mobile and wearable applications [5].

Especially, CS is a promising solution for realizing the on-body WSNs in pervasive health systems toward 24/7 health monitoring [6]. Electrocardiogram (ECG), electromyography (EMG), and electroencephalogram (EEG) signals (collectively referred to as ExG) contain critical information about human body status and are therefore the main targets in health monitoring applications. As shown in Fig. 1, a CS-based wireless health monitoring system includes the on-body WSNs that utilize a unified random encoding scheme to compress different physiological signals to reduce the data size for transmission (thereby saving transmit energy), and a mobile data aggregator that performs real-time signal reconstruction to promote on-site analysis and processing for real-time applications. Such a system has numerous benefits. First, it brings the signal intelligence closer to the user for timely prediction and decision-making. This is particularly important for real-time tasks such as arrhythmia and seizure detection, EMG-driven machine actuation, and brain–computer interface. Second, by reconstructing the sparse coefficients of the original signal only, the data size for on-site storage or transmission to the cloud can be further reduced. For practical use, the data aggregator is desired to have a sufficient throughput for reconstructing > 50 channels of physiological signals (sampled at $\leq 1\,\text{kHz}$) in real time [7]. Additionally, to minimize the overhead of adding such a function to a mobile device, the power consumption of the data aggregator is desired to be bounded within 1% of a mobile device's 2 W power budget. This implies a sparse-approximation (SA) engine that can support $> 50\,\text{kS/s}$ throughput in $<20\,\text{mW}$ of power (see Fig. 2). It is also desirable to have flexibility for varying sparsity parameters, orthogonal basis, and the number of channels. Such a set of specifications imposes significant challenges to the hardware implementation.

Manuscript received May 09, 2015; revised August 23, 2015; accepted September 11, 2015. Date of publication October 15, 2015; date of current version December 30, 2015. This paper was approved by Guest Editor Edith Beigne. This work was supported in part by Broadcom Fellowship Program and in part by TSMC University Program.

F. Ren was with the University of California, Los Angeles, CA 90095 USA. He is now with Arizona State University, Tempe, AZ 85281 USA (e-mail: renfengbo@asu.edu).

D. Marković is with the University of California, Los Angeles, CA 90095 USA (e-mail: dejan@ee.ucla.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JSSC.2015.2480862

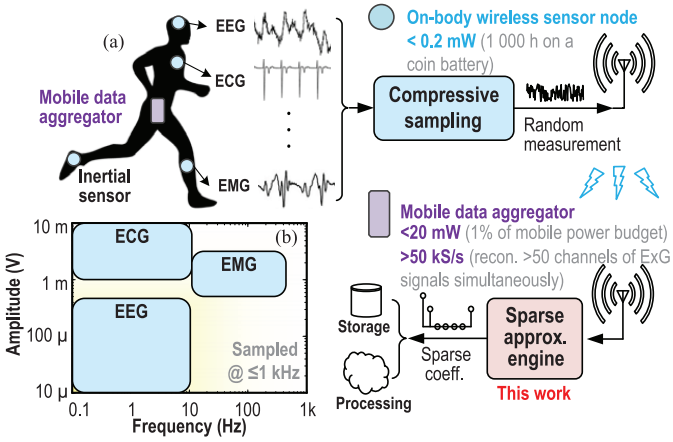

Fig. 1. (a) CS-based wireless health monitoring system with desired system requirements: on-body WSNs that utilize a unified random encoding scheme to compress data for low energy and a mobile data aggregator that performs real-time signal reconstruction for timely prediction and proactive prevention. To further reduce the data size for storage or processing, only the sparse coefficients of the signal are reconstructed. Reconstruction takes $<1\%$ of a mobile device's power budget, allowing 2–3× energy saving of the sensors [5]. (b) Amplitude and frequency characteristics of ExG signals.

The first challenge is the complexity of SA algorithms. SA is an optimization problem that involves complex operations in an iterative process with intensive memory access. Compared to the orthogonal transformations used in the Nyquist framework, SA algorithms have greater computational complexity (CC) and higher data dependency (DD). The second challenge stems from the intricacies of physiological signals. ExG signals can span three orders of magnitude in both amplitude (10 μV to 10 mV) and frequency (0.1–500 Hz) (see Fig. 1). In addition, due to the difference in physiological activity of the signal sources, these signals could have sparse representations on completely different orthogonal bases. Furthermore, their sparsity is time-varying depending on the subject's activity [8]. For the best reconstruction results, the hardware design must be able to handle a high dynamic range and flexible problem settings, such as reconstruction basis (Ψ), error tolerance (ϵ), signal and measurement dimensions (n and m), and signal sparsity level (k).

So far, there has been very limited work and demonstration of dedicated SA solver chips [9]–[11]. The application-specific integrated circuit (ASIC) implementations of three greedy algorithms are first presented in [9] for the long-term evolution (LTE) channel estimation in wireless communication applications. These implementations in 180 nm CMOS feature a target throughput of 2 kS/s with the power consumptions of 88–209 mW. A 65 nm generic solver chip implementing the approximate message passing (AMP) algorithm is demonstrated in [10] for an audio restoration application. This chip achieves a target throughput of 397 kS/s at the power consumption of 177.5 mW for processing audio signals that have a relatively lower sparsity. Prior designs mainly focused on achieving the target throughputs, with much less emphasis on power/energy and area efficiency. Besides, prior designs were optimized for a limited dynamic range and a fixed problem setting, making them unsuitable for biosensing applications.

In this paper, we present a *configurable* and *energy-efficient* SA engine chip in 40 nm CMOS that addresses above challenges and makes the CS technology accessible to mobile users. The chip testing results illustrate a reconstruction throughput of 66–237 kS/s and a power consumption of 12.8 mW when operating at $V_{\mathrm{DD}} = 0.7$ V. Such level of performance can support the simultaneous reconstruction of over 200 channels of compressively sampled ExG signals in real time while consuming $<1\%$ of a smartphone's power budget. The high energy-efficiency of our chip results from an algorithm-architecture codesign approach that facilitates the tight interactions between 1) algorithm reformulations that reduce the algorithm complexity by an order of magnitude; 2) a configurable system architecture that leads to nearly 100% utilization of computing resources; and 3) an efficient memory control scheme that cuts down the memory usage by half. The system architecture of the SA engine chip is optimized toward mapping the orthogonal matching pursuit (OMP) algorithm and its variants [12], [13]. Because human body is expected to have a low activity on average where ExG signals feature a high sparsity, especially when dynamic thresholding schemes are used [8], this is where OMP has better complexity–accuracy tradeoff than other SA algorithms [14]. The SA engine chip implements domain transformation by explicit matrix multiplication thereby supporting signal reconstruction on arbitrary basis. Additionally, the SA engine adopts the single-precision floating-point data format to achieve a large dynamic range and can be configured at run time to handle flexible problem settings and accurately recover a wide range of physiological signals.

II. Algorithm Reformulation Toward Energy Efficiency

Energy efficiency is the metric indicating how much computing can be performed with a finite energy source. For dedicated algorithms running on hardware, energy efficiency is usually defined as the energy consumption per algorithmic execution, which can be measured by the ratio of power (J/s) and processing throughput (S/s). From the hardware perspective, both the CC and the DD characteristics of an algorithm impact the energy efficiency. A high CC indicates a large amount of computations per algorithmic execution, implying more switching energy from the logic gates. On the other hand, a high *loop-carried* DD indicates low concurrency of computations, generally implying increased memory usage and longer execution time that leads to higher leakage energy.

