University of California
Los Angeles

# A Scalable VLSI Architecture for Real-Time and Energy-Efficient Sparse Approximation in Compressive Sensing Systems

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Electrical Engineering

by

**Fengbo Ren**

2015

Abstract of the Dissertation

# A Scalable VLSI Architecture for Real-Time and Energy-Efficient Sparse Approximation in Compressive Sensing Systems

by

**Fengbo Ren**

Doctor of Philosophy in Electrical Engineering

University of California, Los Angeles, 2015

Professor Dejan Marković, Chair

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 very sparse representations on some other orthogonal (non-Fourier) basis. This mismatch implies a large redundancy in Nyquist-sampled data, making compression a necessity prior to storage or transmission. Recent advances in compressive sensing (CS) theory offer us an alternative data acquisition framework, which can greatly impact power-starved applications such as wireless sensors. CS techniques provide a universal approach to sample compressible signals at a rate significantly below the Nyquist rate with limited information loss. Therefore, CS is a promising technology for realizing configurable, cost-effective, miniaturized, and ultra-low-power data acquisition devices for mobile and wearable applications.

However, the digital signal processing of compressively-sampled data involves solving a sparse approximation problem, which requires iterative-searching algorithms that have high computational complexity and require intensive memory access. As a result, existing software solutions are neither energy-efficient nor cost-effective for real-time processing of

compressively-sampled data, especially when the processing is to be performed on energy-limited devices. To solve this problem, this dissertation presents a scalable VLSI architecture that can be implemented on field-programmable gate arrays (FPGAs) or system-on-chips (SoCs) to perform dedicated-hardware-driven sparse approximation. A VLSI soft-IP core of the sparse approximation engine is developed in Verilog-HDL, which supports a floating-point data format with 10 design parameters, providing a high dynamic range and the flexibility for application-specific user customizations. Taking advantage of the algorithm-architecture co-design that leverages algorithm reformulations, configurable architectures, and efficient memory mapping schemes, the proposed VLSI architecture features a 100% utilization of the computing resources and is scalable in terms of computation parallelism and memory capability.

The hardware emulation of the soft-IP core on a 28-nm Xilinx Kintex-7 FPGA shows that our design achieves the same level of accuracy as the double-precision C program running on an Intel Core i7-4700MQ mobile processor, while providing 47–147× speed-up for ECG signal reconstruction. Furthermore, a 12–237 KS/s 12.8 mW sparse approximation engine chip is realized in a 40-nm CMOS technology for enabling the mobile data aggregation of compressively sampled biomedical signals in CS-based wireless health monitoring systems. The measurement results show that the sparse approximation engine chip operating at the minimum energy point achieves a real-time throughput for reconstructing 61–237 channels of biomedical signals simultaneously with $< 1\%$ of a mobile device's 2W power budget, which is 14,100× more energy-efficient than the software solver running on the CPU. For high-sparsity signal reconstruction, the sparse approximation engine chip is 76–350× more energy-efficient than prior hardware designs. With a <1% power budget of a mobile device, the 5.13 $mm^2$ sparse approximation engine chip integrated in 40-nm CMOS can enable a 2–3× energy saving at CS-based sensor nodes while providing timely feedback and bringing signal intelligence closer to the user, presenting a significant advantage for 24/7 wireless health monitoring.

The dissertation of Fengbo Ren is approved.

Miloš D. Ercegovac

Babak Daneshrad

William J. Kaiser

Dejan Marković, Committee Chair

University of California, Los Angeles

2015

*To my parents, and my wife.*

# Table of Contents

# List of Figures

# List of Tables

# Acknowledgments

My sincere and deep gratitude goes to my advisor, Professor Dejan Marković, who has guided my research since 2009. Dejan's extraordinary foresight constantly drives me to keep eyes on real-life problems and has directed my research to bridge theory with practical applications. His exceptional wisdom on refining things has made me understand the power of less and the importance of details, which not only impacted my writing and drawing styles but also improved my presentation skills profoundly. I am also indebted to him for his encouraging support for my academic job hunting.

I am grateful that I had the opportunities to work as intern in leading companies during my graduate study. These experiences are fun and eye-opening. I would like to thank Chih-Tsung Huang and his switch ASIC team in the Data Center Group at Cisco Systems Inc. for helping me understand the fundamentals of local area network and sharing with me their experience in VLSI architecture design and project management. I would also like to express my gratitude to David Garrett and the DSP Microelectronics Group at Broadcom Corporation for giving me the opportunity to participate in the design of state-of-the-art communication chips and connecting me with the world-leading chip designers for research discussions.

The research presented in this dissertation is partially funded by Broadcom Corporation. The chip fabrication is supported by TSMC.

Special gratitude goes to Wenyao Xu, Chia-Hisang Yang, Boyu Hu, Hao Yu, and Chenxin Zhang. The research discussions with them have always been inspiring and fruitful. I am also grateful to my labmates in the Parallel Data Architecture Group at UCLA for their valuable input to my research and their kind help on paper proofreading. In addition, I am very blessed to have a bunch of friends playing regular basketball games together at UCLA. This routine of exercise has helped me better handle the pressure from work both mentally and physically. I will miss the games played with Ningyi Wang, Roy Lee, Yen-Hsiang Wang, Hao Wu, Yafeng Zhang, Hao Xu, I-Ning Ku, and Shen Shen.

Most importantly, I would like to thank my parents and my wife for their endless love and support, which have always been giving me the strength and courage to overcome difficulties.

# Vita

2004–2008 B.E. (Electrical Engineering), Zhejiang University, Hangzhou, China.

2008–2010 M.S. (Electrical Engineering), UCLA, Los Angeles, California.

# Publications

**J5** F. Ren, C. Zhang, L. Liu, W. Xu, V. Öwall, and D. Marković, "A Square-Root-Free Matrix Decomposition Method for Energy-Efficient Least Square Computation on Embedded Systems, *IEEE Embedded Syst. Lett.*, vol. 6, no. 4, pp. 73-76, Dec. 2014.

**J4** F. Ren, W. Xu, and D. Marković, "A Scalable and Parameterized VLSI Architecture for Compressive Sensing Sparse Approximation, *IET Electron. Lett.*, vol. 49, no. 23, pp. 1440-1441, Nov. 2013.

**J3** F. Ren, H. Park, C.-K. K. Yang, and D. Marković, "Reference Calibration of Body-Voltage Sensing Circuit for High-Speed STT-RAMs, *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 60, no. 11, pp. 2932-2939, Nov. 2013.

**J2** R. Dorrance, F. Ren, Y. Toriyama, A. A. Hafez, C.-K. K. Yang, and D. Marković, "Scalability and Design-Space Analysis of a 1T-1MTJ Memory Cell for STT-RAMs," *IEEE Trans. Electron Devices*, vol. 59, no. 4, pp. 878-887, Apr. 2012.

**J1** F. Ren, and D. Marković, "True Energy-Performance Analysis of the MTJ-Based Logic-in-Memory Architecture (1-Bit Full Adder)," *IEEE Trans. Electron Devices*, vol. 57, no. 5, pp. 1023-1028, May 2010.

**C6** F. Ren, and D. Marković, "A Configurable 12-to-237KS/s 12.8mW Sparse-Approximation Engine for Mobile ExG Data Aggregation", *Proc. IEEE Inter. Solid-State Circuits Conf.*, Feb. 2015, to appear.

**C5** R. Dorrance, F. Ren, and D. Marković, "An Efficient Sparse Matrix-Vector Multiplication (SpMxV) Kernel For Sparse-BLAS on FPGAs", *Proc. ACM/SIGDA Inter. Symp. on Field-Programmable Gate Arrays*, Feb. 2014, pp. 161-170.

**C4** F. Ren, R. Dorrance, W. Xu, and D. Marković, "A Single-Precision Compressive Sensing Signal Reconstruction Engine on FPGAs", *Proc. Inter. Conf. on field-programmable Logic and Applications*, Sep. 2013, pp. 1-4.

**C3** F. Ren, H. Park, R. Dorrance, Y. Toriyama, C.-K. K. Yang, and D. Marković, "A Body-Voltage-Sensing-Based Short Pulse Reading Circuit for Spin-Torque Transfer RAMs (STT-RAMs)," *Proc. Inter. Symp. on Quality Electronic Design*, Mar. 2012, pp. 275-282.

**C2** H. Park, R. Dorrance, A. Amin, F. Ren, D. Marković, and C.-K.K. Yang, "Analysis of STT-RAM Cell Design with Multiple MTJs Per Access," *Proc. ACM/IEEE Inter. Symp. on Nanoscale Arch.*, Jun. 2011, pp. 32-36.

**C1** R. Dorrance, F. Ren, Y. Toriyama, A. Amin, C.-K.K. Yang, and D. Marković, "Scalability and Design-Space Analysis of a 1T-1MTJ Memory Cell," *Proc. ACM/IEEE Inter. Symp. on Nanoscale Arch.*, Jun. 2011, pp. 53-58.

**P2** D. Marković, and F. Ren, "A Scalable and Parameterized VLSI Architecture for Compressive Sensing Sparse Approximation, U.S. Patent, 20150032990, Jan. 2015.

**P1** K.-L. Wang, C.-K. Yang, D. Marković, and F. Ren, "Body Voltage Sensing Based Short Pulse Reading Circuit, International Patent, WO2013043738 A1, Mar. 2013.

# CHAPTER 1

# Introduction

## 1.1 Background and Scope of The Dissertation

In recent years, compressive sensing (CS) has attracted great research attention in fields of applied mathematics, computer science, and electrical engineering. CS theory is established on the fundamental fact that most natural signals are highly compressible—they can be well represented by only a small portion of their coefficients on a suitable basis [Ca08]. Figure 1.1 shows the Nyquist framework that is the basis of the digital industry today, where an analog signal is first sampled at a high temporal or spatial resolution constrained by the Nyquist frequency. Basically, the Nyquist sampling theorem requires to double the size (sampling rate) of the signal representation in the Fourier basis to avoid information loss. However, most natural signals have very sparse representations on some other orthogonal (non-Fourier) basis. This mismatch implies a large redundancy in Nyquist-sampled data, making compression a necessity prior to storage or transmission (e.g. JPEG-2000, MPEG-4, etc.). From the analog signal to the sparse information of interest (compressed digital signal), the information-acquiring path of the Nyquist framework seems to involve an unavoidable detour. A great question to ask is whether there exists a shortcut or a smarter way of sampling that can bridge the gap between analog signals and sparse information. The answer is compressive sensing.

CS theorems tell us that by performing a linear mapping on the signal with randomness, one is able to well encode the sparse information of a signal with the least amount of redundancy, implying that sampling and compression can be performed at the same time through a compressive sampling process. Interestingly, random encoding is such a powerful

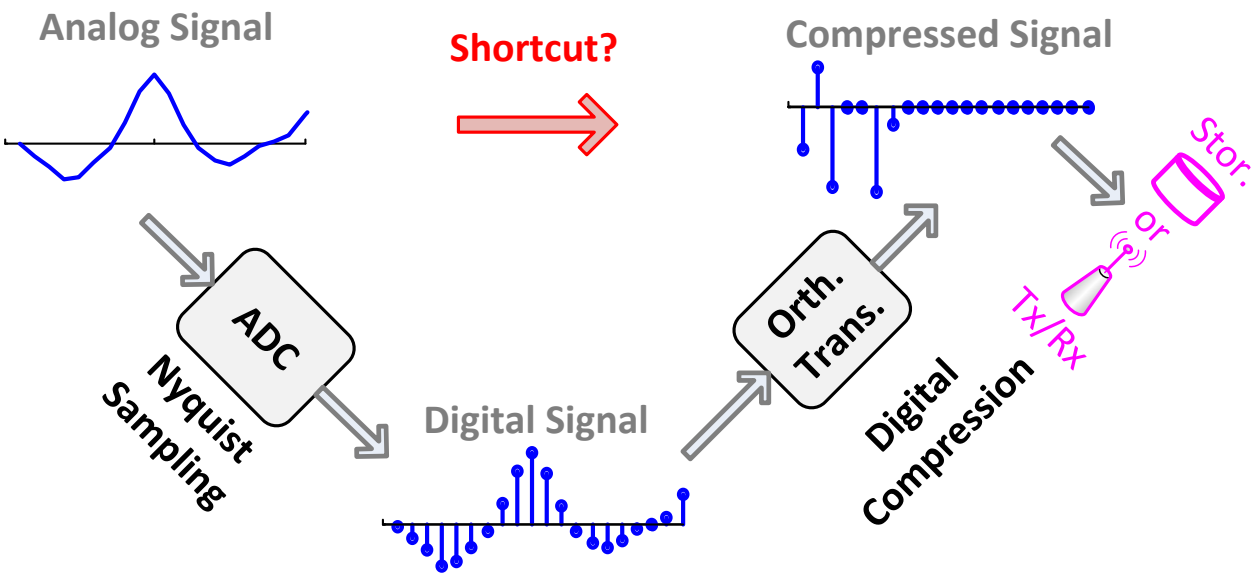



Figure 1.1: Nyquist framework.

scheme that the sparse information can be well preserved regardless of which domain a signal is sparse on. As a result, CS theorems offer us a universal framework for direct data sampling in a compressed format and analog-to-information conversion at a much lower frequency, surpassing the traditional limit of the Nyquist framework. This presents tremendous application values to the data acquisition devices that are sensitive to cost per device, portability, and battery life, especially in mobile and wearable applications.

However, great challenges remain in bringing CS technology into real-life applications. The signal recovery in the CS framework involves solving a sparse approximation (SA) problem, which is an optimization problem of finding the sparest vector from the solution space of a linear or quadratic equation. Unlike orthogonal transformation algorithms used in the Nyquist framework, SA algorithms involve iterative-searching process that leads to high computational complexity and intensive memory access (see Table 1.1). As a result, the software solutions are neither energy-efficient nor cost-effective for real-time processing of compressively-sampled data. For instance, state-of-the-art software solvers running on general computing platforms usually can achieve a real-time processing throughput of 50–500

[1]see Section 4.1 for details.

Table 1.1: Qualitative Comparison of SA and Orthogonal Transformation

| | **SA** | **Orth. Trans.** |
|---|---|---|
| Computational Complexity | $\mathcal{O}(n^3)$ | $\mathcal{O}(n \log n)$ |
| Operational Complexity[1] | High (Iterative) | Low |
| Memory Access Intensity | High | Low |

KSamples/s (KS/s) at the power consumption of 10–100 W (see Fig. 7.11). Unfortunately, such performance is a deal-breaker for most real-time CS applications, especially on mobile and wearable platforms, where the target throughput must be achieved very much limited power budgets.

To solve this problem, this dissertation presents a scalable VLSI architecture (the soft-IP core) that can be implemented in reconfigurable logic devices, such as field-programmable gate arrays (FPGAs), or in system-on-chips (SoCs) to perform hardware-accelerated SA for supporting the real-time and energy-efficient signal recovery in CS systems. The soft-IP core is developed in Verilog-HDL. It supports 10 user-specified parameters and can be customized at the compile time, providing the scalability for application-specific user customizations. Taking advantage of the algorithm-architecture co-design based upon the reformulated OMP algorithm, the proposed VLSI architecture features high parallelism, scalability, and configurability, in which all the computing resources are shared for executing the entire algorithm, leading to a 100% utilization of the computing resources.