OMP is a fast and heuristic algorithm that can recover a k-sparse signal in exact k iterations given the constraints in the context of CS [3], [4], [12]. The pseudocode of the original OMP algorithm is shown in Table I (see Appendix for notations). Note that Cholesky factorization is favored over QR factorization as the numerical method for solving the least-squares (LS) problem since QR factorization requires three times more memory for storing the factorization matrices, which is undesired for memory-leakage-limited design. In each iteration, three tasks are performed: 1) atom searching (AS) for updating the active set; 2) LS solving for computing the

TABLE I
PSEUDOCODE OF THE OMP ALGORITHM

Task	Step	Operation
	1	$r_0 = y, x_0 = \vec{0}, d = \vec{0}, \Lambda_0 = \emptyset, t = 1.$
AS	2*	$c = \boldsymbol{A}^T r_{t-1}$, $\varphi = \arg\max_i \lvert c(i)\rvert$, $\Lambda_t = \Lambda_{t-1} \cup \varphi.$
LS	3	$x(\Lambda_t) = \arg\min_b \lVert y - \boldsymbol{A}_{\Lambda_t} b \rVert_2^2.$
EU	4	$r_t = r_{t-1} - \boldsymbol{A}_{\Lambda_t} x(\Lambda_t),\ t = t + 1.$
	5	If $\lVert r_t \rVert_2 \leq \varepsilon$, break; otherwise, go to Step 2).

*Assuming that all atoms in $\boldsymbol{A}$ are normalized.
#Refer to Appendix and [12] for notations.

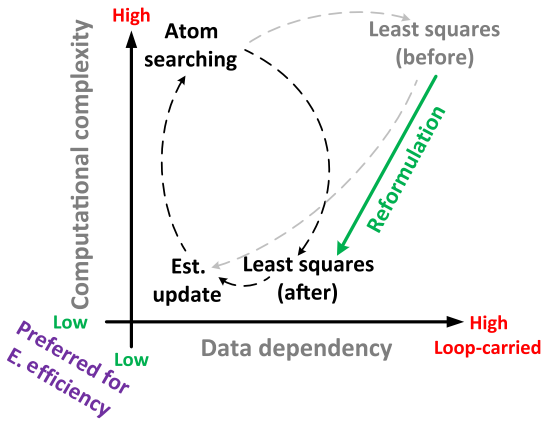

Fig. 2. Complexity characteristic of the OMP algorithms. The impact of the reformulation techniques is making OMP more energy-efficient for hardware implementations by simplifying the LS task.

estimation; and 3) estimation update (EU). The complexity characteristic of the OMP algorithms is illustrated in Fig. 2. Note that the LS task is a minimization problem that involves both high CC and loop-carried DD [14]. For energy-efficient mapping, three algorithm reformulation techniques are previously introduced to break down the LS task into four basic linear algebra (BLA) operations (at each iteration) [14]–[16]. The pseudocode of the reformulated OMP algorithm is shown in Table II (see Appendix for notations). Such simplification not only reduces the total CC of the LS task from $O(mk^3)$ to $O(mk^2)$ but also cuts the number of data-dependent loops involved from k^3 down to k^2 (see Appendix B for definitions of n, m, k in the context of CS), making the reformulated OMP algorithm much more suitable for an energy-efficient hardware implementation [14].

In the reformulated OMP, the AS task that features high CC but low DD has the biggest impact on throughput. Parallelization is applied in our architecture design to relax the transistor switching speed for gaining energy efficiency. On the other hand, the LS task plays a pivotal role on hardware utilization. Note that any hardware resource designed exclusively for the LS task will have a very low utilization rate due to the low CC. Consequently, resource sharing is applied in the architecture design to improve hardware utilization and gain energy efficiency from reduced area and leakage costs.

TABLE II
PSEUDOCODE OF THE REFORMULATED OMP ALGORITHM

Task	Op.	Algorithm
	1	**Initialize**: $r_0 = y, x_0 = \vec{0}, d = \vec{0}, \Lambda_0 = \emptyset, t = 1$
	2	**While** $\lVert r_{t-1} \rVert_2^2 < \varepsilon$ or $t \leq t_{\max}$, **do**
AS	3*	$c = \boldsymbol{A}^T r_{t-1}$, $\varphi = \arg\max_i \lvert c(i)\rvert$, $\Lambda_t = \Lambda_{t-1} \cup \varphi$
	4	$h = \boldsymbol{A}_{\Lambda_t}{}^T a_\varphi$
	5	$\boldsymbol{L}_{t-1} w = h(1{:}t-1)$, $l_{21} = w./\mathrm{diag}\ (\boldsymbol{D}_{t-1})$
	6a	$d_{22} = h(t) - l_{21}{}^T w$
LS	6b†	$\boldsymbol{L}_t = \begin{bmatrix} \boldsymbol{L}_{t-1} & \vec{0} \\ l_{21}{}^T & 1 \end{bmatrix}$, $\boldsymbol{D}_t = \begin{bmatrix} \boldsymbol{D}_{t-1} & \vec{0} \\ \vec{0}^T & d_{22} \end{bmatrix}$ ($\boldsymbol{L}_1$=1, $\boldsymbol{D}_1 = a_\varphi{}^T a_\varphi$)
	7	$\boldsymbol{L}_t{}^T d = \begin{bmatrix} \vec{0} \\ c(\varphi)/\, d_{22} \end{bmatrix}$
EU	8	$x_t = x_{t-1} + d, r_t = r_{t-1} - \boldsymbol{A}_{\Lambda_t} d,\ t = t + 1.$
		End while, return x_t

*Assuming that all atoms in $\boldsymbol{A}$ are normalized.
†Memory operation only.
#Refer to Appendix and [12] for notations.

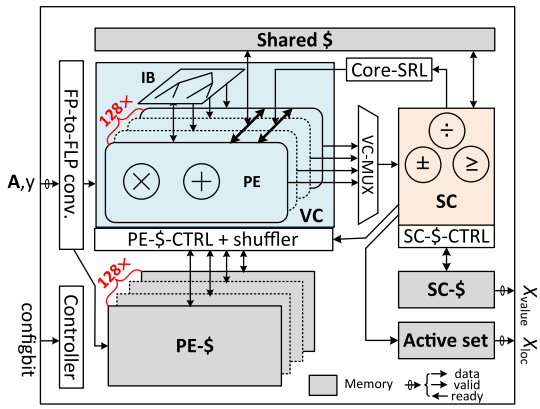

Fig. 3. System architecture of the SA engine chip.

III. ARCHITECTURE DESIGN

A. System Architecture

The system architecture of the SA engine chip is shown in Fig. 3. The computing resources include the vector and scalar processing cores (VC and SC). In order to support a real-time throughput with high energy efficiency, 128 processing elements (PEs) are coordinated in parallel through the interconnect block (IB) in the VC. These PEs can process independent data in a single-instruction-multiple-data (SIMD) fashion or interconnected by the IB to perform pipelined operations. The large parallelism of PEs allows the SA engine to achieve the target throughput at a scaled supply voltage and reduced operating frequency. Therefore, additional energy efficiency can be gained from the relaxed transistor performance. Depending on the top-level data-path configuration, SC can either postprocess a selective result from the VC through the VC-multiplexer (VC-MUX) or process independent data from memories in parallel.

For efficient local memory access, a dedicated cache is assigned to each PE in the VC and the SC, respectively. To facilitate the data communication between VC and SC in long delay lines, such as carrying over intermediate results between different tasks or different iterations of the algorithm, a shared cache can be accessed by all the PEs in the VC and the SC. In addition, a core-level shift-register logic (SRL) unit (core-SRL) is used to connect all the PEs with the SC. This customized feedback path minimizes the loop latency between VC and SC by avoiding any memory access, thereby accelerating the iterative BLA operations such as forward and backward substitution (FS and BS). A dedicated memory unit stores the index of the active set. The controller of this memory unit is also responsible for accessing the data in the sampling matrix from external memories.

To allow the processing of different signal representations, a parallelized fixed-to-floating-point conversion interface is available at the external input of the VC. Note that there are several data-paths bridging the VC and SC in the system architecture. The complex data flow of the reformulated OMP is enforced by the customized local memory controllers ("PE-$-CTRL" and "SC-$-CTRL" in Fig. 3), which are coordinated by a global controller. All these controllers are dedicated finite-state machines (FSMs) with programmable state transition contingent upon the values of the configuration bits ("config-bit" in Fig. 3). The configuration bits set the problem size (m, n) and error tolerance (ε) for each run of the algorithm. Therefore, the SA engine can be configured with a different problem setting for each reconstruction. Note that the memory-based data-flow-control schemes are efficient in handling data reordering operations, such as matrix transpose, since most of the data movements can be realized by pointer manipulations. The dynamic configuration of the computation cores is also controlled by dedicated FSMs in a similar fashion. The SA engine uses first-in-first-out (FIFO) interfaces to handle the flow control at the data I/Os. When a reconstruction is done, the chip loads new random samples (y) and unloads reconstructed sparse coefficients ("x_{value}" in Fig. 3) with their index ("x_{loc}" in Fig. 3) simultaneously. Once the loading and unloading are complete, the next reconstruction is kicked off.

B. Computation Cores

The block diagram of the PE in the VC is shown in Fig. 4. The PE integrates two basic arithmetic units in a pipeline: a multiplier and an adder. Flexible data-path connections are realized by inserting multiplexers at each input of the arithmetic units. Therefore, the PE can be dynamically configured to execute different operations or take different operands through the control bits of the multiplexers. Note that the multiplier can be bypassed by setting one of its inputs to 1, and the adder can be bypassed by resetting the SRL output to 0. Therefore, the PE can perform a selective set of operations including multiplication, recursive multiplication, power operation, addition, accumulation, and MAC.

On the VC level, the 128 PEs can perform vector operations in an SIMD fashion including vector addition, element-wise multiplication, element-wise MAC, and vector–scalar product.

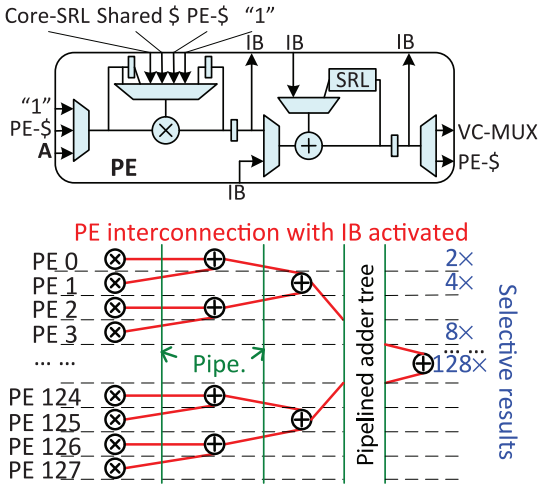

Fig. 4. Block diagram of PE and PE interconnections provided by IB.

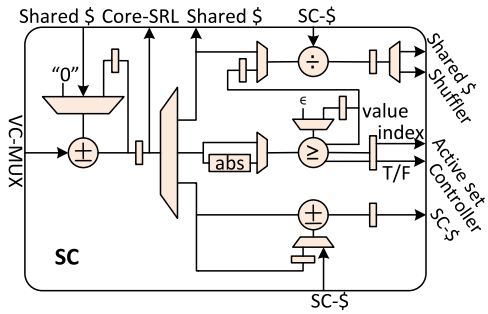

Fig. 5. Block diagram of SC.

To enable folded processing of long vectors, an SRL unit is inserted into the feedback path of each PE. The folding factor is dictated by the latency of the SRL unit. In addition, the PEs can be coordinated through the IB to compute vector inner products (Fig. 4). The IB connects the adders distributed in different PEs into a pipelined adder tree through the registers and multiplexers inside the PEs, so that the element-wise products can be added up to a scalar. The inner product computation in this mode is highly scalable: for a different vector length, the corresponding result can be selected by the VC-MUX at a different pipeline stage. The folded inner product computation in this mode needs an additional accumulator at the output of VC-MUX, which is carried by the SC.

The block diagram of the SC is shown in Fig. 5. The SC integrates a comparator, a sequential divider, and two adders with configurable data-paths. Similar to the VC, the SC can also be configured to perform a variety of operations through the control bits of the multiplexers. When the SC is cascaded with the VC, complex operations such as correlation sorting, FS, and BS can be performed.

The first stage adder in the SC plays a critical role in two tasks. First, it accumulates the result from the VC to support folded inner product. Second, it adds the RHS of a linear equation to the LHS for performing FS and BS. The sequential divider is used to handle the inverse of a diagonal matrix as in op. 5 of Table II. Note that the division in op. 5 is not part of the data-dependent loops in solving FS. Therefore, the latency

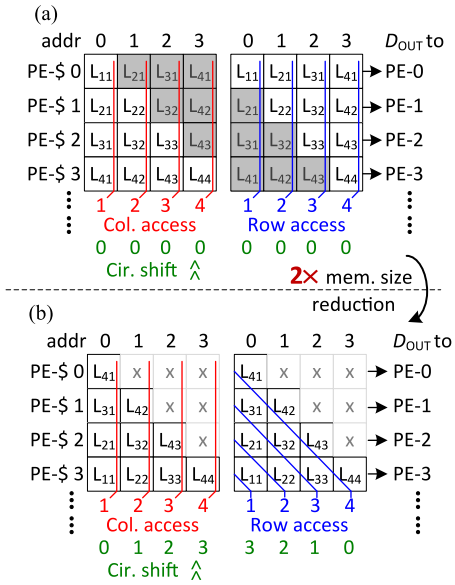

Fig. 6. Data mapping scheme of PE caches in the (a) mirror and (b) shuffle mode for handling Cholesky factorization.

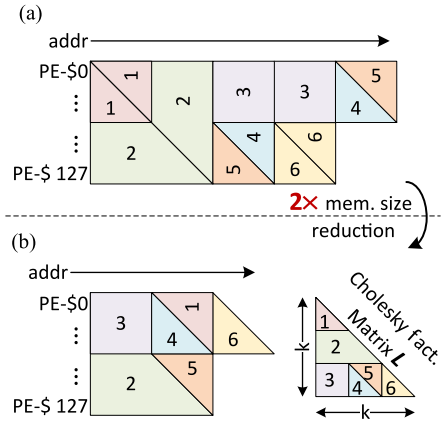

Fig. 7. Data folding scheme of PE caches in the (a) mirror and (b) shuffle mode for handling Cholesky factorization ($k = 192$).

of the divider has no impact on the throughput of FS execution. Five pipelined stages are inserted (through retiming) to remove the divider from critical path. The comparator is used to perform sorting tasks, such as sorting the correlation coefficients in the AS task. The second-stage adder can be used to update the results of folded inner product as in op. 6a of Table II or to update independent data as in op. 8 of Table II.

C. Memory Control Scheme

In the LS task of the reformulated OMP, the parallel column and row access of the triangular Cholesky factorization matrix $\boldsymbol{L} \in \mathbb{R}^{k \times k}$ are required for performing FS and BS, respectively [17]. As accessing a row of $\boldsymbol{L}$ is equivalent to accessing a column of $\boldsymbol{L}^T$, a straightforward memory mapping scheme of PE caches is to store both $\boldsymbol{L}$ and $\boldsymbol{L}^T$ in a square matrix as illustrated in Fig. 6(a). We refer to this data mapping scheme as the mirror mode. In the mirror mode, columns of $\boldsymbol{L}$ and $\boldsymbol{L}^T$ can be accessed at the same address of each PE cache in an ascending and descending order, respectively. For instance (see Fig. 6), the column vector $\boldsymbol{l}_1$, $\boldsymbol{l}_2$, and $\boldsymbol{l}_3$ can be accessed in parallel by reading the data at address 0, 1, and 2 of each PE cache, respectively. The row vector $\boldsymbol{l}_1^T$, $\boldsymbol{l}_2^T$, and $\boldsymbol{l}_3^T$ can be accessed in parallel by reading the data at address 4, 3, and 2 of each PE cache, respectively. An advantage of the mirror mode is that a large square matrix can be easily folded into smaller sub-blocks so that a large-size Cholesky factorization can be computed in a folded fashion by utilizing the PE-SRL and the core-SRL units. An example data folding scheme in the mirror mode is illustrated in Fig. 7(a), where a folding factor of 1.5 is presented with 128 parallel PEs and $k = 192$.