## 1.2 CS-based Wireless Health Monitoring

Wireless health technology makes medical resources, including medical facilities, medicine and professionals, accessible to anyone, at anytime and anywhere. It enables reducing the medical cost, increasing the engagement between patients and doctors, and promoting connectivity of individuals to the world. The ultimate goal of wireless health is to

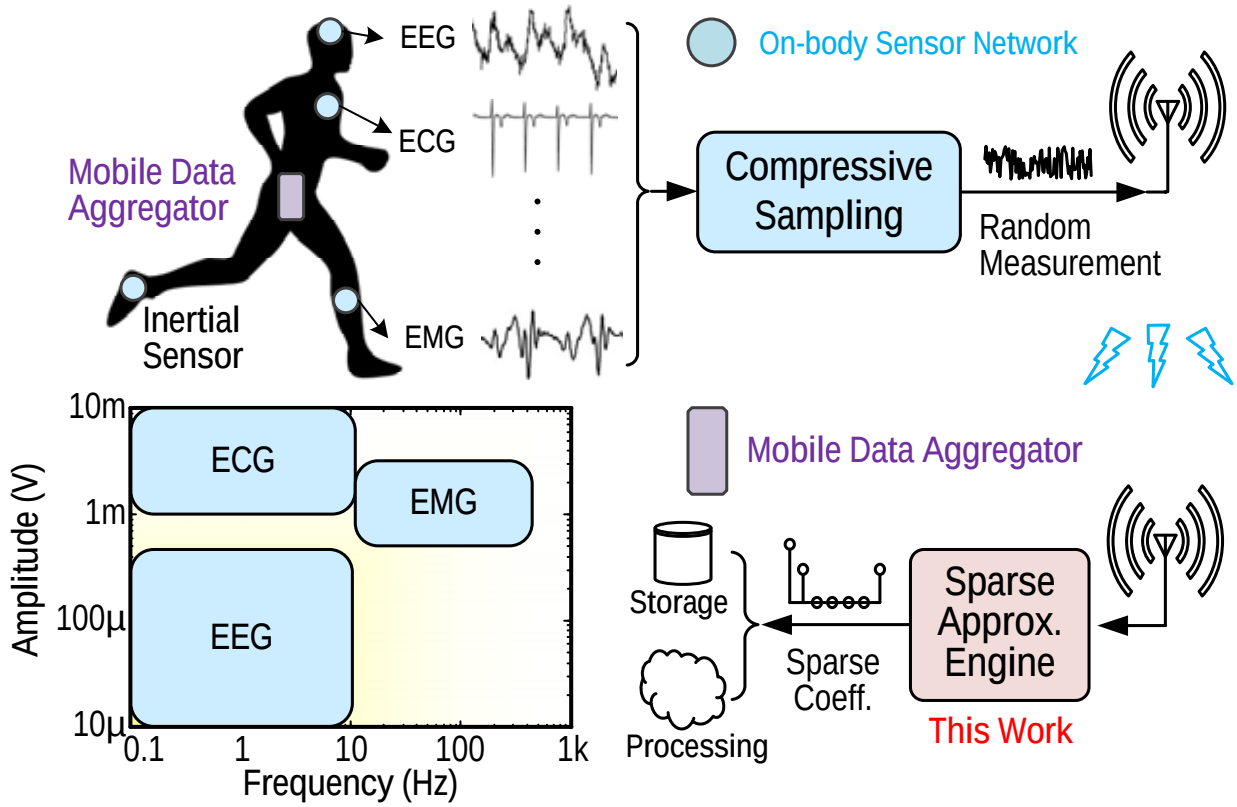



Figure 1.2: A CS-based wireless health monitoring system: always-on sensors compress data for low energy, a mobile data aggregator performs real-time signal reconstruction for timely prediction and proactive prevention.

revolutionize the operation model of the medical system to transform health related services from the system based on episodic examination, disease diagnosis, and treatment to one with continuous monitoring, disease prediction, and prevention [Var07, Xa13]. These changes will make health care systems more effective and economic, benefiting billions of individuals.

One of the key challenges in wireless health research is efficient sensing technology, as the continuous monitoring on health status will inevitably generate large amount of data for transmission, storage, and analysis. CS technology provides a viable solution to building the low-power and cost-effective on-body sensors for performing 24/7 wireless health monitoring. Figure 1.2 illustrates a CS-based wireless health monitoring system that includes always-on CS-based sensors that compress data for energy saving, and a mobile data aggregator

that performs real-time signal reconstruction for timely prediction and proactive prevention. Electrocardiography (ECG), electroencephalography (EEG), and electromyography (EMG) signals, collectively referred to as ExG, contain critical information about the human body status, thereby are the main targets of the monitoring [Ka11, Ca12, Da12].

The CS-based wireless health system has several advantages over the Nyquist-based counterpart. First of all, the total energy consumption of the sensor nodes in such a system is typically dominated by the radio frequency (RF) power for data transmission. By reducing the data size for wireless transmission through random encoding, a 2–3$\times$ total energy saving can be achieved at the sensor nodes [Ca12]. Second, since different biomedical signals have sparse representations on different basis, multiple compression standards must be implemented on the sensor nodes of the Nyquist-based system in order to reduce the data size for wireless transmission, making it an impractical approach due to the design complexity implied. Alternatively, a simple and universal random encoding scheme can be realized by using low-power microprocessor and pseudo-random number generators in the CS-based sensor nodes for a wide range of bio-medical signals, making the system highly scalable and easy to upgrade.

Note that deploying a mobile data aggregator to perform real-time signal reconstruction is crucial in this application scenario. The mobile data aggregator not only enables timely feedback but also brings signal intelligence closer to the user. To further reduce the data size for storage or post-processing, only the sparse coefficients of the signal are reconstructed. For supporting the real-time reconstruction of multi-channel ExG data without moving the energy needle of a mobile platform, the SA engine in Fig. 1.2 is required to achieve a throughput of $> 50$ KS/s at the power consumption of $< 20$ mW ($< 1\%$ of a mobile device's 2W power budget). Existing software solutions are not suitable for the intended application due to the low energy-efficiency for real-time recovery. A hardware solution based on VLSI implementation is critically needed.

ExG signals can span 3 orders of magnitude in amplitude (10 $\mu$–10 mV) and frequency (0.1 Hz–500 Hz), and can have a large difference in sparsity depending on the subject's activity. For instance, a low and high activity can produce a signal sparsity ratio of $< 2\%$

and $> 15\%$, respectively [Da12]. As a result, prior chip implementations of generic SA solvers [Ma10a, Ma12, Sa12] that are optimized to handle a limited dynamic range and a fixed problem size are not suitable for the intended application. To solve this problem, our soft-IP core supports a floating-point data format for achieving a large dynamic range and can be configured at the run time for handling flexible problem settings required by ExG signal recovery. Specifically, every implemented SA engine can be configured to handle flexible signal and measurement dimensions ($n$ and $m$), signal sparsity level ($k$), reconstruction basis ($\boldsymbol{A}$), and error tolerance ($\epsilon$).

## 1.3 Dissertation Outline

**Chapter 2** introduces the sparse approximation problem along with its applications. The preliminary knowledge about $\ell_p$ norm and signal sparsity is first explained, followed by the problem definition of SA. Lastly, several classical applications of the SA problem are reviewed and discussed.

**Chapter 3** reviews three different hardware-friendly SA algorithms. A benchmarking study on the SA algorithms is also conducted, based on which their potentials for efficient hardware implementations are discussed.

**Chapter 4** presents the algorithm design of the reformulated OMP. Specifically, the complexity characteristic of the original OMP algorithm is first analyzed. Then, three algorithm reformulation techniques are incorporated to (1) reduce the computational complexity of the least-squares (LS) task from $\mathcal{O}(mk^3)$ to $\mathcal{O}(mk^2)$ and (2) break down and simplify the LS task into 4 basic linear algebra (BLA) operations per iteration. Additionally, a hierarchical atom searching method is proposed to greatly reduce the computational complexity of the atom searching (AS) task.

**Chapter 5** proposes the scalable VLSI architecture of a SA engine soft-IP core co-designed based upon the reformulated OMP algorithm. The system architecture and the block diagram of computation cores are described. In addition, the memory control schemes

for handling Cholesky factorization and the dynamic configuration scheme of the system architecture for achieving 100% utilization of the computing resources are explained. Discussions on the scalability of the VLSI architecture is also provided.

**Chapter 6** elaborates on the FPGA evaluation of the developed soft-IP core. Specifically, the FPGA implementation results in comparison to prior designs are first reported. Additionally, the accuracy and performance benchmarking results in comparison to an Intel Core i7-4700MQ mobile processor are discussed.

**Chapter 7** shows the measurement results of a 12-to-237 KS/s 12.8 mW SA engine chip for mobile ExG data aggregation. It is demonstrated that the SA engine achieves a real-time throughput for reconstructing 61–237 channels of ExG data simultaneously with $< 1\%$ of a mobile device's power budget.

**Chapter 8** concludes the dissertation and proposes some possible directions for future research.

# CHAPTER 2

# Sparse Approximation (SA)

Sparse approximation (SA) has been formulated and researched by the signal processing community since 1990s [Nat95]. To date, SA has been recognized as a powerful framework in a variety of fundamental and applied research fields, including compressive sensing [Ca08, Don06], information coding [Ca05], computer graphics [Sa11], geographic data analysis [Ma09], etc. In this chapter, the sparse approximation problem along with its applications are introduced. The preliminary knowledge about $\ell_p$ norm and signal sparsity is first explained, followed by the problem definition of SA. Lastly, several classical applications of the SA problem are reviewed and discussed.

## 2.1 Preliminary

### 2.1.1 Notation

The following conventions apply to the notations in this dissertation. 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 $i^{\text{th}}$ column vector of matrix $\boldsymbol{A}$. $a_i$, when not bolded, represents an arbitrary vector indexed by $i$. $x(i)$ represents the $i^{\text{th}}$ element of vector $x$. When operating on a vector $x$, $sgn(x)$ denotes a sign vector, where $sgn(x(i)) = -1$ if $x(i) < 0$, and $sgn(x(i)) = 1$ otherwise. A set of index is denoted by an upper case Greek letter (e.g. $\Lambda$). $\boldsymbol{A_\Lambda}$, when bolded, represents the set of column vectors of $\boldsymbol{A}$ that are indexed by $\Lambda$, and $x(\Lambda)$ represents the set of elements of $x$ that are indexed by set $\Lambda$. When operating on a set $\Lambda$, $|\Lambda|$ denotes the cardinality of $\Lambda$, and $\Lambda^C$ denotes the absolute complement of $\Lambda$.

### 2.1.2 $\ell_p$ Norm

In digital signal processing (DSP), a signal is often modeled as a vector $x \in \mathbb{R}^n$, living in an $n$-dimensional space. The $\ell_2$ norm of $x \in \mathbb{R}^n$ is defined as

$$\|x\|_2 = (\sum_{i=1}^{n} |x(i)|^2)^{\frac{1}{2}}, \tag{2.1}$$

which represents the length of a vector in the Euclidean space, a.k.a. the Euclidean distance, or the square root of a signal's energy.

The $\ell_p$ norm of $x \in \mathbb{R}^n$ generalizes the length of a vector, or the distance between two points, in all $\ell_p$ spaces. For a real number $p \geq 1$, the $\ell_p$ norm of $x \in \mathbb{R}^n$ is defined as

$$\|x\|_p = (\sum_{i=1}^{n} |x(i)|^p)^{\frac{1}{p}}. \tag{2.2}$$

The $\ell_\infty$ norm is the limit of the $\ell_p$ norm when $p \to \infty$, which is defined as

$$\|x\|_\infty = \max_{i=1,\cdots,n} |x(i)|. \tag{2.3}$$

For all $p \geq 1$, the $\ell_p$ norms in (2.2) and (2.3) define a norm function that satisfies the following properties, where $c \in \mathbb{R}, x, y \in \mathbb{R}^n$.

1. If $\|x\|_p = 0$, then $x$ is the zero vector.
2. $\|c \cdot x\|_p = |c| \cdot \|x\|_p$.
3. $\|x + y\|_p \leq \|x\|_p + \|y\|_p$ (triangle inequality).

Besides, the unit circles in $\ell_p$ norms are all convex when $p \geq 1$. Some common examples of the unit circle in 2-dimensional (2-D) space are shown in Fig. 2.1.

Note that when $p < 0 < 1$, the resulting (2.2) defines a quasi-norm rather than a norm function, because the triangular inequality no longer holds true. In addition, the resulting unit circles become concave instead of convex (see Fig. 2.1). A special case is when $p = 0$, where the definition in (2.2) is not valid any more.

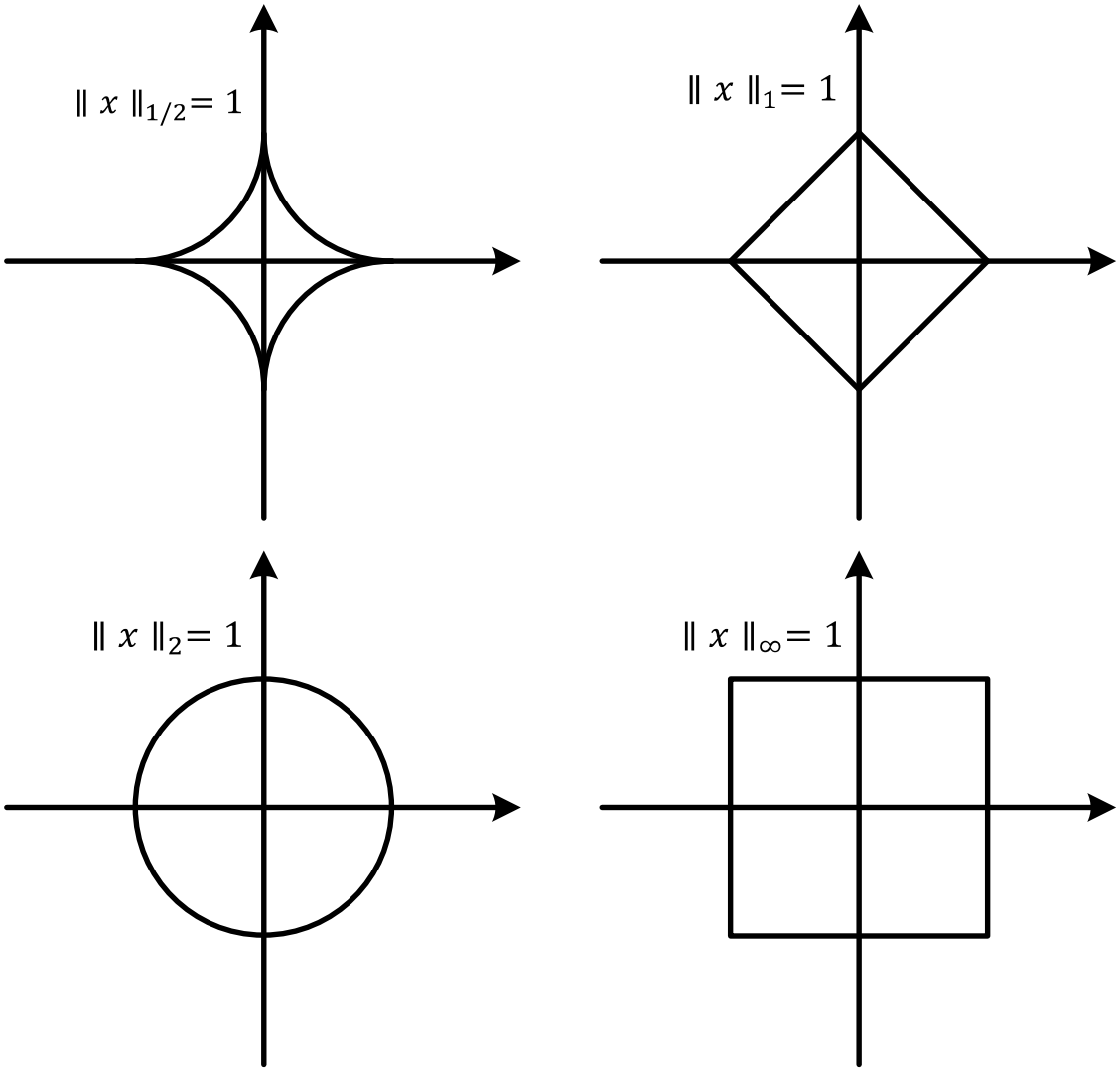



Figure 2.1: Illustration of unit circles in 2-D space.

### 2.1.3 Signal Sparsity

The sparsity of a signal is closely related to the $\ell_0$ pseudo-norm of its vector form. The $\ell_0$ pseudo-norm of a signal $x \in \mathbb{R}^n$ is defined as

$$\|x\|_0 = |supp(x)|, \tag{2.4}$$

where the operator $|\cdot|$ denotes the cardinality of a set, and $supp(x)$ is the support of $x$ defined as

$$supp(x) = \{i \mid x(i) \neq 0\}. \tag{2.5}$$

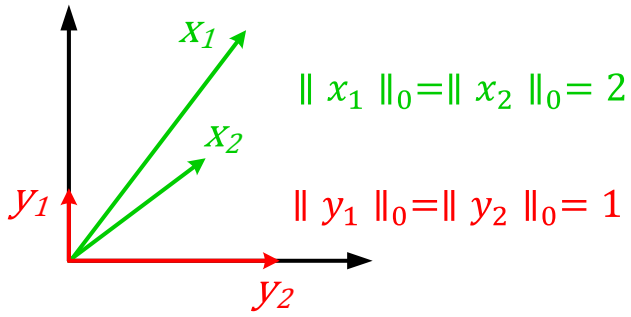



Figure 2.2: Illustration of 1- and 2-sparse vectors in 2-D space.

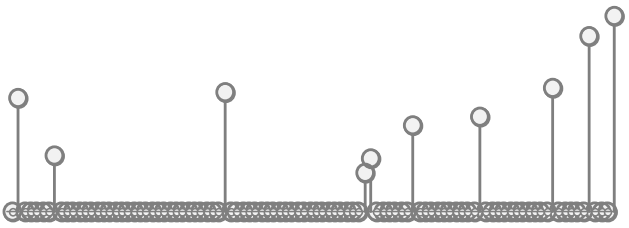


Figure 2.3: Example of a 10-sparse signal in 100-dimensional space $x \in \mathbb{S}_{10}^{100}$.

Namely, the $\ell_0$ pseudo-norm of $x$ measures the number of non-zero elements in the vector. Examples of 1- and 2-sparse vectors with different lengths in 2-D space are shown in Fig. 2.2. Unlike the $\ell_p$ norms in (2.2) and (2.3), the $\ell_0$ pseudo-norm contains no information about the vector length or the signal energy but indicates its sparsity level. A signal $x$ is defined to be $k$-sparse if

$$\|x\|_0 \leq k, \tag{2.6}$$

and the set of all $k$-sparse signals in $\mathbb{R}^n$ can be denoted as

$$\mathbb{S}_k^n = \{x \in \mathbb{R}^n \mid \|x\|_0 \leq k\}. \tag{2.7}$$

It is important to note that $\mathbb{S}_{k-1}^n \subsetneqq \mathbb{S}_k^n$ as indicated by (2.7). An example of a 10-sparse signal in 100-dimensional space $x \in \mathbb{S}_{10}^{100}$ is shown in Fig. 2.3.

In practice, the digital samples of natural signals are never ideally sparse on any basis due to the presence of various noise. However, their sorted coefficients often exhibit a power-law decay property on a proper basis [Ca08]. Specifically, given $x$ is the coefficient of a signal

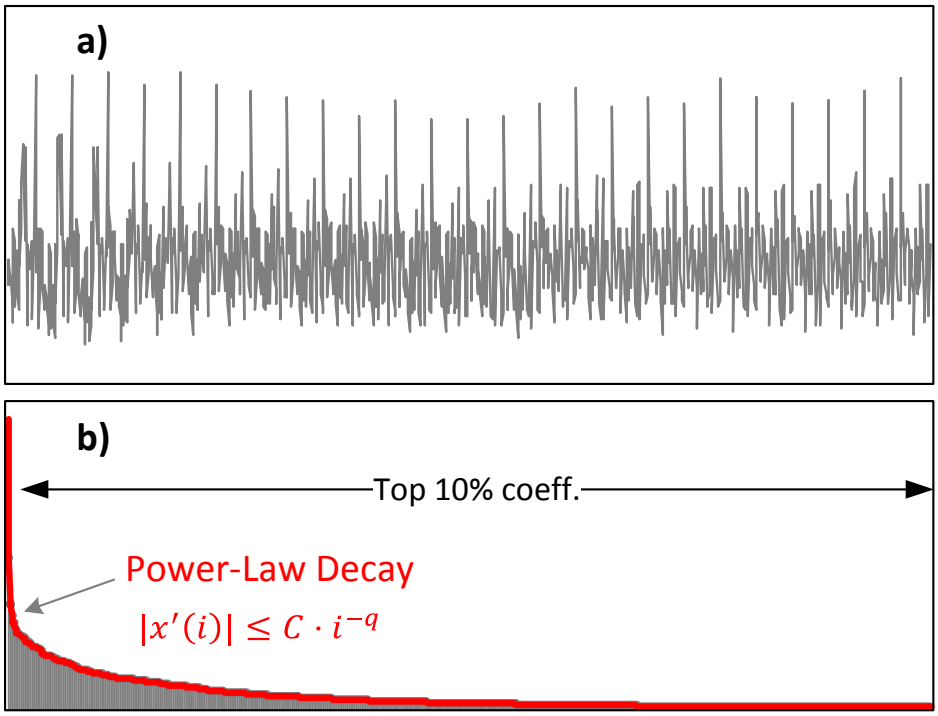



Figure 2.4: Example of (a) an EEG signal and (b) its top 10% coefficients (sorted by amplitude) on the DCT basis.

$\alpha$ on an orthogonal basis $\mathbf{\Phi}$ as $\alpha = \mathbf{\Phi} x$, the sorted coefficients $x' = sort(x)$ often obey a power-law decay defined by

$$|x'(i)| \leq C \cdot i^{-q}, \tag{2.8}$$

where $C$ is a constant and $q \geq 0$. Equation (2.8) indicates that on a proper basis, a small portion of the coefficients often carries a significant portion of the signal energy. An example of an electroencephalography (EEG) signal and its top 10% coefficients on the discrete cosine transform (DCT) basis are shown in Fig. 2.4. Note that the sorted coefficients follow a fast power-law decay, and most of the signal energy is carried by only the top 5% of the coefficients. Similarly, most natural signals are compressible in the way that the insignificant coefficients can be disregarded with very limited information loss. In other words, most natural signals can be well represented (with bounded errors) by a sparse vector on a proper basis [Ca08, Don06]. The sparsity level $k$ of the vector is determined by the target approximation error $\epsilon$ and the power-law decay factor $q$. The larger value of $\epsilon$ and $q$,

the lower $k$ of the sparse vector, and the higher the compression rate.

## 2.2 Problem Definition

Sparse approximation (SA), a.k.a $\ell_0$ pseudo-norm minimization, is the problem of finding the vector with the least $\ell_0$ pseudo-norm in the solution space defined by a linear equation. The mathematical formulation of the SA problem is defined as

$$\begin{aligned} &\min \|x\|_0, \\ &\textit{subject to } y = \boldsymbol{A}x, \end{aligned} \tag{2.9}$$

where $x \in \mathbb{S}_k^n$ is a sparse signal to be estimated, $y \in \mathbb{R}^m$ is an noiseless observation or a measurement taken from the linear mapping of $\boldsymbol{A}$ (usually $k \ll m < n$), and $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ is an under-determined matrix called the dictionary, representing a dimensionality reduction from $\mathbb{R}^n$ to $\mathbb{R}^m$. The linear constraint in (2.9) is an under-determined equation, which has infinite possible solutions. The SA problem is to find the sparsest vector out of the solution space constrained by $y = \boldsymbol{A}x$.

In practice, the observation $y$ is often contaminated by noise, denoted as $\beta$. The SA problem of estimating $x$, given a noisy observation $y = \boldsymbol{A}x + \beta$, is defined as

$$\begin{aligned} &\min \|x\|_0, \\ &\textit{subject to } \|y - \boldsymbol{A}x\|_2 \leq \epsilon, \end{aligned} \tag{2.10}$$

where $\epsilon$ is the energy level of the noise, given by $\|\beta\|_2 \leq \epsilon$. Note that the second-order cone constraint in (2.10) relaxes the solution space by taking into account the impact of noise.

One should note that $x \in \mathbb{S}_k^n$ implies only $k$ elements of $x$ are non-zero. The set of the non-zero elements in $x$ is called the active set, and the column vectors of $\boldsymbol{A}$ are called the atoms. According to $y = \boldsymbol{A}x$ (noiseless case), $y$ can be decomposed as a linear combination of only a few atoms of $\boldsymbol{A}$. Consider $\boldsymbol{A}$ as a over-complete dictionary containing many atoms that can possibly describe $y$, the SA problem is essentially to find the most sparse (compact) representation by the dictionary out of all possible cases. In general, the sparsity level (compactness) of $x$ is an indication of how well the dictionary can describe the observation.

The higher the sparsity, the better the dictionary does. In addition, the selective atoms corresponding to the active set are often closely correlated to $y$, otherwise the representation would not have been compact. By utilizing the principles behind sparse representation and interpreting its indications accordingly, the SA problem has found wide use in a wide range of applications. A few of the important applications will be discussed in the next section.

## 2.3 Applications of SA

Over the past decade, SA received a lot of attention in engineering research, especially in the field of electrical engineering and computer science. SA has found substantial use in a wide range of applications. In this chapter, we will review and discuss a few popular applications where SA is hailed as the key enabler for unprecedented advances.

### 2.3.1 Compressive Sensing (CS)

Digital electronic industry today relies on Nyquist sampling theorem, which requires to double the size (sample rate) of the signal representation in the Fourier basis to avoid information loss. However, most natural signals have very sparse representations on some other orthogonal (non-Fourier) basis. This implies a large redundancy in Nyquist-sampled data, making compression a necessity prior to storage or transmission. Recent advances in CS theory offer us an alternative data acquisition framework, which can greatly impact power-starved applications such as wireless sensors. CS provides a universal approach to sample compressible signals at a rate significantly below the Nyquist rate with limited information loss. Therefore, CS techniques present great application values for improving the data acquisition devices that are sensitive to cost, battery life, and portability.

#### 2.3.1.1 Sampling and Reconstruction

The sampling process in CS is often modeled as a linear system given by

$$y = \mathbf{\Psi}\alpha + \beta, \tag{2.11}$$

where $\alpha \in \mathbb{R}^n$ is the digital sample of a signal from the Nyquist sampling, $\boldsymbol{\Psi} \in \mathbb{R}^{m \times n}$ is a random matrix, $\beta \in \mathbb{R}^m$ is a random noise with bounded energy given as $\|\beta\|_2 \leq \epsilon$, and $y \in \mathbb{R}^m$ is a linear measurement or sampled data. Given that $\alpha$ can be well represented by a $k$-sparse vector $x \in \mathbb{S}_k^n$ on a orthogonal basis $\boldsymbol{\Phi}$ as $\alpha = \boldsymbol{\Phi} x$, (2.11) can be also expressed as

$$y = \boldsymbol{A}x + \beta, \tag{2.12}$$

where $\boldsymbol{A} = \boldsymbol{\Psi}\boldsymbol{\Phi} \in \mathbb{R}^{m \times n}$, is still a random matrix, called the sampling matrix, representing a linear mapping from $\mathbb{R}^n$ to $\mathbb{R}^m$. Since $\boldsymbol{A}$ is a fat matrix with $m < n$, the linear mapping also represents a dimensionality reduction from $n$ to $m$. Therefore, $y$ is a compressed measurement of the signal's sparse coefficient $x$, which is encoded by the sampling matrix $\boldsymbol{A}$.

Equation (2.11) and (2.12) indicate that applying a random projection on a compressible signal is as if taking a random measurement on the sparse coefficient of the signal. This means that by sampling the random measurements of a signal, one can indirectly access the sparse domain information of the signal. This presents significant advantages over the Nyquist framework (see Fig. 1.1) as long as the needed information can be exactly recovered from the random measurements.

In order to recover $x$ (or equivalently $\alpha$ as $\alpha = \boldsymbol{\Phi} x$) from $y$, the linear equation in (2.12) must be solved (assuming $\beta = \vec{0}$ for now). Note that (2.12) is an under-determined problem, which has infinite possible solutions. However, by utilizing the prior knowledge that $x \in \mathbb{S}_k^n$, it is possible to retrieve $x$ from the solution space by finding the sparsest solution. It is proven that a signal $x \in \mathbb{S}_k^n$ can be exactly recovered by solving the SA problem defined in (2.9) as long as $\boldsymbol{A}$ satisfies the Null Space Property (NSP) of order $2k$ [Ca08, Don06].

A necessary and complete condition for (2.9) to work is that every $x \in \mathbb{S}_k^n$ must have an

unique image through the linear mapping. In other words, for any two vectors $x, x' \in \mathbb{S}_k^n$,

$$\begin{gathered} \boldsymbol{A}x \neq \boldsymbol{A}x' \\ \Leftrightarrow \\ \boldsymbol{A}(x - x') \neq 0, \\ \Leftrightarrow \\ (x - x') \notin Null(\boldsymbol{A}), \end{gathered} \tag{2.13}$$

must hold true. It can be proven that if $x, x' \in \mathbb{S}_k^n$, then $x - x' \in \mathbb{S}_{2k}^n$. Therefore, a linear mapping $\boldsymbol{A}$ uniquely represents all $x \in \mathbb{S}_k^n$ if and only if

$$\mathbb{S}_{2k}^n \nsubseteq Null(\boldsymbol{A}). \tag{2.14}$$

A matrix $\boldsymbol{A}$ is said to satisfy the NSP of order $2k$ if (2.14) applies. The NSP of order $2k$ rigorously guarantees that if $x \in \mathbb{S}_k^n$, there exists one and only one $k-$sparse vector in the solution space of (2.12). Consequently, $x$ can be exactly recovered by solving the SA problem defined in (2.9).

A matrix $\boldsymbol{A}$ satisfies the Restricted Isometry Property (RIP) of order $2k$ if there exists a small constant $\delta_k \in (0, 1)$ such that

$$(1 - \delta_k) \leq \frac{\|\boldsymbol{A}\|_2}{\|x\|_2} \leq (1 + \delta_k), \tag{2.15}$$

holds true for all $x \in \mathbb{S}_{2k}^n$. Equation (2.15) assures that the length distortion of all $2k$-sparse vectors is strictly bounded by $\delta_k$ in the transformed space. The boundary of distortion indicates that $\boldsymbol{A}$ acts like an orthonormal matrix when operating on $2k-$sparse vectors—it preserves the distance and angle between any two $k-$sparse vectors so that they are as distinguishable in the transformed space as they are in the original space. Therefore, the NSP and the RIP of order $2k$ guarantee the success of signal recovery even in a noisy environment. When $\beta \neq \vec{0}$, it is proven that a signal $x \in \mathbb{S}_k^n$ can be exactly recovered by solving the SA problem with a second-order cone constraint given by (2.10) as long as $\boldsymbol{A}$ satisfies the NSP and the RIP of order $2k$, where $\epsilon$ is the upper bound of the square root of the noise energy, expressed as $\|\beta\|_2 \leq \epsilon$. In fact, the RIP is a sufficient (but not necessary) condition of the NSP [Ca08].

#### 2.3.1.2 Sampling Matrix

A sufficient condition for a normalized (column-wise) matrix $\boldsymbol{A}$ to satisfy the RIP of order $2k$ is

$$\mu(\boldsymbol{A}) < \frac{1}{2k-1}, \tag{2.16}$$

where $\mu(\boldsymbol{A})$ is the matrix coherence defined as

$$\mu(\boldsymbol{A}) = \max_{i \neq j} \frac{\boldsymbol{a_i}, \boldsymbol{a_j}}{\|\boldsymbol{a_i}\|_2 \cdot \|\boldsymbol{a_j}\|_2}. \tag{2.17}$$

Equation (2.16) and (2.17) state that the maximum correlation coefficient between any two atoms of $\boldsymbol{A}$ must be bounded. Otherwise, the basis of $\boldsymbol{A}$ would carry too much mutual information so that each component of the measurement will contain a lot of redundancy rather than new information of the signal. From this point of view, randomness plays an important role here, as independent random variables are hardly correlated. Therefore, random matrices inherently have low coherence.

It is proven that a random matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ consisting of independent and identically distributed (i.i.d.) variables following sub-Gaussian distributions, such as a random Gaussian or a random Bernoulli matrix, can satisfy the RIP of order $2k$ with an overwhelming probability if

$$m \geq C \cdot k \cdot \log(\frac{n}{k}), \tag{2.18}$$

where $C$ is a mysterious constant. Different from random Gaussian matrices, the entries of random Bernoulli matrices are either 0/-1 or 1 with equal probability. Implementing the linear mapping of such a matrix only requires addition operations, because the multiplication with $\pm 1$ is just a sign manipulation. This merit makes random Bernoulli matrices the preferred candidates for hardware implementation [Don06, Rom08, Da08]. In addition, partial orthogonal matrices whose rows are randomly selected from a full size $(n \times n)$ orthogonal matrix are also good candidates for potential hardware implementation [Da06]. One intrinsic advantage is that fast algorithms exist for computing the vector transformation by partial orthogonal matrices.