The down side of the mirror mode is that it doubles memory space for storing $\boldsymbol{L}$. Since the SA engine is a memory-leakage-limited design, where memory leakage has significant impact on the system's energy efficiency, the mirror mode is highly undesired. To avoid such an overhead, we propose a shuffle-mode scheme that is more efficient in utilizing memory space in our design. In the shuffle mode, the row elements of $\boldsymbol{L}$ are stored across adjacent PE caches in a shuffle order as illustrated in Fig. 6(b). The data are shuffled such that the row access pattern remains the same, but the column of $\boldsymbol{L}$ can be accessed from the same memory space by using an incremental address pattern across PE caches. Note that a circular position shift must be performed at the memory output in order to recover the correct data order. For instance (see Fig. 6), the column vectors $\boldsymbol{l}_1$, $\boldsymbol{l}_2$, and $\boldsymbol{l}_3$ can be accessed in parallel by reading the data at address 0, 1, and 2 of each PE cache with a position up-shift by 0, 1, and 2, respectively. Differently, the row vector $\boldsymbol{l}_1^T$, $\boldsymbol{l}_2^T$, and $\boldsymbol{l}_3^T$ can be accessed in parallel by reading the data at the address set of [0, 1, 2, 3], [X, 0, 1, 2], and [X, X, 0, 1] with a position up-shift by 0, 1, and 2, respectively.

By adopting the shuffle-mode scheme, a $2\times$ memory size reduction is achieved as compared to the mirror-mode case. According to the postlayout simulation results, this leads to another 40% saving in total power consumption of the chip due to the reduced memory leakage. The corresponding data folding scheme of the shuffle mode is illustrated in Fig. 7(b).

D. Dynamic Configuration of System Architecture

Taking advantages of the reformulated OMP algorithm with a simplified LS task, we manage to reuse computing resources to perform all the three tasks through dynamic configuration. Due to the intrinsic DD between the six BLA operations in Table II, the proposed resource sharing scheme maximizes the hardware utilization rate and area efficiency without introducing throughput overhead.

Fig. 8 illustrates the dynamic configuration of the system architecture in three tasks. In the AS task, the VC is cascaded with the SC in pipeline. The VC accesses $\boldsymbol{a_i}$ and r_{t-1} in parallel from the external memory and the PE caches, respectively. The PEs are configured to compute their inner product as $c(i) = \boldsymbol{a_i} r_{t-1}$. The SC accumulates the result when folding is enabled and compares the absolute values of $c(i)$ with that of $c(i-1)$. The smaller value is dropped, while the larger value and the associated column index is buffered for the next comparison. After all the correlation coefficients are compared, the

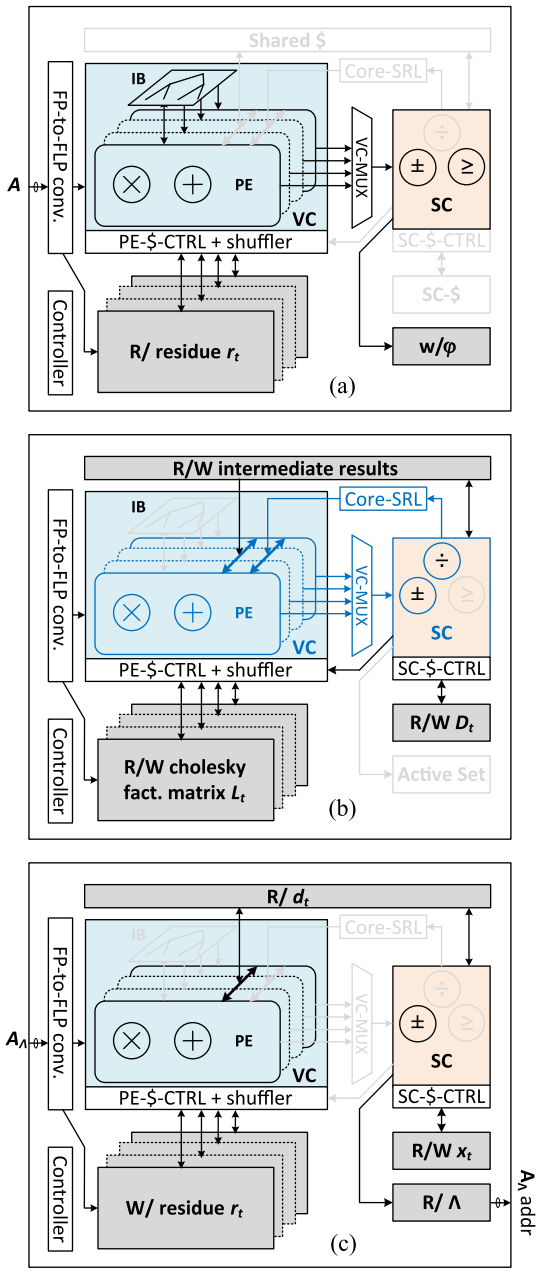

Fig. 8. Dynamic configuration of the system architecture in the (a) AS, (b) LS, and (c) EU tasks.

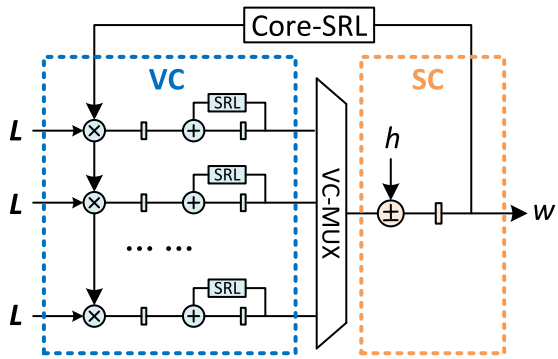

Fig. 9. Data-path configuration of the computing resources in the VC and the SC for computing FS and BS.

column index of the maximum component is written into the active set memory.

In the LS task, a series of matrix–vector multiplications, FS, divisions, and BS need to be executed (see ops. 4–7 in Table II). For computing matrix–vector multiplications in ops. 4 and 6a, the same configuration as in the AS task is used. Differently, in order to compute FS and BS using recursive vector operations, the core-SRL is enabled to link the adder in SC with the PEs in the VC into parallel loops. The SRL units in the PEs are also enabled to support the folded computation of large-size FS and BS. Fig. 9 illustrates the data-path configuration of computing resources in the VC and the SC for computing the FS in op. 5 (Table II). Note that performing FS and BS in an iterative fashion has little impact on the system throughput. This is because 1) FS and BS are intrinsically an iterative process that has loop-carried DD and 2) the LS task is not the throughput bottleneck in the reformulated OMP as shown in Fig. 2. In addition, computing FS and BS using vector-based operations allows for the reutilization of the VC and improves the hardware utilization rate. When the FS in op. 5 (Table II) executes iteratively using the configuration shown in Fig. 9, the subsequent divisions can be then scheduled to the SC and executed by the pipelined sequential divider cascaded.

In the EU task, the two computation cores are configured to update the estimation results separately. The VC accesses $\boldsymbol{A}_{\boldsymbol{\Lambda}_t}$ and $d(\Lambda_t)$ from the external memory and the shared cache, respectively. Note that the active atoms $\boldsymbol{A}_{\boldsymbol{\Lambda}_t}$ are accessed by using the contents from the active set memory as the read address. One should also note that the matrix–vector multiplication $c(i) = \boldsymbol{A}r_{t-1}$ in the AS task is executed by computing the independent inner products as $\boldsymbol{a}_i r_{t-1}$. Differently, the matrix–vector multiplication $v = \boldsymbol{A}_{\boldsymbol{\Lambda}_t} d(\Lambda_t)$ in the EU task is computed in a column-wise fashion by configuring the PEs into an element-wise MAC mode. Each clock cycle, one column of $\boldsymbol{A}_{\boldsymbol{\Lambda}_t}$ and a single element of $d(\Lambda_t)$ are accessed and multiplied, and the results are accumulated element-wise in the SRL units in PEs. After $t \times F$ cycles, where F is the folding factor, the result v will be available in the SRLs. Then, the residual r_t is updated by the PEs in parallel as $r_t = r_{t-1} + v$. Meanwhile, the SC updates x_t element-wise as $x_t(i) = x_{t-1}(i) + d(i)$ whenever $d(i)$ is read out from the shared cache. The overall dynamic configuration scheme of the SA architecture is summarized in Table III.