#### 2.3.1.3 Comparison to The Nyquist Framework

Equation (2.18) has very important implications. In the context of CS, $m$ is the measurement dimension, representing the data size resulting from the compressive sampling, which is proportional to the sampling rate $f_{CS}$. $n$ is the signal dimension, representing the data size resulting from the Nyquist sampling, which is proportional to $f_{Nyquist}$. $k$ is the sparsity level of a signal on the sparsest domain. Therefore, $k/n$ is the signal sparsity ratio, representing the relative sparsity level of a signal, and $m/n$ is the undersampling ratio that is equal to $f_{CS}/f_{Nyquist}$. Given that $k \ll m < n$, and $k/n$ is usually fixed a ratio (at least with a bounded range) for a certain type of signal, it can be inferred from (2.18) that

$$m \propto k. \tag{2.19}$$

Equation (2.19) means that the number of measurements needed in CS for successful recovery is proportional to the signal sparsity level on the sparsest domain. The higher the sparsity level, the fewer measurements are needed. When implemented in hardware, the relationship in (2.19) directly translates into the proportionality given as

$$f_{CS} \propto k. \tag{2.20}$$

The CS framework features several attractive advantages over the Nyquist framework. First of all, the sampling rate in CS is always proportional to the size of the signal presentation on its sparest domain (see (2.20)). In the Nyquist framework, the sampling rate is required to double the size of the signal representation on the Fourier basis only. Since most natural signals have much sparser representations on non-Fourier basis, the CS framework allows for data sampling at a much lower rate than the Nyquist framework. Another interpretation of (2.20) is that the sampling method in CS intrinsically performs compression (random encoding) at the same time, which not only reduces the data size but also may speed up the sampling process. Such a compressive sampling method enables the indirect access to the information of interest in a signal with the least amount redundancy. Second, the compressive sampling method is universal to all types of compressible signals [Ca08, Don06]. Unlike the Nyquist framework, where unique compression methods are

needed for different signals, the random encoding approach used in the CS framework is independent on the sparse domain of a signal. This makes the various data fusion with a unified hardware architecture practical. Last but not least, the compressive sampling method is democratic in the way that each sampled data (random measurement) carries similar amount of information, making it robust to sample loss or corruption [Ca08, Don06]. This property is a result from the low coherence of the sampling matrix guaranteed by the randomness (see (2.16)), which implies that the compression rate in the CS framework can become highly configurable given the target tolerance of reconstruction error.

Overall, the above-mentioned merits render CS a promising technology for realizing configurable, cost-effective, miniaturized, and ultra-low-power data acquisition devices for mobile and wearable applications [Da08, Ka11, Rom08, Ma09].

#### 2.3.1.4 CS Systems in Real Life

The first proof of concept of a working CS system is the signal-pixel camera developed by Durate et al. [Da08]. Compared to traditional cameras with a large image sensor array containing millions of pixels, the single-pixel camera is able to record high-resolution images with only a single photo image sensor, making the analog-to-digital (ADC) conversion much more cost-effective. This presents a paramount application value to the expensive imagers used for inspections in microscopy and life sciences such as integrated circuit imaging, material inspection, human tissue analysis, etc.

There are also a few successful applications of CS theory in the fields of medical imaging and biomedical engineering. The first one is sparse magnetic resonance imaging (MRI). Lustin et al. [La07] discovered that SA algorithms can effectively reconstruct MRI images by using 8 times fewer magnetic resonance samples as compared to the traditional MRI reconstruction method, which leads to faster data acquisition. Kanoun et al. [Ka11] implemented a CS-based data acquisition device for sampling ECG signals. Normally, ECG signals have to be sampled at more than 100 Hz (Nyquist rate) in order to capture the fine-grained details of heart status. This will produce a huge amount of data if it is to

be monitored continuously over a long time. By adopting the CS framework in their ECG device, the sampled data size is effectively reduced by 2×.

Looking into the future, we believe CS is a promising technology for realizing the low-power sensor nodes for the 24/7 wireless health monitoring. On one hand, the compressive sampling at sub-Nyquist rates allows for reduced data size for transmission at the sensor nodes, thereby saving energy from wireless transmission. On the other hand, the simplicity and universality of the compressive sampling process can simplify and unify the building block pipelining in the sensor system and enable the unified sensing of a wide range of biomedical signals as a whole. However, for the purpose of timely prediction and proactive prevention desired by such an application, the sampled data must be recovered in real-time for further processing on a mobile platform. The key challenge is that general purpose processors cannot provide the real-time performance with the energy efficiency required by mobile platforms. In Chapter 7, a silicon solution developed by this work will be presented.

### 2.3.2 Sparse Representation Based Classification (SRC)

Classification in an important problem in the field of machine learning. The most common application is in biometrics, such as face, iris, and behavior recognition. Prior study has revealed that feature selection is the most critical step in the standard signal classification framework in terms of the recognition accuracy [Bis06]. However, the complexity of biological features and the lack of prior knowledge about them create difficulties in selecting the optimal features for different biometric classification problems.

Applying the principle of SA to classification problems opens the gate for solving this problem [Wa09]. Consider a classification problem that has $p$ different classes, and each class $i$ has $n_i$ training samples with each sample having $m$ features. In total, there are $n = \sum_{i=1}^{p} n_i$ training samples. If we collect all the training samples to form a training matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ as

$$\boldsymbol{A} = [\boldsymbol{A_1}, \cdots, \boldsymbol{A_p}] = \left[a_{11}, \cdots, a_{1n_1}, \cdots, a_{p1}, \cdots, a_{pn_p}\right], \tag{2.21}$$

where $\boldsymbol{A}_i$ is the training matrix for class $i$ formed by the $n_i$ training samples, and $a_{ij}$ is the

$j^{\text{th}}$ training sample in class $i$, then a given test object $y$ with unknown label can be well approximated as a linear combination of the training samples as

$$y = \boldsymbol{A}x = \boldsymbol{A_1}x(\Omega_1) + \cdots + \boldsymbol{A_p}x(\Omega_p), \tag{2.22}$$

where $\Omega_i$ is the subset of indices that corresponds to class $i$. Note that the coefficient $x$ is expected to be sparse if the training samples are sufficiently representative. The principle behind this statement is that a signal should be well reconstructed by using a few samples of the same class with bounded reconstruction error. In this sense, the sparse reconstruction coefficient $x$ actually contains the information for determining the class label $y$.

In case of $m < n$, where we have infinite possible coefficients that can reconstruct $y$, the sparsest one can be obtained by solving the SA problem given in (2.9). Then, the label $\hat{y}$ of $y$ can be determined as the class that best reconstructs $y$ with the lowest reconstruction error as

$$\hat{y} = \arg\min_i \|y - \boldsymbol{A_i}x(\Omega_i)\|_2. \tag{2.23}$$

Simulation results show that the SRC framework exhibits higher recognition rate with less dependency on feature selection methods as compared to conventional classifiers [Bis06].

There are plenty of successful applications of sparse approximation algorithms in biometric classification. Wright et al. [Wa09] first proposed to use SA algorithms for human face recognition problems. It is shown that the classification accuracy of SRC framework is constantly better than conventional classifiers, such as principal component analysis (PCA), linear discriminant analysis (LDA), with different features. Also, it is experimentally shown that the performance of SRC framework is not sensitive to the feature selection method. Xu et al. [Xa12] adapted the framework in [Wa09] with Bayesian formula for sensor location and human activity co-recognition. Experimental results show that the recognition rate of SRC framework reaches 87% on average, which is significantly improved compared the two classical methods, Nearest Neighbor (74%) and Nearest Subspace (79%). Pillai et al. [Pa11] applied the SRC framework with random projection to iris recognition, which achieved an excellent accuracy of 99.15% with better robustness than other classification methods.

### 2.3.3 Signal Denoising

Another natural application of SA algorithms is signal denoising. Noise is inevitable in signals and their energy usually spans across the whole spectrum. Denoising has been an important topic in signal processing for decades. Note that frequency filtering although can remove the out-of-band noise components. The noise components in the band of interest still exists and could deteriorate the signal-to-noise ratio (SNR). For complete denosing, a global solution is needed.

Assume that in the formulation of (2.12), $y$ is an observation of an ideally noise-free signal $\hat{y}$ such that $y$ is contaminated by an additive noise $\beta$ with bounded energy as $\|\beta\|_2 \leq \epsilon$. Also, assume $\hat{x}$ is the sparse domain coefficient of $\hat{y}$ on basis $\boldsymbol{A}$ as $\hat{y} = \boldsymbol{A}\hat{x}$. Then, given the noisy observation $y$, $\hat{x}$ can be identified by solving the SA problem in (2.10) or equivalently the BPDN problem in (3.2). A clean signal $\hat{y}$ can be then recovered as $\hat{y} = \boldsymbol{A}\hat{x}$. The signal denoising through BPDN relies on the fact that white noise usually has dense energy spectrum on almost every basis. As a result, by finding the sparsest coefficient that can approximate $y$ with a certain error tolerance $\epsilon$ as shown in (3.2), the components of dense noise will be consequently disregarded.

Gaudes et al. [Ga11] used BPDN to clean up the noise in functional MRI images, which enables more accurate blood oxygenation level analysis. Compared to paradigm free mapping techniques which reaches 95% recognition rate [Ca13], the BPDN method can detect all task-related events in brain blood veins. Xu et al. [Xa06] proposed a complete solution based on BPDN to remove the noise from electroencephalography (EEG) signals. With a realistic head model processed by SA algorithms, the analytic correction rate is improved by 3-5$\times$ under different SNR conditions. Similarly, the same concept can also be used to restore signals from the corruption by partially gross errors or globally small errors [E 08, Ca10].

### 2.3.4 Data Separation

Data separation or decomposition is one of the most fundamental problems in signal processing. It involves decomposing a signal or image into superposed components

contributed by different sources. Consider symphonic music for instance, which can be regarded as the superposition of acoustic components generated from different instruments. Can we separate the contributions from each instrument as if it is played and recorded separately? In practice, such a problem is often an under-determined problem due to the lack of knowledge about the mixing weights and the original sources. However, literature has shown that by using sparsity as prior knowledge, such separation tasks might be possible through the help of SA algorithms [Sa05].

Assume that a signal $y$ can be well represented as the superposition of $d$ different components with each coming from a unique source, expressed as

$$y = z_1 + z_2 + \cdots + z_d + \beta, \tag{2.24}$$

where $z_i$ is the $i^{\text{th}}$ component, and $\beta$ is an error vector that contains noise or mismatches. With the similar concept introduced in Section 2.3.2, it is possible to find an orthogonal basis or to construct a complete or even over-complete dictionary $\boldsymbol{A_i}$ for each $z_i$ such that there exist a sparse representation $x_i$ on $\boldsymbol{A_i}$ for each $z_i$, expressed as

$$z_i = \boldsymbol{A_i} x. \tag{2.25}$$

Replacing $z_i$ in (2.24) with (2.25), we can deliver (2.12) in a partitioned form as

$$y = [\boldsymbol{A_1}, \cdots, \boldsymbol{A_d}] \begin{bmatrix} x_1 \\ \vdots \\ x_d \end{bmatrix} + \beta = \boldsymbol{A}x + \beta, \tag{2.26}$$

where $\boldsymbol{A}$ is an over-complete dictionary consisting of $\boldsymbol{A_i}$, and $x$ is a sparse vector composed by $x_i$. In this case, one can actually estimate $x$ by solving the SA problem in (2.10), given an object signal $y$ and a customized dictionary $\boldsymbol{A}$. As long as each $\boldsymbol{A_i}$ leads to a sparse representation over the part of the signal it is contributing while being highly insufficient in representing the remaining parts, each estimated $x_i$ will become a unique and sparse representation of a separated part of the signal. The corresponding component $z_i$ can be separated from $y$ by back-projecting $x_i$ to the original domain given by (2.24).

Note that each $z_i$ is a linear combination of a few atoms of $\boldsymbol{A_i}$, indicating that the separated component must be well described by the corresponding sub-dictionary. Consequently, one can decompose or separate a certain component from a signal by carefully designing the dictionary. For instance, the periodic and piece-wise-smooth components, such as the texture and the contour part of an image, can be separated by incorporating a Gabor system and a curvelet basis in the dictionary, respectively [Fa09]. Wavelet basis can be used to separate the singular component among smoothness, such as a spike in a 1-D time series or a point in a 2-D image [Kut12]. In addition, over-complete basis composed by selective training samples can be used to separate the component that is closely correlated to the selected samples [Wa09].

# CHAPTER 3

# SA Algorithms

Because the objective function in (2.9) and (2.10) is non-convex, finding the true solution to the SA problem is NP-hard [Ca08]. Fortunately, a rich variety of prior studies have suggested that the SA problem can be translated into a more friendly form through a convex relaxation of the objective function [Ca98, Tib96, Ta79]. Specifically, it has been proven that the solution to the SA problem defined in (2.9) coincides with that of the $\ell_1$ norm minimization problem, a.k.a the basis pursuit (BP) problem, defined as

$$\begin{aligned} &\min \|x\|_1, \\ &\textit{subject to } y = \boldsymbol{A}x, \end{aligned} \tag{3.1}$$

with an overwhelming probability [Ca08]. Similarly, in the case of a noisy environment, the solution to (2.10) coincides with that of the basis pursuit denoising problem (BPDN), defined as

$$\begin{aligned} &\min \|x\|_1, \\ &\textit{subject to } \|y - \boldsymbol{A}x\|_2 \leq \epsilon. \end{aligned} \tag{3.2}$$

Note that the formulations in (3.1) and (3.2) are convex optimizations. Therefore, the BP and BPDN problems can be solved by general convex optimization algorithms such as interior point methods (IPM) [Ba04]. In addition, the BP problem in (3.1) is also a linear programming (LP) problem, which can be solved by LP-based algorithms, such as Simplex algorithm. However, these general methods exhibit extremely high computational complexity with complicated data-flow control schemes and are not suitable for efficient hardware implementations.

In this chapter, we review three different hardware-friendly SA algorithms. A benchmarking study on the SA algorithms is also presented, based on which their potentials for efficient

Table 3.1: Pseudo-Code of the OMP Algorithm

| Step | Operation |
|---|---|
| 1 | $r_0 = y,\ x_0 = \vec{0},\ d = \vec{0},\ \Lambda_0 = \varnothing,\ t = 1.$ |
| 2[1] | $c = \boldsymbol{A}^T r_{t-1},\ \varphi = \arg\max_i \lvert c(i)\rvert,\ \Lambda_t = \Lambda_{t-1} \cup \varphi.$ |
| 3 | $x(\Lambda_t) = \arg\min_\alpha \lVert y - \boldsymbol{A}_{\boldsymbol{\Lambda_t}}\alpha\rVert_2^2$ . |
| 4 | $r_t = r_{t-1} - \boldsymbol{A}_{\boldsymbol{\Lambda_t}} x(\Lambda_t),\ t = t + 1.$ |
| 5 | If $\lVert r_t\rVert_2 \leq \epsilon$, break, otherwise, go to step 2. |

hardware implementations are discussed.

## 3.1 Hardware-Friendly Algorithms

### 3.1.1 Orthogonal Matching Pursuit (OMP)

OMP is one of the greedy algorithms that can efficiently solve the SA problems in (2.9) and (2.10). OMP is able to approximate a $k$-sparse solution in exact $k$ iterations [Ta07]. The concept of OMP is built upon the following observation. When $x \in \mathbb{S}_k^n$ has only $k$ non-zero elements, the linear measurement $y = \boldsymbol{A}x$ can also be represented as $y = \boldsymbol{A}_{\boldsymbol{\Lambda}} x(\Lambda)$, where $\Lambda$ is the index set of the non-zero elements in $x$, called the active set. Therefore, to recover a $x \in \mathbb{S}_k^n$, we only need to recover the unknown part of $x$ as $x(\Lambda)$. Note that given $\boldsymbol{A}_{\boldsymbol{\Lambda}} \in \mathbb{R}^{m\times k}$ and $m > k$, $x(\Lambda)$ can be best estimated by solving the least-squares (LS) problem defined by

$$\min \lVert y - \boldsymbol{A}_{\boldsymbol{\Lambda}} x(\Lambda)\rVert_2^2 \ , \tag{3.3}$$

as long as the active set $\Lambda$ can be identified. Consequently, identifying the correct active set is the key task in the OMP algorithm.