IV. Chip Implementation

The die photo and chip summary are shown in Fig. 10. The SA engine chip is implemented in a 40 nm 1P8M CMOS process using a standard-cell-based design flow. The RTL codes are synthesized in synopsys design compiler (DC). To achieve the target throughput, a clock period of 60 ns (16.7 MHz) evaluated at the worst-case process, voltage, and temperature (PVT) corner is targeted throughout the chip implementation. Taking into account the overhead to be introduced by the subsequent physical design, a 22% timing slack is used during the synthesis. Specifically, the SA engine is synthesized with a target clock frequency of $16.7/(1-0.22) = 21.4$ MHz. To reduce leakage

TABLE III
SUMMARY OF DYNAMIC CONFIGURATION SCHEME OF THE SYSTEM ARCHITECTURE

Task	Op.	PEs	PE-SRL	IB	VC-MUX	Core-SRL	SC	PE-$*	SC-$*	Shared $*	Active set*
AS	3	Inner Prod.	OFF	ON	Static	OFF	ACC, CMP	R/ r_t	OFF	OFF	W/ Λ
LS	4	Inner Prod.	OFF	ON	Static	OFF	ACC	OFF	OFF	W/ h	R/ Λ
	5	MAC	ON	OFF	Dynamic	ON	ACC, DIV	R/W $\boldsymbol{L}$	R/ $\boldsymbol{D}$	R/ h, W/ w	OFF
	6	Mult./Inner Prod.	OFF	ON	Static	OFF	ACC	R/ $\boldsymbol{L}$	W/ $\boldsymbol{D}$	R/ w	OFF
	7	MAC	ON	OFF	Dynamic	ON	ACC, DIV	R/ $\boldsymbol{L}^T$	OFF	W/ d	OFF
EU	8	MAC	ON	OFF	OFF	OFF	ACC, ADD	R/W r	R/W x	R/ d	R/ Λ

*R/W = read/write.

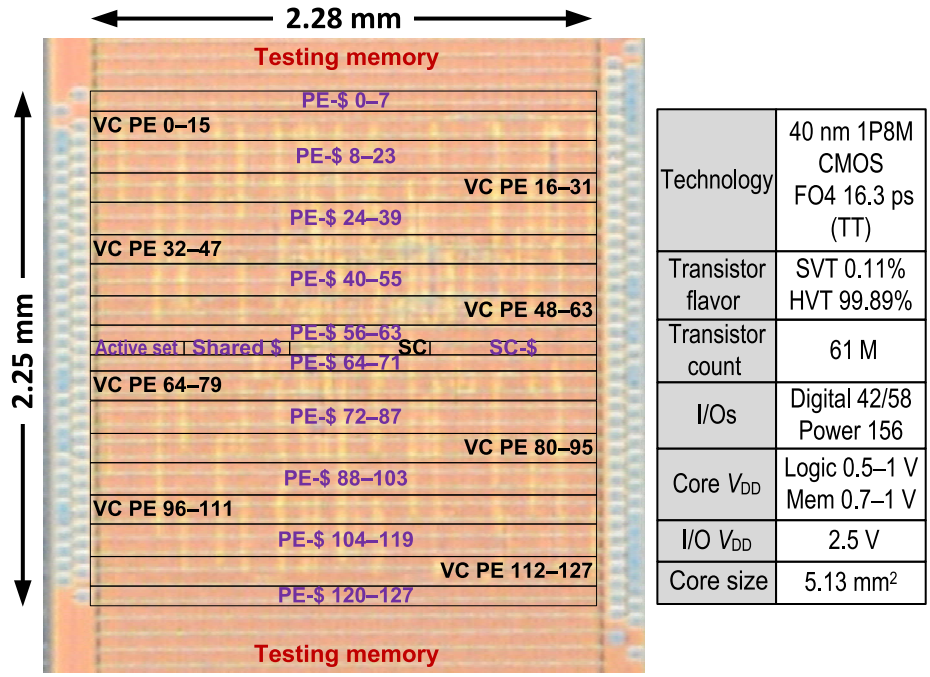

Fig. 10. Die photo and summary of the SA engine chip.

power, the SA engine is first synthesized using high-threshold (HVT) standard cells only.[1] Then, standard-threshold (SVT) standard cells are selectively inserted to the critical paths for timing improvement. This is carried out by switching ON the leakage optimization tool in DC.

The PE cache, SC cache, and shared cache in the SA engine have a memory size of 1.2 KB, 1.5 KB, and 768 B, respectively. Note that there are a total of 128 instances of the PE cache in the design. To reduce area cost, the PE caches are realized using dual-port SRAM hard macros. Differently, the SC cache and the shared cache are realized using synthesized RAMs mainly because they can be flattened during the physical design to facilitate floorplanning. For voltage scaling purposes, the SA engine is split into two power domains. The PE caches realized by SRAM macros are under the memory (high voltage) domain, while the rest of the design is under the logic (low voltage) domain. Lever shifters are placed along the boundary of SRAM macros to handle signaling across the two voltage domains.

[1]The HVT option is not available in our memory complier. The transistors used in the memory macros are still SVT. Therefore, the memory leakage dominates the system leakage power.

The physical design of the SA engine is performed in Cadence Encounter. To reduce the run time, a bottom-up hierarchical design method is adopted. Specifically, the PE and the PE cache are first placed and routed separately at the block level. During the chip-level floorplanning, these two blocks are treated as hard macros. For the best implementation results, the rest of the design is flattened during the top-level placement and routing. Note that the PE macro is routed using M1–M4 only so that the 128 instances will not block the routing channels on M5–M8 during the top-level placement and routing.

To facilitate the top-level routing, the PE and PE cache instances are grouped into 128 pairs, which are then placed into 16 rows. In each row, eight pairs of the PE group are placed evenly with a 50 μm space in between. To enhance power delivery, a global power grid is routed across both of the voltage domains over the entire chip. To minimize IR drops, the global power stripes are routed using the redistribution (RDL) and M8 layers that have smaller resistance and support higher current density.

Overall, the SA engine chip occupies a core area of 5.13 mm^2 with an aspect ratio of 0.99 and integrates 61 M transistors. For the leakage reduction purpose, HVT devices are used in 99.89%

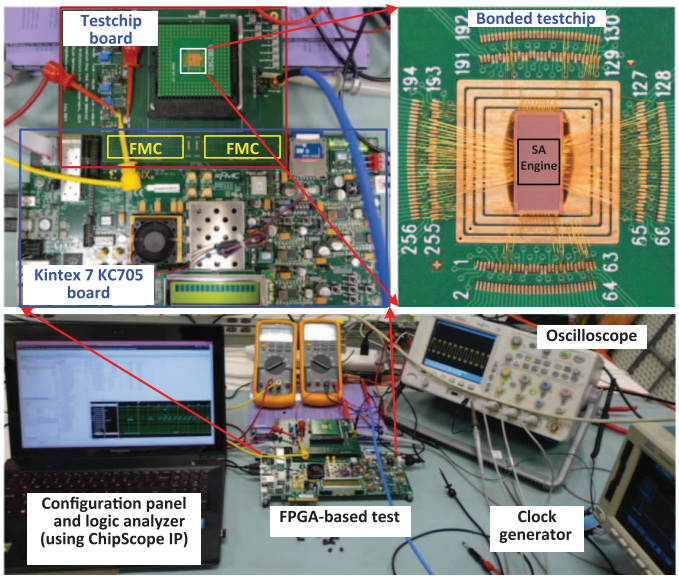

Fig. 11. Testing environment of the SA engine chip.

of the logic cells. The SA engine chip has 42 digital inputs, 58 digital outputs, and 156 power pads supplying three different power domains. The I/O domain has a constant supply voltage of 2.5 V. The logic and memory domain both have a nominal supply voltage of 0.9 V, while each operates up to 1 V and down to 0.5 and 0.7 V, respectively.

V. Chip Testing

A. Testing Environment

The chip testing environment is illustrated in Fig. 11. A Kintex-7 KC705 FPGA board is used as the testbed for mapping hardware test benches. A customized printed circuit board (PCB) is designed to host the SA engine chip for testing. The SA engine chip is wire-bonded to a 256-pin pin grid array (PGA) package and then mounted to the host PCB through a zero insertion force (ZIF) socket. The host PCB is connected to the KC705 board through two high-speed FPGA Mezzanine Card (FMC) connectors. A clock generator is used as the external clock source for both the FPGA and the SA engine chip. The clock is injected to the host PCB through an SMA connector and then passed to the FPGA board through the dedicated clock pins in the FMC connector. In order to control and monitor the chip testing process on a computer, Xilinx ChipScope IPs are utilized in the test bench design. Specifically, ChipScope virtual I/O (VIO) is used as the soft registers both to store the static control bits of the SA engine chip and the test bench and to monitor the static outputs indicating the chip status. In addition, ChipScope integrated logic analyzer (ILA) is used as the probes to capture the dynamics of all the digital I/Os of the SA engine chip.

B. Testing Results

Several 1 min recordings of real ExG signals downloaded from the PhysioBank database are used in the signal reconstruction test [18]. Specifically, the original ExG signals are encoded

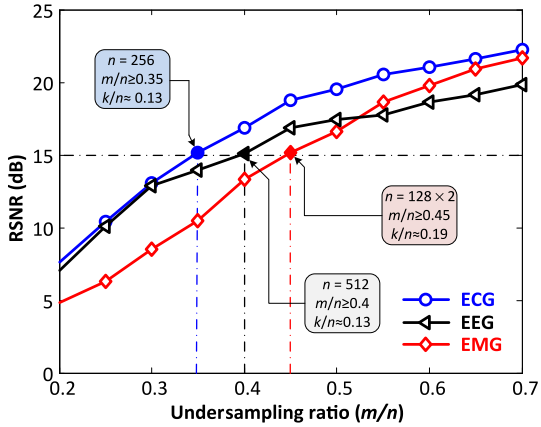

Fig. 12. Averaged RSNR performance of the SA engine chip for ExG signal reconstruction. The ECG, EEG, and EMG signals are reconstructed on the Haar DWT, DCT, and DWT–DCT joint basis, respectively.

by random Bernoulli matrices with a 5% overlapping window applied at different signal dimensions and under-sampling ratios (see Appendix B). Then, the random samples are fed into the SA engine chip to reconstruct the signal coefficients on a specific sparsifying basis. The original signal can then be recovered by back projecting the reconstructed sparse coefficients into time domain. In order to observe the raw signal sparsity, no thresholding scheme is applied in our test. To measure the reconstruction accuracy, we use the metric of reconstruction SNR (RSNR) defined as

$$\text{RSNR} = 20\log_{10}\left(\frac{\|x\|_2}{\|x-\hat{x}\|_2}\right) \quad (1)$$

where x is the original signal and $\hat{x}$ is the recovered estimation of x. The averaged RSNR performance measured on the SA engine chip is shown in Fig. 12. The best orthogonal basis for reconstructing the chosen ECG, EEG, and EMG signals is found to be Haar discrete wavelet transform (DWT), discrete cosine transform (DCT), and DWT–DCT joint basis, respectively. It is also found that the RSNR performance is sensitive to the error tolerance (ε) setting of the chip. Dynamically configuring ε to 3%–5% of the energy of random samples results in the best RSNR performance. In general, higher under-sampling ratio improves the RSNR performance at the cost of higher data rate for radio transmission. In addition, at the same under-sampling ratio, using a higher signal dimension in compressive sampling improves RSNR slightly at the cost of reduced throughput and increased energy consumption. Therefore, given a target RSNR, there exists an optimal chip setting for achieving the maximum throughput. For reconstructing the ECG, EMG, and EEG with a target RSNR of > 15 dB, the preferred chip setting is found to be $\{n = 256, m \geq 90\}$, $\{n = 128, m \geq 58\}$, and $\{n = 512, m \geq 205\}$, respectively. These settings indicate that an under-sampling ratio (m/n) of 0.35, 0.45, and 0.4 can be achieved (for > 15 dB RSNR) on the ECG, EMG, and EEG sensor nodes through compressive sampling, which corresponding to an approximate sensor energy saving of $2.8\times$, $2.5\times$, and $2.2\times$, respectively, due to the reduced data size (m/n) for wireless transmission [5]. Example ExG signals reconstructed at the preferred settings are

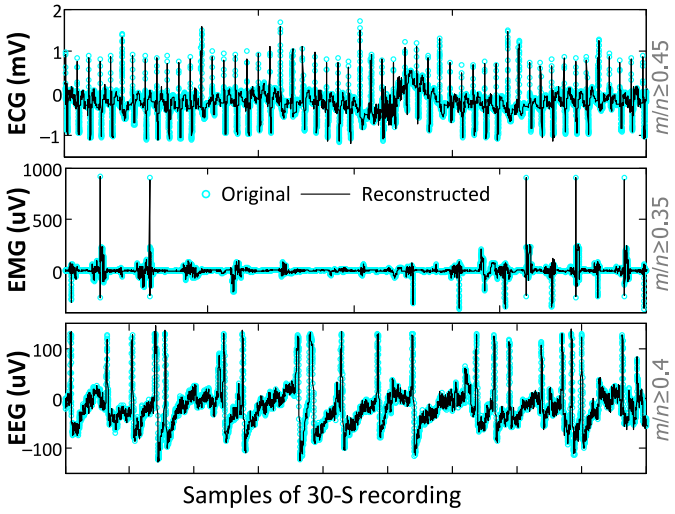

Fig. 13. Examples of ExG signals reconstructed on the SA engine chip with a > 15 dB RSNR performance. The ECG, EEG, and EMG signals are reconstructed on the Haar DWT, DCT, and DWT–DCT joint basis, respectively.

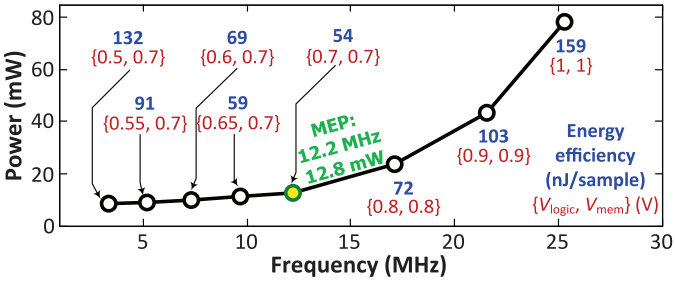

Fig. 14. Measured power and operating frequency at different supply voltages.

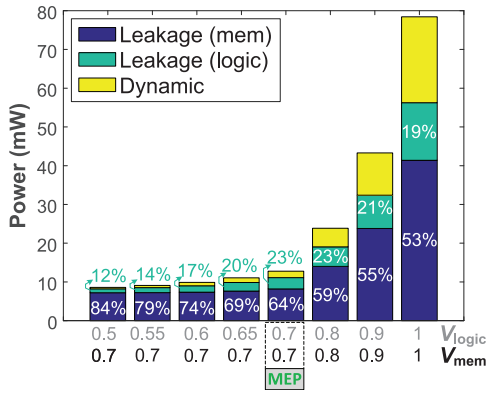

Fig. 15. Power breakdown at different supply voltages. At the MEP ($V_{\rm logic} = V_{\rm mem} = 0.7$ V), memory and logic leakage power contribute to 64% and 23% of the total power consumption, respectively.

TABLE IV
MEASURED THROUGHPUT AND ENERGY EFFICIENCY OF THE SA ENGINE CHIP

@ MEP		Signal dimension (n)							
		128	256	384	512	640	768	896	1024
Throughput* (kS/s)	ECG	387	237†	102	79	64	38	33	29
	EMG	213	123†	54	41	24	20	13	12
	EEG	331	201	87	66†	54	32	27	19
Energy efficiency* (nJ/sample)	ECG	33	54†	125	163	200	337	390	444
	EMG	60	104†	238	312	536	640	954	1087
	EEG	39	63	147	194†	238	399	466	685

*Measured at the MEP for ExG signal reconstruction.
†The highlighted numbers are the best performance at an RSNR of >15 dB.

illustrated in Fig. 13. Note that the chip is flexible enough to accommodate various physiological signals and reconstruction quality requirements through the configurable settings of m, n, and ε and the support for different reconstruction bases.