The pseudo-code of the OMP algorithm is described in Table 3.1. The algorithm starts

[1] Assuming all the atoms in $\boldsymbol{A}$ are normalized.

from an initial estimation $x_0 = \vec{0}$ and a residue $r_0 = y - \boldsymbol{A}x_0 = y$. In iteration $t$, the correlation coefficients $c$ of all the atoms in $\boldsymbol{A}$ and the current residue $r_{t-1}$ are computed as

$$c = \boldsymbol{A}^T r_{t-1}. \tag{3.4}$$

Then, the index of the single atom that has the largest correlation coefficient in magnitude is added to the active set $\Lambda_{t-1}$, and a new estimation $x_t$ is made by solving the LS problem shown in (3.3) based on the updated $\Lambda_t$. Unless the estimation error $\|r_t\|_2$ meets the stopping criterion, iteration $t+1$ will be performed for getting a better estimation. Note that in OMP, the updated residue $r_t$ is always orthogonal to the active set atoms $\boldsymbol{A}_{\boldsymbol{\Lambda_t}}$ in iteration $t$. As a result, zero correlation coefficients of these atoms should be expected in the next iteration. This guarantees no active atom will be duplicated in the active set. In addition, the active set size increases by one in every iteration, so does the sparsity of the estimated solution. This incremental fashion enforces OMP to reconstruct $x$ with as few elements as possible, thereby approaching the sparsest solution to (2.9) and (2.10).

The OMP algorithm is of great interest to hardware implementations for two reasons. First, the atom searching part of OMP (Step 2 in Table 3.1) only involves inner product and scalar comparison operations. This part of the computation has low data dependency and can be parallelized. In addition, comparison operations often have a low precision requirement. Consequently, high-level quantization can be applied to significantly simplify the computation load of the hardware implementation. Second, OMP holds a $k$-iteration-solution property, meaning that the total number of iterations of the OMP algorithm is bounded by $k$. This property is attractive in the sense that it sets a lower-bound on the system throughput, especially for the applications featuring high signal sparsity (low $k$).

### 3.1.2 Homotopy

Homotopy [Da06, Ea04] is a fast algorithm that is developed from the least absolute shrinkage and selection operator (LASSO) problem defined as

$$\begin{gathered} \min \|y - \boldsymbol{A}x\|_2^2, \\ \textit{subject to } \|x\|_1 \leq q. \end{gathered} \tag{3.5}$$

or equivalently, the unconstrained version of the LASSO problem defined as

$$\min \frac{1}{2}\|y - \boldsymbol{A}x\|_2^2 + \lambda\|x\|_1, \tag{3.6}$$

where $\lambda$ is a Lagrange multiplier. The LASSO problem has a close relationship to the BP problem. Assuming the solution to the LASSO problem in (3.5) is $x_q$ for a given value of $q$, then the solution path $\widetilde{x_q}$ defined by

$$\widetilde{x_q} = \{x_q \mid q \in [0, +\infty)\}, \tag{3.7}$$

starts from $x_q = 0$ for $q = 0$ and converges to the solution to the BP problem in (3.1) as $q$ increases. Similarly, the solution path $\widetilde{x_\lambda}$ of the unconstrained version of the LASSO problem in (3.6), defined by

$$\widetilde{x_\lambda} = \{x_\lambda \mid \lambda \in [0, +\infty)\}, \tag{3.8}$$

starts from $x_\lambda = 0$ for large $\lambda$ and converges to the solution to the BP problem in (3.1) as $\lambda$ approaches zero. The Homotopy algorithm is built upon the following observations. First, the solution path $\widetilde{x_\lambda}$ of (3.6) is polygonal or piece-wise linear. Second, the vertices of $\widetilde{x_\lambda}$ correspond to the change of the sparsity level of $x_\lambda$. Specifically, either a new atom is added to or an old atom is removed from the active set at each vertex of the solution path $\widetilde{x_\lambda}$. Therefore, the Homotopy algorithm identifies the active set by tracing the solution path $\widetilde{x_\lambda}$ along the vertices as $\lambda$ approaches 0.

The pseudo-code of Homotopy algorithm is described in Table 3.2. The algorithm starts to trace the solution path $\widetilde{x_\lambda}$ from $x_0 = \vec{0}$ with $\lambda_0$ being the largest correlation coefficient between $y$ and the atoms of $\boldsymbol{A}$. Note that when $\lambda \in [\lambda_0, +\infty)$, $x_0 = \vec{0}$ is always the solution to (3.6) [Da06]. When $\lambda = {\lambda_0}^-$, the solution starts to change linearly, leading to the first vertex on $\widetilde{x_\lambda}$. In iteration $t$, the solution path $\widetilde{x_\lambda}$ remains linear within $\lambda \in [\lambda_t, \lambda_{t-1})$ as long as the following two conditions

$$c(\Lambda_t) = \lambda \cdot sgn(x_\lambda(\Lambda_t)), \tag{3.9}$$

[2] if $x < 0$, $sgn(x) = -1$, otherwise, $sgn(x) = +1$.
[3] $\Lambda_{(t-1)}^C$ is the absolute complement of $\Lambda_{(t-1)}$.

Table 3.2: Pseudo-Code of the Homotopy Algorithm

| Step | Operation |
|---|---|
| 1 | $r_0 = y,\ d = \vec{0},\ c = \boldsymbol{A}^T r_0,\ \lambda_0 = \max_i \lvert c(i)\rvert,$<br>$\Lambda_0 = \arg\max_i \lvert c(i)\rvert, x_0 = \vec{0},\ t = 1.$ |
| $2^2$ | Solve $\alpha$ from $\boldsymbol{A}^T_{\boldsymbol{\Lambda_{t-1}}} \boldsymbol{A}_{\boldsymbol{\Lambda_{t-1}}} \alpha = sgn(c(\Lambda_{t-1})),\ d(\Lambda_{t-1}) = \alpha$ |
| $3^3$ | $v = \boldsymbol{A}_{\boldsymbol{\Lambda_{t-1}}} d(\Lambda_{t-1}),$ |
| 4 | $\varphi^- = \arg\min_{i\in\Lambda_{t-1}} \frac{-x_{t-1}(i)}{d(i)}$<br>$s = \min_{i\in\Lambda_{t-1}} \frac{-x_{t-1}(i)}{d(i)},\ z^- = \max(s, 0),$<br>$\varphi^+ = \arg\min_{j\in\Lambda^C_{t-1}}\ \min(\frac{\lambda_{t-1}+c(j)}{1+\boldsymbol{a}_{\boldsymbol{j}}^T v}, \frac{\lambda_{t-1}-c(j)}{1-\boldsymbol{a}_{\boldsymbol{j}}^T v}),$<br>$z^+ = \min_{j\in\Lambda^C_{t-1}}\ \min(\frac{\lambda_{t-1}+c(j)}{1+\boldsymbol{a}_{\boldsymbol{j}}^T v}, \frac{\lambda_{t-1}-c(j)}{1-\boldsymbol{a}_{\boldsymbol{j}}^T v}),$<br>if $z^- < z^+,\ z = z^-,\ \Lambda_t = \Lambda_{t-1} \setminus \varphi^-,$<br>otherwise, $z = z^+,\ \Lambda_t = \Lambda_{t-1} \cup \varphi^-.$ |
| 5 | $\lambda_t = \lambda_{t-1} - z,\ x_t = x_{t-1} + z \cdot d,$<br>$r_t = r_{t-1} - z \cdot v,\ c = \boldsymbol{A}^T r_t,\ d = \vec{0}.$ |
| 6 | If $\lambda_t \le \epsilon$ or $\lVert r_t\rVert_2 \le \epsilon$, break, otherwise, go to step 2. |

and

$$|c(\Lambda_t^C)| \le \lambda, \tag{3.10}$$

are satisfied [Da06]. The operator $(\cdot)^C$ in (3.10) is the absolute complement of a set. Therefore, Homotopy algorithm finds the value of $\lambda_t$ that leads to the next vertex on $\widetilde{x_\lambda}$ by checking the breaking points of condition (3.9) and (3.10). As $\lambda$ decreases from $\lambda_{t-1}$, the dissatisfaction of condition (3.9) infers that a bad atom should be removed from the active set. In contrast, the dissatisfaction of condition (3.10) infers that a new atom should be added to the active set. Note that the estimation $x_\lambda$ is updated at every vertex jump based on (3.9). The whole algorithm terminates when either $\lambda_t$ or the estimation error $\lVert r_t\rVert_2$ is small enough to produce an accurate estimate.

Different from OMP, Homotopy allows bad atoms to be removed from the active set so

Table 3.3: Pseudo-Code of the IST Algorithm

| Step | Operation |
|---|---|
| 1 | $r_0 = y,\ x_0 = \vec{0},\ c_0 = \boldsymbol{A}^T r_0,\ \lambda_0 = \max_i \lvert c(i) \rvert,\ t = 1.$ |
| 2[4] | $x_t = \eta_{\lambda_{t-1}}(x_{t-1} + c_{t-1}),\ \lambda_t = \mu\lambda_{t-1}.$ |
| 3 | $r_t = y - \boldsymbol{A}x_t,\ c_t = \boldsymbol{A}^T r_t,\ t = t + 1.$ |
| 4 | If $\lVert r_t \rVert_2 \leq \epsilon$, break, otherwise, go to step 2. |

that it is possible to fix the wrong decisions made in early stages. It is proven that, as $\lambda$ approaches zero, the solution produced by the Homotopy algorithm rigorously converges to the solution to the BP problem in (3.1) [Da06]. Since Homotopy provides exactly the solution set of the LASSO problems (3.5) for all the values of $\lambda$, it also provides the solution set of the BPDN problem (3.2) for all the values of $\epsilon$ in the noisy case. Furthermore, the Homotopy algorithm also features the $k$-iteration-solution property under a certain general conditions [Da06]. The uniform recovery guarantee, OMP-like complexity, and the $k$-iteration-solution property render Homotopy a strong candidate for the hardware implementation for accuracy-driven applications.

### 3.1.3 Iterative Soft Tresholding (IST)

IST is another fast algorithm to search for the solution to the LASSO problem in (3.5) [Ba08]. The pseudo-code of the IST algorithm is described in Table 3.3. Given the constraint of $y = \boldsymbol{A}x$, an initial guess of $x$ can be approximated by $x_1 = \eta_{\lambda_1}(\boldsymbol{A}^T y)$ assuming $\boldsymbol{A}$ is a nearly-orthogonal matrix, where the operator $\eta_\lambda(\cdot)$ denotes a certain thresholding scheme that prunes the input vector by keeping the top-ranked elements and disregarding the rest.

[4] $\mu \in (0, 1]$.

An example of the thresholding scheme is given by

$$\eta_\lambda(v(i)) = \begin{cases} v\,(i) - \lambda, \text{ if } |v\,(i)| > \lambda, \\ 0, \text{ otherwise.} \end{cases} \tag{3.11}$$

Since, the estimation error $e_1 = x - x_1$ can be characterized as

$$e_1 \approx \boldsymbol{A}^T \boldsymbol{A} e_1 = \boldsymbol{A}^T \boldsymbol{A}\,(x - x_1) = \boldsymbol{A^T}\,(y - \boldsymbol{A}x_1)\,, \tag{3.12}$$

a more accurate second estimation can be made by removing $e_1$ as

$$x_2 = \eta_{\lambda_1}(x_1 + \boldsymbol{A}^T\,(y - \boldsymbol{A}x_1)). \tag{3.13}$$

Similarly, the estimation error can be reduced gradually by iteratively thresholding the new estimation given as

$$x_t = \eta_{\lambda_{t-1}}(x_{t-1} + \boldsymbol{A}^T\,(y - \boldsymbol{A}x_{t-1})), \tag{3.14}$$

where the value of $\lambda$ is also shrunk gradually by a multiplier $\mu \in (0, 1]$ given as

$$\lambda_t = \mu \cdot \lambda_{t-1}. \tag{3.15}$$

Although the IST algorithm is much faster than LP-based algorithms, but extensive simulations have shown that IST performs worse than LP-based algorithms when it comes to the sparsity measurement tradeoff [Ma10b]. Fortunately, a slightly tuned version of the IST algorithm that can perform as well as LP-based algorithms, called approximate message passing (AMP), has been developed by D. Donoho et al. [Da09]. Different form OMP and Homotopy, where a LS problem is involved in each iteration, IST algorithms only require basic vector operations throughout the whole algorithm with a simpler data-flow control scheme that involves fewer variables. Therefore, IST algorithms are of great interest to efficient hardware implementations.

## 3.2 Algorithm Benchmarking

Other than the computational complexity, the performance of approximation algorithms is also a critical consideration for practical applications. In this section, a benchmarking

study is performed to compare the computational complexity and the sparse signal recovery performance of the above-mentioned SA algorithms.

### 3.2.1 Experiment Setting

In the benchmarking study, the above-mentioned SA algorithms are tested for recovering ideally sparse signals from the noisy observations obtained by random Bernoulli matrices. Specifically, $k$-sparse signals $x \in \mathbb{S}_k^n$ are first generated based on i.i.d. Gaussian random variables $z_x \sim \mathcal{N}(0, \sigma_x^2)$. Note that the indices of non-zero elements in $x$ are also randomly selected. Random measurements are generated by sampling $x$ through random Bernoulli matrices $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ as $y' = \boldsymbol{A}x$. Then, the measurements $y'$ are contaminated by an additive white Gaussian noise (AWGN) $\beta$ as $y = y' + \beta$, where $\beta$ is generated based on i.i.d. Gaussian random variables $z_\beta \sim \mathcal{N}(0, 10^{-0.1 \cdot SNR}\sigma_x^2)$ for setting a fixed SNR target. Last, the original signals $x$ are recovered by the SA algorithms given $y$ and $\boldsymbol{A}$. For the entire experiment, a fixed setting of $n = 512$ is used. In order to observe the recovery performance with respect to different undersampling ratios $(m/n)$, $m/n$ is swept from 0.1 to 0.9 with a step size of 0.05. Additionally, to understand the algorithm performance under different cases of signal sparsity ratio $(k/n)$ and SNR, a $k/n = 0.03,\ 0.1$, and 0.2 is used in representing the high-, medium-, and low-sparsity case, respectively, and a target SNR of 20 and 100 dB is adopted for the low- and high-SNR case, respectively. For each problem setting $(n, m, k)$, 1000 trials of the recovery test are performed using each algorithm. The results of each problem setting are then averaged across all the trials.

The signal recovery tests are performed in MATLAB simulation running on a 2.4 GHz Intel Core i7-4700MQ CPU. In the experiment, the IPM algorithm used is the log-barrier solver from the $\ell_1$-Magic package [Ca06] (available at link). The OMP and Homotopy algorithms used are developed based upon the pseudo-code in Tables 3.1 and 3.2. The IST algorithm used is the IST solver from the SparseLab package [Da06] (available at link). The AMP algorithm used is the Generalized AMP (GAMP) solver from the GAMP package [Ran11] (available at link). For the best recovery results, the distribution models of $x$ and

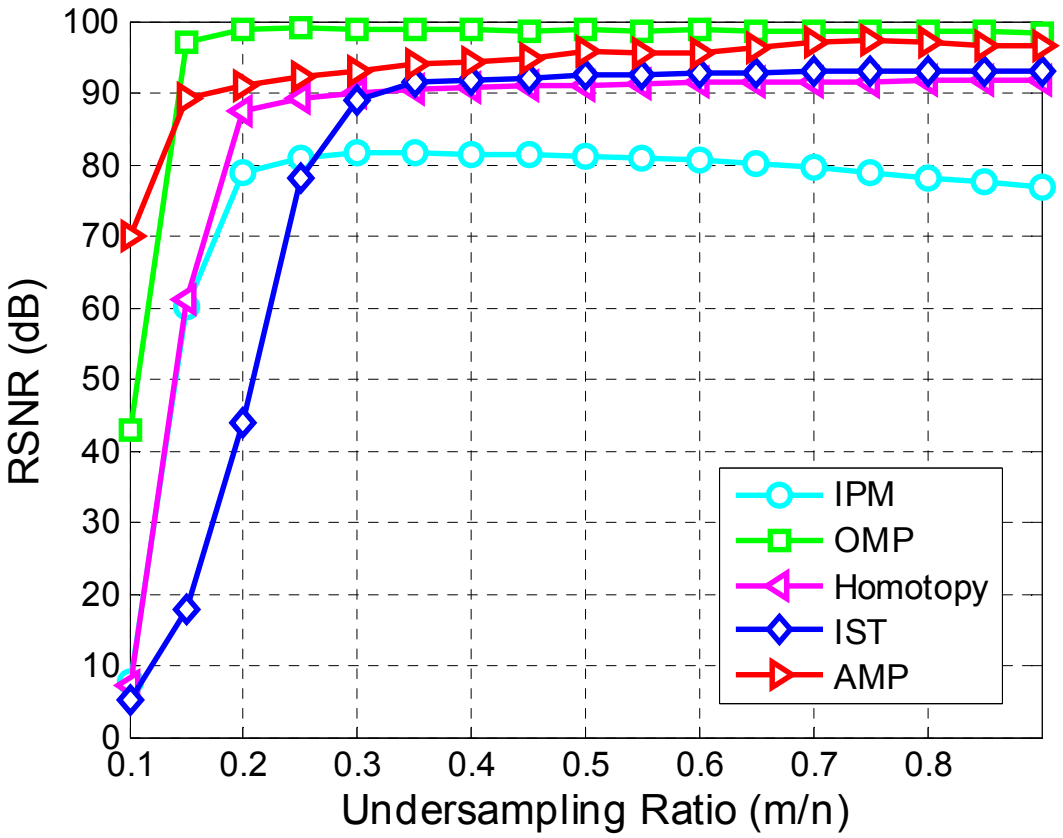



Figure 3.1: RSNR performance of the SA algorithms for a high signal sparsity ratio ($k/n = 0.03$) in the high-SNR case (SNR=100 dB).