The measured power and operating frequency of the SA engine chip at different supply voltages are shown in Fig. 14. The memory and logic domain ($V_{\rm mem}$ and $V_{\rm logic}$) of the SA engine chip can operate down to 0.7 and 0.5 V, respectively. The minimum energy point (MEP) for operation is found at $V_{\rm logic} = V_{\rm mem} = 0.7$ V, which is the minimum supply voltage the SRAM macros can operate at. At the MEP, the chip has an operating frequency of 12.2 MHz and a power consumption of 12.8 mW. The breakdown of total power consumption by dynamic power, logic leakage power, and memory leakage power is shown in Fig. 15. Note that the SA engine chip is a memory-leakage-limited design. At the MEP, memory and logic leakage contribute to 64% and 23% of the total power consumption, respectively. As we further scale down $V_{\rm logic}$ while keeping $V_{\rm mem}$ at 0.7 V (for functionality), the memory leakage power becomes increasingly dominant. At the minimum supply voltages, 84% of the total power is consumed by memory leakage. This indicates that lowering $V_{\rm logic}$ below 0.7 V will reduce the operating frequency without making much impact on the total power consumption, thereby degrading the energy efficiency. Compared to the MEP, a two-time higher operating frequency can be achieved at $V_{\rm DD} = 1$ V with a six-time higher power consumption.

The measured throughput and energy efficiency of the SA engine chip when operating at the MEP for ExG signal reconstruction are summarized in Table IV. At the MEP, the chip achieves a throughput of 237, 123, and 66 kS/s and an energy efficiency of 54, 104, and 194 nJ/sample for reconstructing ECG, EMG, and EEG signals at RSNR > 15 dB, respectively. Such level of performance is sufficient to support the simultaneous reconstruction of 237, 61, and 132 channels of ECG, EMG, and EEG signals, respectively. Operating at $V_{\rm DD} = 1$ V, the chip can achieve a two-time higher throughput at the cost of a three-time lower energy efficiency.

The SA engine chip is compared to an Intel Core i7-4700MQ processor and two existing SA solver chips [9], [10] designed for different applications in Fig. 16. For fair comparison, the designs that implement fast algorithms (such as FFT) for a dedicated sampling matrix are not considered for our comparison, since the SA engine chip implements domain transformation explicitly and supports arbitrary sampling matrices. In addition, we apply the same problem settings used in the reference design when making the comparison. While the reference designs targeted a fixed problem setting and a limited dynamic range, our chip handles flexible problem settings at run time and supports a large dynamic range. Overall, the SA engine chip achieves

Design	Intel i7-4700MQ	[9]			[10]	This work
Technology	22 nm	180 nm			65 nm	40 nm
Target app	General	LTE channel estimation			Audio	Biomedical sensing
Algorithm	OMP/AMP	MP	GP	OMP	AMP	OMP, K-OMP
Max. signal dim. (n)	Large	256			512[1]	Up to 1024
Max. meas. dim. (m)	Large	200			512	Up to 512
Max. sparsity level (k)	Large	50	18	10	–	Up to 192
Core area (mm^2)	174.4	0.73	1.21	2.42	0.629	5.13
Data format	Double precision	Fixed point			Fixed point	Single precision
PE parallelism	SSE4	2	8		32	128
Local memory (kB)	7 424	–			5.76	147
Freq. (MHz)	2 394	140	128		333	27.4
Throughput (kS/s)	5–98[3]	2			397	12–237[2]
Power (mW)	47,000	88	209	200	177.5	8.6–78
Energy efficiency* (nJ/sample)	451,322	2 444	5 778	11 222	–	32[4] (high sparsity)
	503,028	–			223	389[5] (low sparsity)

* Technology scaling to 40 nm: delay~1/S, power~1/U^2, where $S = L/40$ nm, $U = V_{DD}/0.9$ V.
[1] The supported problem size is 1024×512, but only half of the sampling matrix is generic.
[2] ExG reconstruction throughput measured at MEP.
[3] ExG reconstruction throughput measured in MATLAB simulation.
[4,5] For fair comparison, the numbers are reported using the same problem settings (m,n,k) as in the reference design.

Fig. 16. Comparison to state-of-the-art.

a two-time higher throughput with up to 14,100 times better energy efficiency for ExG signal reconstruction than the software solver running on the CPU. For high-sparsity signal reconstruction $\left(\frac{k}{n} \leq 4\%\right)$, the SA engine chip is 76–350 times more energy efficient than the reference design [9]. For low-sparsity signal reconstruction $\left(\frac{k}{n} \geq 16\%\right)$, the SA engine chip is less energy efficient than the reference design implementing the AMP algorithm [10] since the number of iterations required by AMP is less dependent on the signal sparsity level.

VI. Conclusion

In this paper, we present a 12–237 KS/s 12.8 mW SA engine chip for enabling the energy-efficient data aggregation of compressively sampled physiological signals on mobile platforms. Taking an algorithm-architecture codesign approach, we apply a combination of techniques to optimize the SA engine chip toward high energy efficiency. First, by applying algorithm reformulations, we reduce the CC and the data-dependent loops involved in the LS task by an order of magnitude. This saves dynamic and leakage energy from eliminated computation and reduced execution time per functionality, respectively. Second, we propose a configurable system architecture, in which all the computing resources are shared across different tasks. This maximizes the hardware utilization and minimizes the overhead of LS computation. Third, by introducing the shuffle-mode memory control scheme, we effectively cut down the memory usage for handling Cholesky factorization by half and saves another 40% of the total power from reduced memory leakage.

The SA engine chip integrated in 40 nm CMOS is able to support the simultaneous reconstruction of over 200 channels of physiological signals by consuming $<1\%$ of a smartphone's power budget. In a CS-based health monitoring system, the SA engine chip can enable a two-to-three-time lower energy at the sensor nodes through compressive sampling [5], while providing timely feedback and bringing signal intelligence closer to the user.

Appendix

A. Notations

The following conventions apply to the notations in this paper. A matrix is denoted as an upper-case bold letter (e.g., $\boldsymbol{A}$). A vector is denoted as a lower case letter (e.g., a). $\boldsymbol{a_i}$, when bolded, represents the ith column vector of matrix $\boldsymbol{A}$. a_i, when not bolded, represents an arbitrary vector indexed by i. $x\,(i)$ represents the ith element of vector $\boldsymbol{x}$. A set of index is denoted by an upper case Greek letter (e.g., Λ). $\boldsymbol{A}_\Lambda$, when bolded, represents the set of column vectors of $\boldsymbol{A}$ that are indexed by Λ, and $x\,(\Lambda)$ represents the set of elements of x that are indexed by set Λ.

B. Compressive Sensing

Let $\alpha \in \mathbb{R}^n$ be a compressible signal that has a sparse representation $x \in \mathbb{R}^n$ on a certain orthogonal basis $\boldsymbol{\Psi} \in \mathbb{R}^{\mathrm{n}\times\mathrm{n}}$, given as

$$\alpha = \boldsymbol{\Psi} x \tag{2}$$

where x is a k-sparse vector that contains only k nonzero elements, denoted as $x \in \mathbb{S}_k^n$. Then, the compressive sampling is performed by applying a linear mapping on α through a random matrix $\boldsymbol{\Theta} \in \mathbb{R}^{m\times n}$, expressed as

$$y = \boldsymbol{\Theta}\alpha + \beta \tag{3}$$

where β is an additive noise imposed by the sampling process. According to (2), the linear mapping in (3) is as if encoding the sparse coefficient x through another random matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ as

$$y = \boldsymbol{A}x + \beta \tag{4}$$

where $\boldsymbol{A}$ is uniquely defined by $\boldsymbol{A} = \boldsymbol{\Theta}\boldsymbol{\Psi}$. CS theorem tells us that as long as $\boldsymbol{\Theta}$ satisfies the null space property (NSP) and the restricted isometry property (RIP) of order $2k$ [3], [4], the signal information x (in the sparse domain) can be well preserved by the random encoding scheme in (3) or (4). This holds true even when the sampling matrix $\boldsymbol{\Theta}$ is a underdetermined matrix with $m < n$ (so does $\boldsymbol{A}$), which represents a dimensionality reduction from $\mathbb{R}^n$ to $\mathbb{R}^m$. In this case, the random measurement y is a compressed representation of the signal's sparse coefficient x that is encoded by $\boldsymbol{A}$. It is proven that $\boldsymbol{\Theta}$ randomly generated from sub-Gaussian distributions, such as random Bernoulli or random Gaussian matrices, can easily satisfy both the NSP and the RIP of order $2k$ given the condition of

$$m \geq C \cdot k \cdot \log\left(\frac{n}{k}\right) \tag{5}$$

where C is a constant. In the context of this paper, n, m, k denote the signal dimension, measurement dimension, and signal sparsity level, respectively. In addition, k/n and m/n denotes the signal sparsity ratio and the under-sampling ratio that indicate the data size reduction achievable by conventional orthogonal transformation-based compression methods and the compressive sampling method, respectively.