$\beta$ are used as prior knowledge in the GAMP solver. For comparison purposes, the CPU execution time of each algorithm is recorded as a general indication of the computational complexity of the algorithm. The recovery performance is measured by reconstruction signal-to-noise ratio (RSNR) defined by

$$RSNR = 20 \cdot \log_{10}\left(\frac{\|x\|_2}{\|x - \hat{x}\|_2}\right), \tag{3.16}$$

where $x$ is the original signal, and $\hat{x}$ is the recovered estimation of $x$.

### 3.2.2 Benchmarking Results

The comparisons of the RSNR performance in the high-SNR case (SNR=100 dB) are shown in Figs. 3.1, 3.2, and 3.3. For a high signal sparsity ratio ($k/n = 0.03$), IMP, OMP,

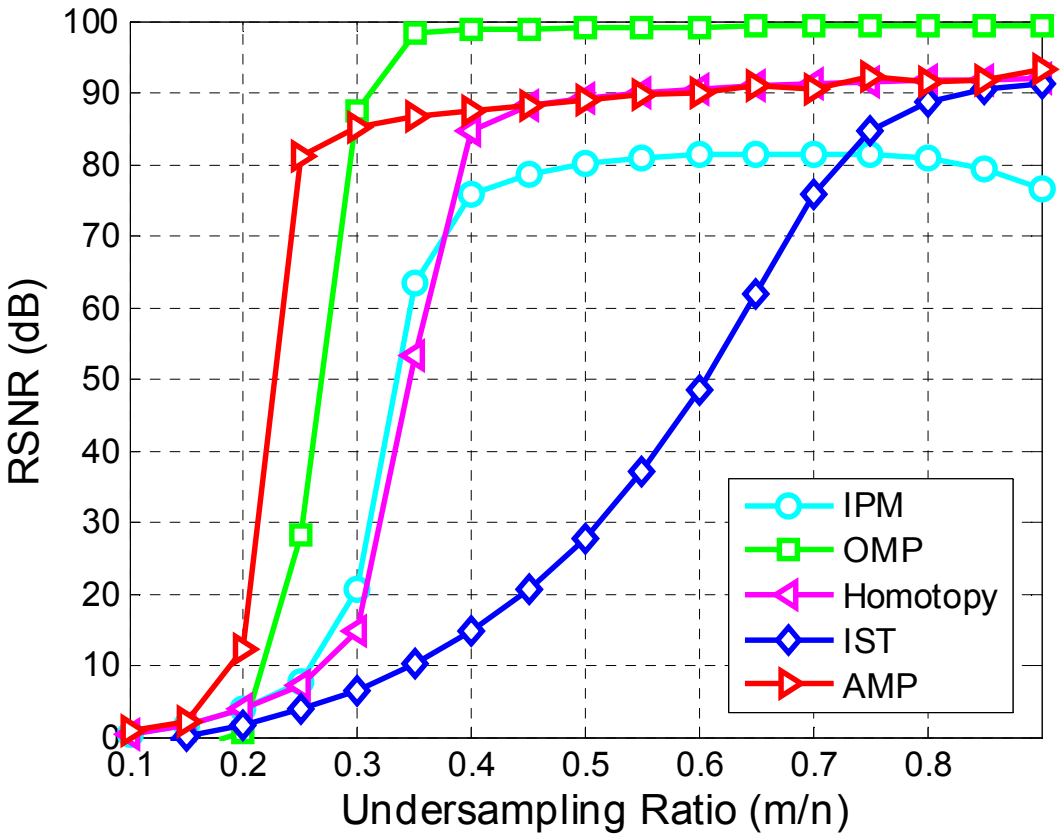



Figure 3.2: RSNR performance of the SA algorithms for a medium signal sparsity ratio ($k/n = 0.1$) in the high-SNR case (SNR=100 dB).

Homotopy, IST, and AMP is able to achieve a RSNR of over 78, 97, 87, 89, and 88 dB at the undersampling ratio of 0.2, 0.15, 0.2, 0.3, and 0.15, respectively. For a medium signal sparsity ratio ($k/n = 0.1$), IMP, OMP, Homotopy, IST, and AMP achieves a RSNR of over 75, 98, 84, 84, and 85 dB at the undersampling ratio of 0.4, 0.35, 0.4, 0.75, and 0.3, respectively. For a low signal sparsity ratio ($k/n = 0.2$), IMP, OMP, Homotopy, and AMP achieves a RSNR of over 80, 97, 87, 32, and 82 dB at the undersampling ratio of 0.6, 0.55, 0.65, 0.9, and 0.45, respectively.

Overall, OMP presents a perfect denoising capability by constantly showing a RSNR close to 100 dB. AMP shows the best undersampling capability with OMP being a very close second. Homotopy also shows a constantly high RSNR performance at the cost of a higher undersampling ratio. In comparison, IST has the worst undersampling capability,

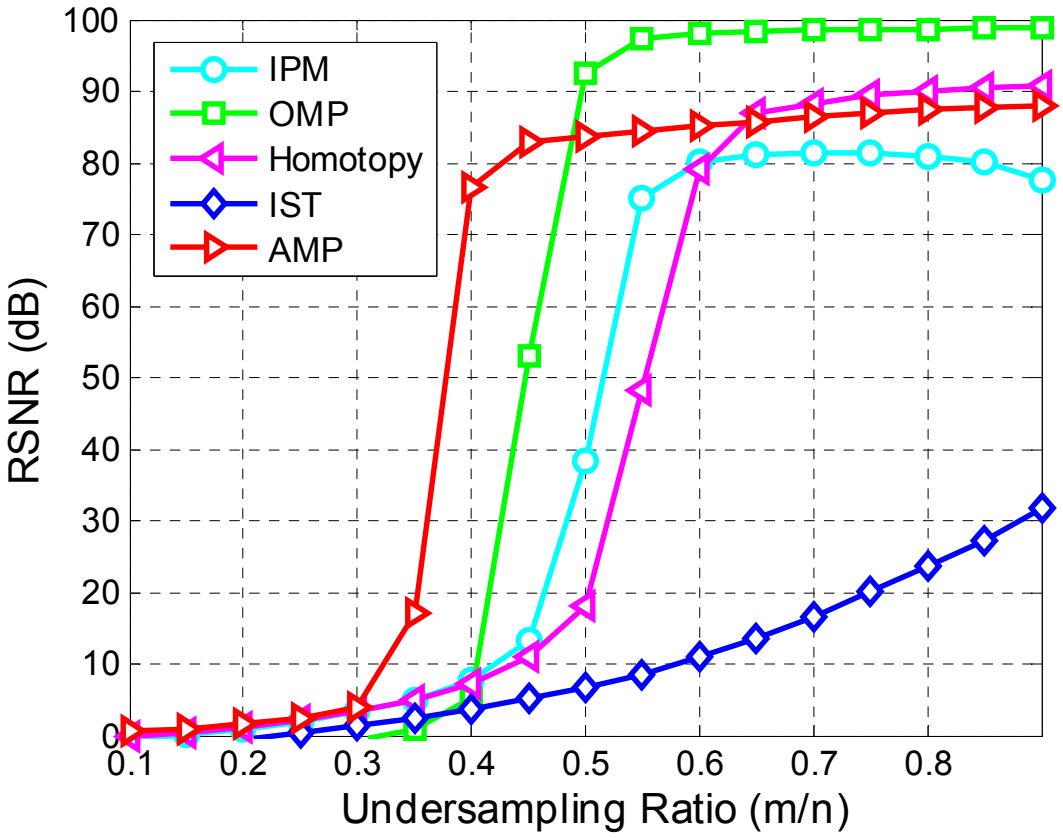



Figure 3.3: RSNR performance of the SA algorithms for a low signal sparsity ratio ($k/n = 0.2$) in the high-SNR case (SNR=100 dB).

which results in a low RSNR performance for recovering low-sparsity signals.

The comparisons of RSNR performance in the low-SNR case (SNR=20 dB) are shown in Figs. 3.4, 3.5, and 3.6. In this case, IMP fails the recovery test in most trials. In addition, higher undersampling ratio will always improve the SNR performance for the other algorithms except for recovering low-sparsity signals. For a high signal sparsity ratio ($k/n = 0.03$), OMP, Homotopy, IST, and AMP is able to achieve a RSNR of over 10 dB at the undersampling ratio of 0.15, 0.25, 0.2, and 0.15, respectively. For a medium signal sparsity ratio ($k/n = 0.1$), OMP, Homotopy, IST, and AMP achieves a RSNR of over 10 dB at the undersampling ratio of 0.35, 0.45, 0.45, and 0.3, respectively. For a low signal sparsity ratio ($k/n = 0.2$), OMP, Homotopy, and AMP achieves a RSNR of over 10 dB at the undersampling ratio of 0.55, 0.6, 0.75, and 0.45, respectively. Overall, AMP still shows the

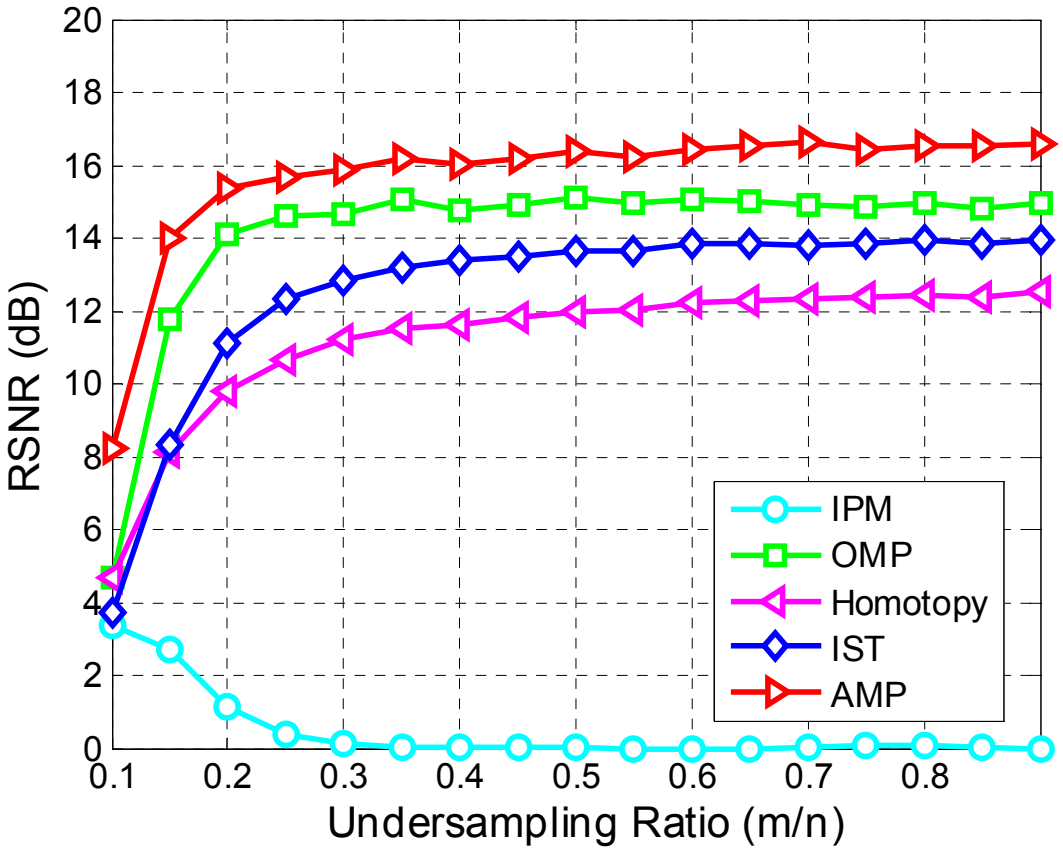



Figure 3.4: RSNR performance of the SA algorithms for a high signal sparsity ratio ($k/n = 0.03$) in the high-SNR case (SNR=20 dB).

best undersampling capability. Different from the high-SNR case, none of the algorithms shows a perfect denoising capability (close to 20 dB RSNR). In comparison, AMP has a slightly better RSNR performance than the other algorithms.

In the context of CS, a successful recovery with a lower undersampling ratio generally indicates that the data acquisition can be performed at a lower sampling rate with less energy consumption. Therefore, a good undersampling capability of the SA algorithm is especially attractive from the application perspective. In addition, for efficient hardware implementations, a low computational complexity of the SA algorithm is also preferred. Figure 3.7 compares the SA algorithms on the plane of undersampling capability versus computational complexity in different signal SNR and sparsity ratio cases. The undersampling

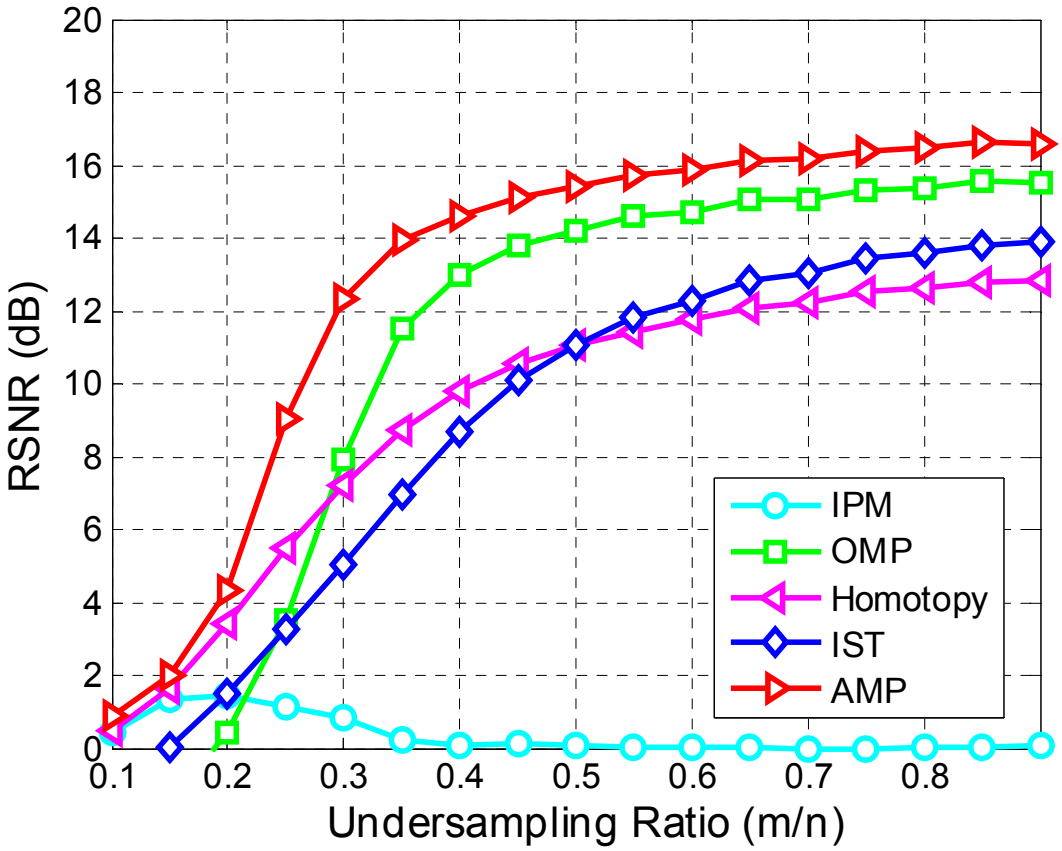



Figure 3.5: RSNR performance of the SA algorithms for a medium signal sparsity ratio ($k/n = 0.1$) in the high-SNR case (SNR=20 dB).

capability is measured as the lowest $m/n$ ratio required for achieving the best or a target RSNR for each algorithm. As previously mentioned in Section 3.2.1, the computational complexity is generally indicated by the execution time of each algorithm on the CPU. For comparison purposes, all the numbers are normalized to those of the IPM algorithm.