To recover the original signal α, or equivalently its sparse coefficient x, we need to solve the linear equation in (4). Note that (4) is an underdetermined system equation with infinite possible solutions. However, it can be proven that by utilizing the sparsity condition $x \in \mathbb{S}_k^n$ as prior knowledge, x can be robustly estimated by solving the ℓ_0 pseudonorm minimization problem, defined as

$$\min \parallel x\parallel_0, \quad \text{subject to } \parallel y - \boldsymbol{A}x\parallel_2 \leq \varepsilon \tag{6}$$

where ε is the error tolerance that should be greater than the noise's energy level given as $\parallel \beta\parallel_2 \leq \varepsilon$. The formulation in (6), a.k.a. the SA problem, is the optimization problem of finding the sparest vector out of the solution space constrained by the linear mapping in (4). Thanks to the rich research in the field of CS, the SA problem in (6) can be either solved by heuristic methods such as OMP [12] and stage-wise OMP (StOMP) [13], or be relaxed to a ℓ_1-norm minimization problem and solved by linear programming [3], [4].

REFERENCES

[1] C. Christopoulos, A. Skodras, and T. Ebrahimi, "The JPEG2000 still image coding system: An overview," *IEEE Trans. Consum. Electron.*, vol. 46, no. 4, pp. 1103–1127, Nov. 2000.

[2] D. Marpe, T. Wiegand, and G. J. Sullivan, "The H.264/MPEG4 advanced video coding standard and its applications," *IEEE Commun. Mag.*, vol. 44, no. 8, pp. 134–143, Aug. 2006.

[3] D. L. Donoho, "Compressed sensing," *IEEE Trans. Inf. Theory*, vol. 52, no. 4, pp. 1289–1306, Apr. 2006.

[4] E. J. Candès and M. B. Wakin, "An introduction to compressive sampling," *IEEE Signal Process. Mag.*, vol. 25, no. 2, pp. 21–30, Mar. 2008.

[5] F. Chen, A. P. Chandrakasan, and V. M. Stojanovic, "Design and analysis of a hardware-efficient compressed sensing architecture for data compression in wireless sensors," *IEEE J. Solid-State Circuits*, vol. 47, no. 3, pp. 744–756, Mar. 2012.

[6] U. Varshney, "Pervasive healthcare and wireless health monitoring," *Mobile Netw. Appl.*, vol. 12, no. 2–3, pp. 113–127, 2007.

[7] I. Goncharova, D. J. McFarland, T. M. Vaughan, and J. R. Wolpaw, "EMG contamination of EEG: Spectral and topographical characteristics," *Clin. Neurophysiol.*, vol. 114, no. 9, pp. 1580–1593, 2003.

[8] A. M. Dixon, E. G. Allstot, D. Gangopadhyay, and D. J. Allstot, "Compressed sensing system considerations for ECG and EMG wireless biosensors," *IEEE Trans. Biomed. Circuits Syst.*, vol. 6, no. 2, pp. 156–166, Apr. 2012.

[9] P. Maechler, P. Greisen, B. Sporrer, S. Steiner, N. Felber, and A. Burg, "Implementation of greedy algorithms for LTE sparse channel estimation," in *Proc. Conf. Record 44th Asilomar Conf. Signals Syst. Comput. (ASILOMAR)*, 2010, pp. 400–405.

[10] P. Maechler *et al.*, "VLSI design of approximate message passing for signal restoration and compressive sensing," *IEEE J. Emerging Sel. Topics Circuits Syst.*, vol. 2, no. 3, pp. 579–590, Sep. 2012.

[11] D. E. Bellasi, L. Bettini, C. Benkeser, T. Burger, Q. Huang, and C. Studer, "VLSI design of a monolithic compressive-sensing wideband analog-to-information converter," *IEEE J. Emerging Sel. Topics Circuits Syst.*, vol. 3, no. 4, pp. 552–565, Dec. 2013.

[12] J. A. Tropp and A. C. Gilbert, "Signal recovery from random measurements via orthogonal matching pursuit," *IEEE Trans. Inf. Theory*, vol. 53, no. 12, pp. 4655–4666, Dec. 2007.

[13] D. L. Donoho, Y. Tsaig, I. Drori, and J.-L. Starck, "Sparse solution of underdetermined systems of linear equations by stagewise orthogonal matching pursuit," *IEEE Trans. Inf. Theory*, vol. 58, no. 2, pp. 1094–1121, Feb. 2012.

[14] F. Ren, "A scalable VLSI architecture for real-time and energy-efficient sparse approximation in compressive sensing systems," Ph.D. dissertation, Dept. Electr. Eng., Univ. California, Los Angeles, CA, USA, 2015.

[15] F. Ren, C. Zhang, L. Liu, W. Xu, V. Owall, and D. Markovic, "A modified square-root-free matrix decomposition method for efficient least square computation on embedded systems," *Embedded Syst. Lett.*, vol. 6, no. 4, pp. 73–76, 2014.

[16] F. Ren, W. Xu, and D. Markovic, "Scalable and parameterised VLSI architecture for efficient sparse approximation in FPGAs and SoCs," *Electron. Lett.*, vol. 49, no. 23, pp. 1440–1441, 2013.

[17] L. Vandenberghe, "Applied numerical computing," LA, USA: UCLA Academic Publishing, 2013.

[18] A. L. Goldberger, *et al.*, "PhysioBank, Physiotoolkit, and Physionet: Components of a new research resource for complex physiologic signals," *Circulation*, vol. 101, no. 23, pp. e215–e220, 2000.

Fengbo Ren (S'10–M'15) received the B.Eng. degree from Zhejiang University, Hangzhou, China, in 2008, and the M.S. and Ph.D. degrees from the University of California, Los Angeles, CA, USA, in 2010 and 2014, respectively, all in electrical engineering.

In 2015, he joined the Faculty of the School of Computing, Informatics, and Decision Systems Engineering, Arizona State University, Tempe, AZ, USA. His research interests include hardware acceleration and parallel computing solutions for data analytics and information processing, with emphasis on compressive sensing, sparse coding, and deep learning frameworks.

Dr. Ren was the receipt of the 2012–2013 Broadcom Fellowship.

Dejan Marković (S'96–M'06) received the Ph.D. degree in electrical engineering from the University of California, Berkeley, CA, USA, in 2006.

He is a Professor of Department of Electrical Engineering at the University of California, Los Angeles (UCLA), Los Angeles, CA, USA. He is also affiliated with UCLA Bioengineering Department as a Co-Chair of the Neuroengineering field. His research interests include implantable neuromodulation systems, domain-specific architectures, embedded systems, energy harvesting, and design methodologies.

Dr. Marković was the recipient of an NSF CAREER Award in 2009, the 2014 ISSCC Lewis Winner Award for Outstanding Paper, the 2007 David J. Sakrison Memorial Prize, and also the corecipient of an ISSCC Jack Raper Award for Outstanding Technology Directions in 2010.