The benchmarking results show that OMP is the preferred algorithm for recovering high- and medium-sparsity signals due to 1) the high RSNR performance, 2) the low computational complexity, and 3) the good undersampling capability. For recovering low-sparsity signals, AMP is the preferred algorithm because of 1) the high RSNR performance especially under a low SNR condition, 2) the lower computational complexity in this case, and 3) the best undersampling capability overall.

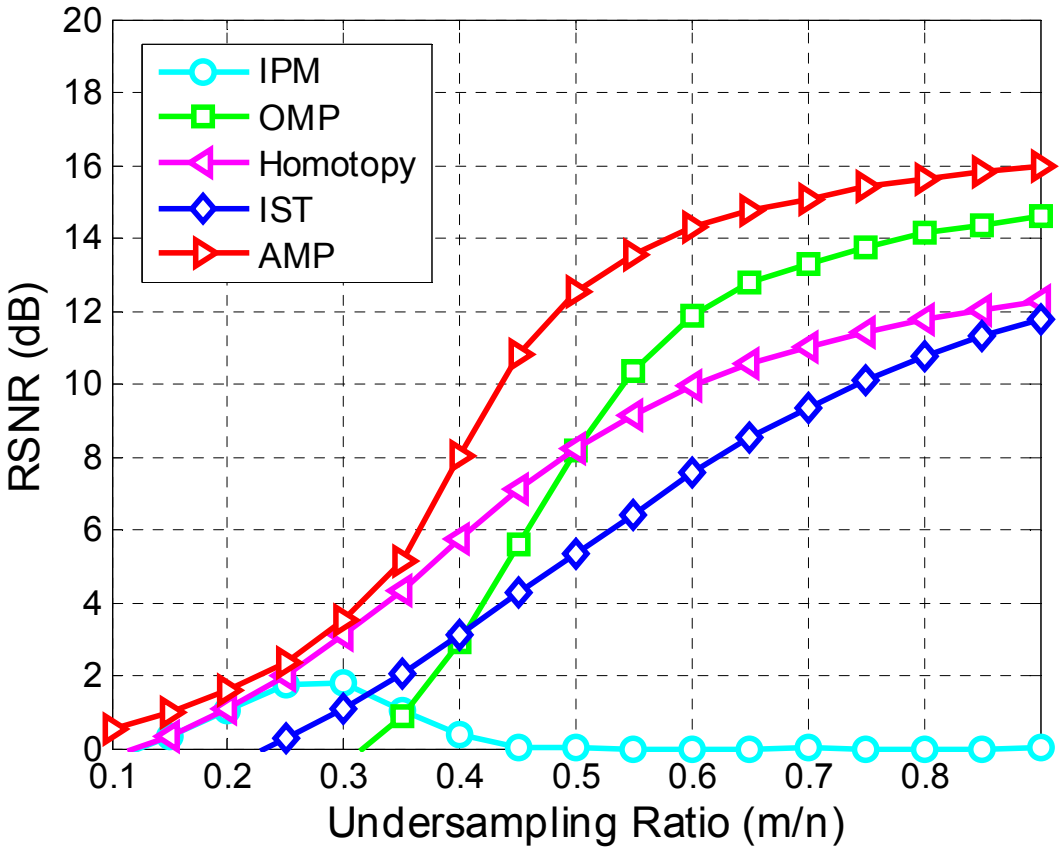



Figure 3.6: RSNR performance of the SA algorithms for a low signal sparsity ratio ($k/n = 0.2$) in the high-SNR case (SNR=20 dB).

In this work, the OMP algorithm is chosen for the hardware implementation for two main reasons. First, the main target application of this work deals with bio-medical signals that generally have a high or medium signal sparsity (see Section 1.2 for discussions), where OMP shows the best overall performance. Second, the OMP algorithm does not depend on the tuning of step-size variables or the prior knowledge about the signal's distribution model for achieving the best RSNR performance, making it very convenient to use in our application scenarios.

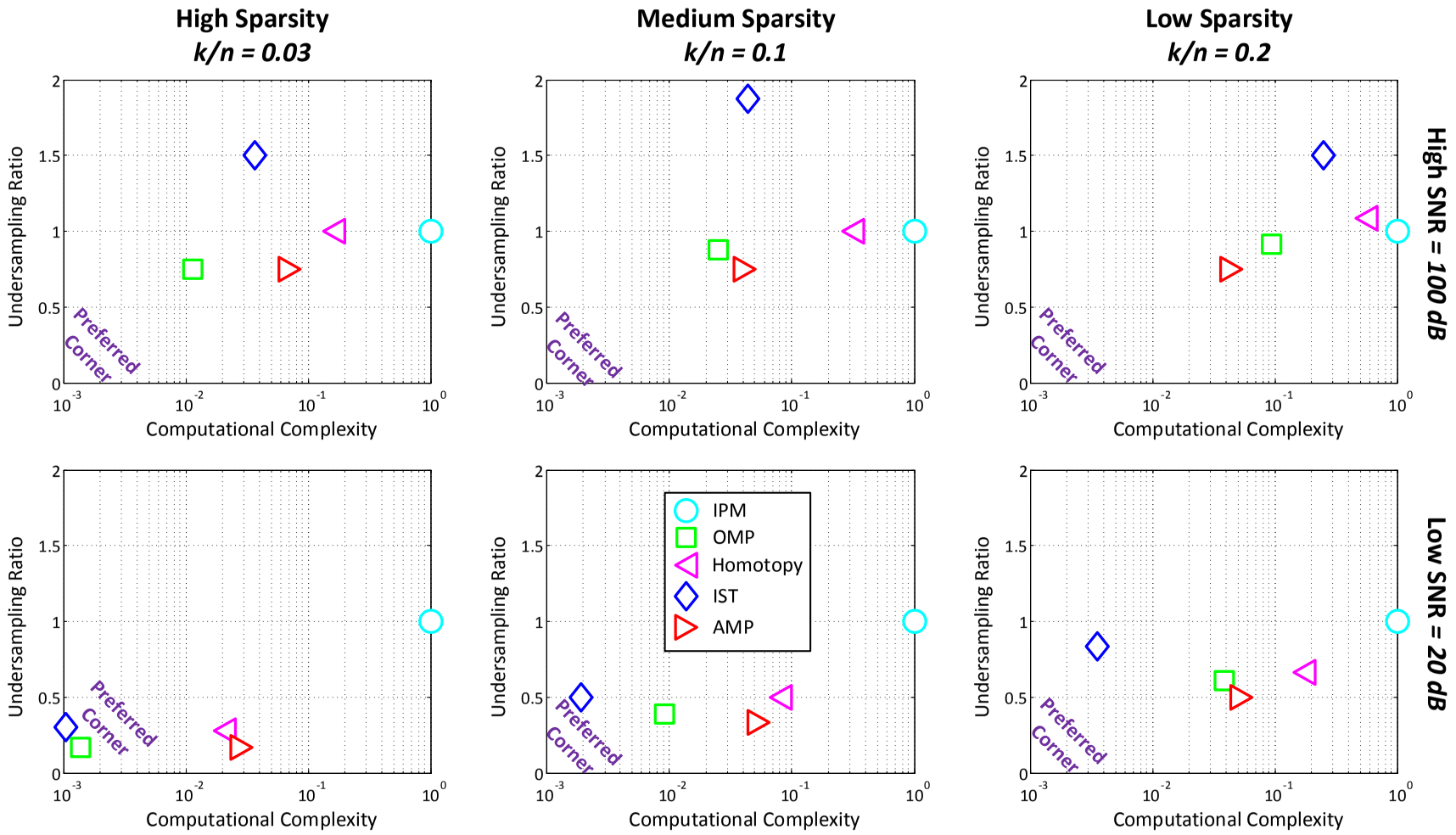



Figure 3.7: Comparison of the SA algorithms on the plane of undersampling ratio versus computational complexity (Norm. to IPM).

# CHAPTER 4

# Algorithm Design

For achieving the best efficiency-flexibility trade-off of the VLSI implementation of a complex algorithm, such as OMP, algorithm and architecture must be co-designed in an intertwined way rather than either domain being optimized separately. In this chapter, we present the algorithm design of the reformulated OMP. Specifically, the complexity characteristic of the original OMP algorithm is first analyzed. Then, three algorithm reformulation techniques are incorporated to (1) reduce the computational complexity of the LS task from $\mathcal{O}(mk^3)$ to $\mathcal{O}(mk^2)$ and (2) break down and simplify the LS task into 4 basic linear algebra (BLA) operations per iteration. Additionally, a hierarchical atom searching method is proposed to greatly reduce the computational complexity of the atom searching (AS) task.

## 4.1 Complexity Analysis of OMP

There are three main tasks performed in each iteration of OMP: the AS task for updating the active set (Step 2 in Table 3.1), the LS task for computing the updating direction (Step 3 in Table 3.1), and the estimation update (EU) task for updating the estimation along that direction (Step 4 in Table 3.1). Table 4.1 summarizes the number of floating-point operations (FLOPs) involved in each task at the iteration $t$, where $t = \{1, 2, \cdots, k\}$ for $x \in \mathbb{S}_k^n$. Note that the FLOP count is calculated assuming that Cholesky factorization is used for solving the LS problem in (3.3) [Van13]. As shown in Table 4.1, the AS task involves $2nm$ FLOPs in each iteration. Note that $n > m \gg k$ in the context of CS. Therefore, the AS task constantly contributes a significant portion of the computations. On the other hand, the

[1]Only the dominant factors are shown.
[2]Assuming Cholesky Factorization is used to solve the LS problem.

Table 4.1: Computations of OMP at Iteration $t$

| Task | FLOPs [1] |
|---|---|
| AS | $2nm$ |
| LS[2] | $mt^2 + t^3/3$ |
| EU | $2mt$ |

FLOPs involved in the LS task increase quadratically with $t$ as $mt^2 + t^3/3$. Consequently, the computations in the LS task become dominant when $t > \sqrt{2n}$. The computations in the EU task only increase linearly with $t$ as $2mt$. Overall, the total computational complexity of OMP (over $k$ iterations) is given by $2mnk + mk^2 + mk^3/3 + k^4/12$.

An example of the computation breakdown of OMP with a problem setting of $n = 500, m = 175, k = 50$ is illustrated in Fig. 4.1. In this example, the computations in the LS task grow dramatically as iteration $t$ goes up and surpass that of the AS task when $t \geq 32$. In terms of total FLOPs, the AS task is the main contributor, which is almost on par with the LS task, and the EU task is negligible. Specifically, the AS, LS, and EU task contributes to 51%, 46%, and 3% of the total computation in this example.

Computational complexity indicates the total computation workload, but not necessarily the actual hardware complexity. As VLSI design also includes memory and control units, the complexity of VLSI implementations also depends on the diversity of operation and the complexity of data flow control and scheduling. In our analysis, these factors are qualitatively referred to as operational complexity. The main operation in the AS and EU tasks is simply inner product, which has low operational complexity. On the contrary, the LS task requires complex operations such as matrix factorization, which involves a variety of basic operations in series with a complicated data flow pattern and a high data dependency, thereby having high operational complexity.

Figure 4.2 illustrates the overall complexity characteristic of the OMP algorithm. Note

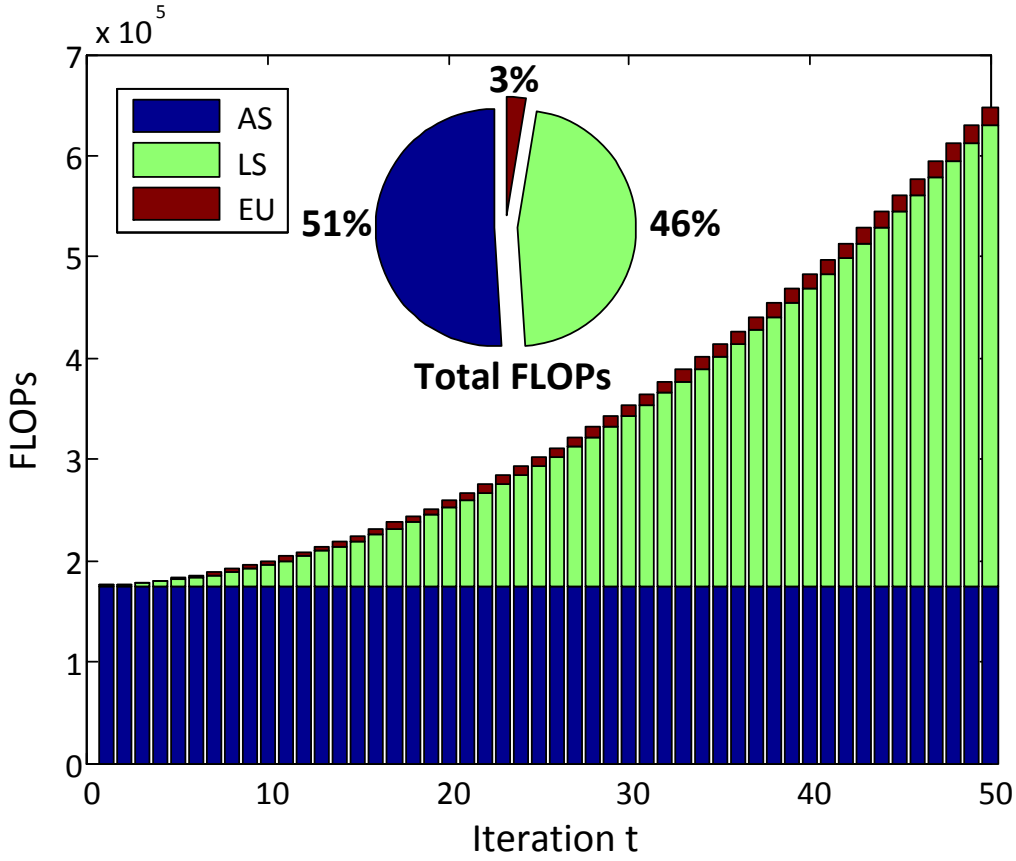



Figure 4.1: Illustration of the computation breakdown of OMP ($n = 500, m = 175, k = 50$).

that the AS and LS tasks both have high computational complexity. In addition, the LS task also features high operational complexity, creating a great challenge for the hardware design. High computational complexity indicates a large computation load for the processing unit and potentially a low system throughput. High operational complexity implies 1) large memory space and complicated memory control scheme are needed, 2) specialized processing units are needed, and 3) hardware resource sharing is difficult. In order to achieve an efficient VLSI implementation, actions must taken on the algorithm level to further simplify the complexity characteristic of OMP before going into the architecture design.

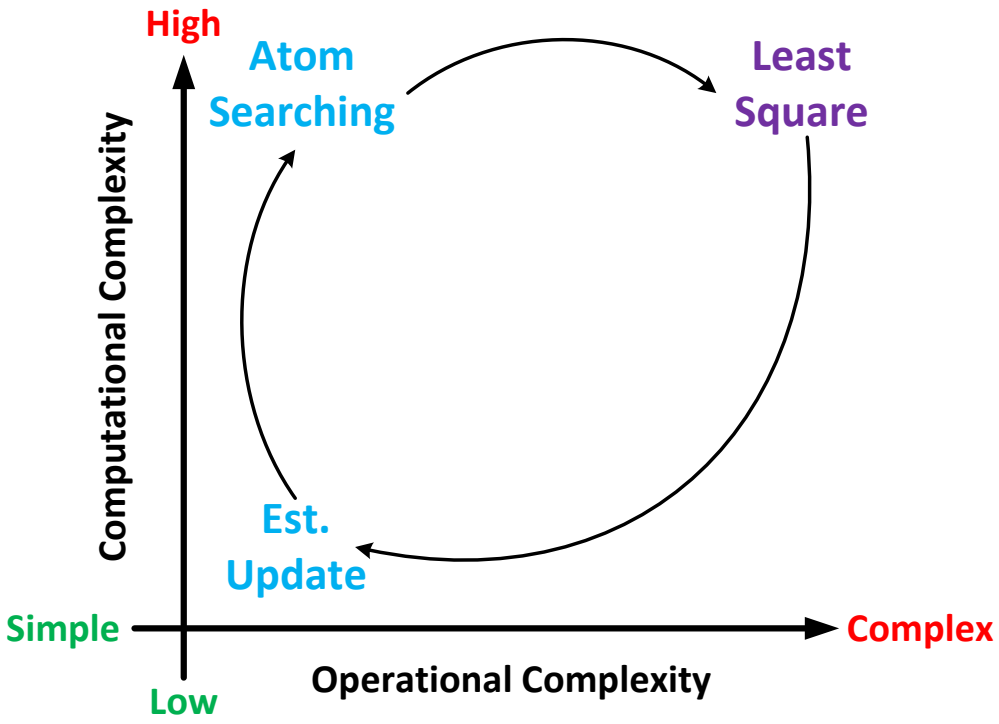



Figure 4.2: Complexity characteristic of OMP.

## 4.2 Algorithm Reformulation

### 4.2.1 Square-Root-Free OMP

There are two steps in the OMP algorithm involving square-root operations. Specifically, at iteration $t$, $t$ square-root operations are required for computing the Cholesky factorization matrices in the LS task, and a single square-root operation is involved in computing the stopping criterion (Step 5 in Table 3.1). An explicit implementation of such non-linear operation is not cost-effective, as it cannot be reused by the other tasks and will have a very low utilization rate. Therefore, we choose to eliminate the square-root operations in the reformulated algorithm.

The solution to the LS problem in (3.3) can be computed by solving the equivalent normal equation given by

$$\boldsymbol{\Phi} x(\Lambda) = \boldsymbol{A}_{\Lambda}^{T} y, \tag{4.1}$$

where $\boldsymbol{\Phi} \in \mathbb{R}^{k\times k} = \boldsymbol{A}_{\boldsymbol{\Lambda}}^T \boldsymbol{A}_{\boldsymbol{\Lambda}}$ is a positive-definite matrix. By using Cholesky factorization, $\boldsymbol{\Phi}$ can be decomposed as

$$\boldsymbol{\Phi}=\boldsymbol{L'}\boldsymbol{L'}^T, \tag{4.2}$$

where $\boldsymbol{L'} \in \mathbb{R}^{k\times k}$ is a lower-triangular matrix. Note that square-root operations are involved in computing the diagonal elements of $\boldsymbol{L'}$ in the conventional Cholesky factorization method [Van13]. To avoid this, we adopt an alternative Cholesky factorization method [Yan11], which essentially eliminates square-root operations by taking out the square-rooted factors from both $\boldsymbol{L'}$ and $\boldsymbol{L'}^T$ as

$$\boldsymbol{\Phi}=(\boldsymbol{L'}\boldsymbol{D'}^{\boldsymbol{-1}})(\boldsymbol{D'}\boldsymbol{D'})(\boldsymbol{D'}^{\boldsymbol{-1}}\boldsymbol{L'}^T)=\boldsymbol{L}\boldsymbol{D}\boldsymbol{L}^T, \tag{4.3}$$

where $\boldsymbol{D'} \in \mathbb{R}^{k\times k}$ is a diagonal matrix that contains all the square-rooted factors and satisfies $diag(\boldsymbol{D'}) = diag(\boldsymbol{L'})$, $\boldsymbol{L} \in \mathbb{R}^{k\times k}$ is a lower-triangular matrix whose diagonal elements are all ones with $\boldsymbol{L}=\boldsymbol{L'}\boldsymbol{D'}^{\boldsymbol{-1}}$, and $\boldsymbol{D} \in \mathbb{R}^{k\times k}$ is a diagonal matrix that is free of square roots with $\boldsymbol{D}=\boldsymbol{D'}^{\boldsymbol{2}}$. By substituting $\boldsymbol{\Phi}$ with (4.3), the normal equation in (4.1) becomes

$$\boldsymbol{L}\boldsymbol{D}\boldsymbol{L}^T x(\Lambda) = \boldsymbol{A}_{\boldsymbol{\Lambda}}^T y. \tag{4.4}$$

Equation (4.4) can be solved through a series of vector operations. First, matrix-vector multiplications are required to compute

$$u = \boldsymbol{A}_{\Lambda}{}^T y. \tag{4.5}$$

Then, forward substitution (FS) is needed for solving $v$ from

$$\boldsymbol{L}v = u. \tag{4.6}$$

As $\boldsymbol{D}$ is a diagonal matrix, divisions are involved in computing

$$w = \boldsymbol{D}^{\boldsymbol{-1}} v. \tag{4.7}$$

Lastly, backward substitution (BS) is required for computing $x(\Lambda)$ from

$$\boldsymbol{L}^T x(\Lambda) = w. \tag{4.8}$$

The stopping criterion $\|r_t\|_2 \leq \epsilon$ can be reformulated as ${r_t}^T r_t \leq \epsilon^2$. Overall, the OMP algorithm is free of square-root operations after the reformulation.

### 4.2.2 Incremental Cholesky Factorization

In each iteration of OMP, only a single atom is added to the active set. More specifically, at iteration $t$, we have

$$\Lambda_t = \Lambda_{t-1} \cup \varphi, \tag{4.9}$$

and

$$\boldsymbol{A}_{\boldsymbol{\Lambda_t}} = [\boldsymbol{A}_{\boldsymbol{\Lambda_{t-1}}}\ \boldsymbol{a}_{\boldsymbol{\varphi}}], \tag{4.10}$$

where $\varphi$ is the index of the new active atom. According to (4.10), the square matrix $\boldsymbol{\Phi}_t \in \mathbb{R}^{t\times t}$ in (4.1) has $\boldsymbol{\Phi}_t = \boldsymbol{A}_{\boldsymbol{\Lambda_t}}{}^T \boldsymbol{A}_{\boldsymbol{\Lambda_t}}$ and can be partitioned as

$$\boldsymbol{\Phi}_t = \begin{bmatrix} \boldsymbol{\Phi}_{t-1} & \boldsymbol{A}_{\Lambda_{t-1}}^T \boldsymbol{a}_{\boldsymbol{\varphi}} \\ \boldsymbol{a}_{\boldsymbol{\varphi}}{}^T \boldsymbol{A}_{\boldsymbol{\Lambda_{t-1}}} & \boldsymbol{a}_{\boldsymbol{\varphi}}{}^T \boldsymbol{a}_{\boldsymbol{\varphi}} \end{bmatrix}, \tag{4.11}$$

where $\boldsymbol{\Phi}_{t-1} \in \mathbb{R}^{t-1\times t-1} = \boldsymbol{A}_{\boldsymbol{\Lambda_{t-1}}}{}^T \boldsymbol{A}_{\boldsymbol{\Lambda_{t-1}}}$ is from iteration $t-1$, $\boldsymbol{A}_{\Lambda_{t-1}}^T \boldsymbol{a}_{\boldsymbol{\varphi}} \in \mathbb{R}^{t-1}$ is a column vector, and $\boldsymbol{a}_{\boldsymbol{\varphi}}{}^T \boldsymbol{a}_{\boldsymbol{\varphi}}$ is a scalar. Note that (4.11) indicates that at iteration $t$, the $\boldsymbol{\Phi}_t$ in the normal equation can be constructed from $\boldsymbol{\Phi}_{t-1}$ by adding both a new row and column. In correspondence to (4.11), the Cholesky factorization matrices in (4.3) must hold the same property, which leads to

$$\boldsymbol{L}_t \boldsymbol{D}_t \boldsymbol{L}_t{}^T = \begin{bmatrix} \boldsymbol{L}_{t-1} & \vec{0} \\ l_{21}{}^T & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{D}_{t-1} & \vec{0} \\ \vec{0}^T & d_{22} \end{bmatrix} \begin{bmatrix} \boldsymbol{L}_{t-1} & \vec{0} \\ l_{21}{}^T & 1 \end{bmatrix}^T, \tag{4.12}$$

where $l_{21} \in \mathbb{R}^{t-1}$ is a column vector and $d_{22}$ is a scalar. By expanding the equation $\boldsymbol{\Phi}_t = \boldsymbol{L}_t \boldsymbol{D}_t \boldsymbol{L}_t{}^T$ with (4.11) and (4.12), we can derive

$$\boldsymbol{L}_{t-1} \boldsymbol{D}_{t-1} l_{21} = \boldsymbol{A}_{\Lambda_{t-1}}^T \boldsymbol{a}_{\boldsymbol{\varphi}}, \tag{4.13}$$

and

$$d_{22} = \boldsymbol{a}_{\boldsymbol{\varphi}}{}^T \boldsymbol{a}_{\boldsymbol{\varphi}} - l_{21}{}^T \boldsymbol{D}_{t-1} l_{21}, \tag{4.14}$$

respectively. Note that $l_{21}$ can be computed using the same method as (4.4) through a series of matrix-vector multiplication, FS, and divisions. Equation (4.11) and (4.12) imply that the Cholesky factorization in OMP can be computed in an incremental fashion: in iteration $t$, the new elements of the factorization matrices (associated with the newly added active atom)

can be computed based upon the factorization matrices from iteration $t-1$. Therefore, in the reformulated OMP, the Cholesky factorization matrices are updated incrementally by computing (4.13) and (4.14) only in each iteration.

### 4.2.3 Incremental Estimation Update

At iteration $t$ of the original OMP algorithm, a new estimation $x_t$ is made by solving (4.1), and the residual $r_t$ is updated by

$$r_t = y - \boldsymbol{A}_{\boldsymbol{\Lambda_t}} x_t(\Lambda_t). \tag{4.15}$$

According to (4.1) and (4.15), we can derive that

$$\boldsymbol{A}_{\boldsymbol{\Lambda_t}}{}^T r_t = \boldsymbol{A}_{\boldsymbol{\Lambda_t}}{}^T y - \boldsymbol{A}_{\boldsymbol{\Lambda_t}}{}^T \boldsymbol{A}_{\boldsymbol{\Lambda_t}} x_t(\Lambda_t) = \vec{0}. \tag{4.16}$$

Equation (4.16) indicates that the updated residual $r_t$ is always orthogonal to the current active atoms $\boldsymbol{A}_{\boldsymbol{\Lambda_t}}$ in OMP. In our design, we take into account this special property as prior knowledge to further simplify the LS task.

By substituting $y$ in (4.1) with $r_{t-1} + \boldsymbol{A}_{\boldsymbol{\Lambda_{t-1}}} x_{t-1}(\Lambda_{t-1})$ according to (4.15) describing iteration $t-1$, we can deliver the equivalent (4.1) at iteration $t$ as

$$\boldsymbol{\Phi_t} x(\Lambda_t) = \boldsymbol{A}_{\boldsymbol{\Lambda_t}}^T r_{t-1} + \boldsymbol{A}_{\boldsymbol{\Lambda_t}}^T \boldsymbol{A}_{\boldsymbol{\Lambda_{t-1}}} x_{t-1}(\Lambda_{t-1}). \tag{4.17}$$

From the superposition property of linear systems, we know that $x(\Lambda_t)$ in (4.17) is the superposition of the solution $x_1(\Lambda_t)$ and $x_2(\Lambda_t)$ to the separate equation

$$\boldsymbol{\Phi_t} x_1(\Lambda_t) = \boldsymbol{A}_{\boldsymbol{\Lambda_t}}^T r_{t-1}, \tag{4.18}$$

and

$$\boldsymbol{\Phi_t} x_2(\Lambda_t) = \boldsymbol{A}_{\boldsymbol{\Lambda_t}}^T \boldsymbol{A}_{\boldsymbol{\Lambda_{t-1}}} x_{t-1}(\Lambda_{t-1}), \tag{4.19}$$

respectively. Note that the right hand side (RHS) of (4.18) is part of the correlation coefficient $c(\Lambda_t)$ that is already computed in the AS task. Additionally, the left hand side (LHS) of (4.19) is equivalent to the RHS. Therefore, the solution $x_2(\Lambda_t)$ to (4.19) is trivial

and can be derived as

$$x_2(\Lambda_t) = \begin{bmatrix} x(\Lambda_{t-1}) \\ 0 \end{bmatrix}, \tag{4.20}$$

Based upon the above observations, we derive an incremental estimation update method as the following two steps. First, the updating direction $d$ can be computed by solving the new normal equation

$$\boldsymbol{\Phi_t} d(\Lambda_t) = c(\Lambda_t), \tag{4.21}$$

where $c(\Lambda_t) \in \mathbb{R}^t = \boldsymbol{A_{\Lambda_t}}^T r_{t-1}$. Second, $r_t$ and $x_t$ can be updated based upon $d$ and their previous value $r_{t-1}$ and $x_{t-1}$ as

$$r_t = r_{t-1} - \boldsymbol{A_{\Lambda_t}} d(\Lambda_t), \tag{4.22}$$

and

$$x_t = x_{t-1} + d, \tag{4.23}$$

respectively. Note that from (4.10) and (4.16) describing iteration $t-1$, we can derive that

$$c(\Lambda_t) = \begin{bmatrix} \boldsymbol{A_{\Lambda_{t-1}}}^T r_{t-1} \\ \boldsymbol{a_\varphi} r_{t-1} \end{bmatrix} = \begin{bmatrix} \vec{0} \\ \boldsymbol{a_\varphi} r_{t-1} \end{bmatrix}. \tag{4.24}$$

Equation (4.24) indicates that $c(\Lambda_t)$ must be a 1-sparse vector that has only one non-zero element. By utilizing this prior knowledge, the computation step in solving (4.21) can be greatly simplified. Specifically, the matrix-vector multiplications in (4.5) and the subsequent FS and divisions in (4.6) and (4.7) can be completely bypassed.

With all of the introduced reformulation techniques applied, the pseudo-code of the reformulated OMP algorithm is described in Table 4.2.

### 4.2.4 Complexity Reduction

Table 4.3 summarizes the FLOP count of the reformulated OMP at iteration $t$. Overall, the total computational complexity of the reformulated OMP (over $k$ iterations) gets reduced

[3]Assuming all the atoms in $\boldsymbol{A}$ are normalized.
[4]Step 6b is a memory operation.
[5]Only the dominant factors are shown.

Table 4.2: Pseudo-Code of the Reformulated OMP Algorithm

| Task | Step | Operation |
|---|---|---|
| | 1 | **Initialize**: $r_0 = y,\ x_0 = \vec{0},\ d = \vec{0},\ \Lambda_0 = \varnothing,\ t = 1.$ |
| | 2 | **While** $\|r_{t-1}\|_2^2 \leq \epsilon^2$ or $t \leq t_{max}$, **do** |
| AS | 3[3] | $c = \boldsymbol{A}^T r_{t-1},\ \varphi = \arg\max_i \|c(i)\|,\ \Lambda_t = \Lambda_{t-1} \cup \varphi.$ |
| | 4 | $h = \boldsymbol{A}_{\boldsymbol{\Lambda_t}}^T a_\varphi$ |
| | 5 | $\boldsymbol{L_{t-1}} w = h(1:t-1),\ \ l_{21} = w./diag(\boldsymbol{D_{t-1}})$ |
| | 6a | $d_{22} = h\,(t) -\ {l_{21}}^T w$ |
| LS | 6b[4] | $\boldsymbol{L}_t = \begin{bmatrix} \boldsymbol{L_{t-1}} & \vec{0} \\ {l_{21}}^T & 1 \end{bmatrix},\ \boldsymbol{D}_t = \begin{bmatrix} \boldsymbol{D_{t-1}} & \vec{0} \\ \vec{0}^T & d_{22} \end{bmatrix}$ |
| | | ($\boldsymbol{L_1}$=1, $\boldsymbol{D_1} = {a_\varphi}^T a_\varphi$) |
| | 7 | ${\boldsymbol{L}_t}^T d = \begin{bmatrix} \vec{0} \\ c(\varphi)/\ d_{22} \end{bmatrix}$ |
| ES | 8 | $x_t = x_{t-1} + d,\ r_t = r_{t-1} - \boldsymbol{A}_{\boldsymbol{\Lambda_t}} d,\ \ t = t+1.$ |
| | | **End while,** return $x_t$ |

to $2mnk + 2mk^2 + \frac{2}{3}k^3$. In comparison to Tables 3.1 and 4.1, the algorithm reformulation techniques applied not only reduce the computational complexity of the LS task (over $k$ iterations) from $\mathcal{O}(mk^3)$ to $\mathcal{O}(mk^2)$ but also break down the LS problem and simplify it into 4 BLA operations per iteration (Steps 4–7 in Table 4.2). Specifically, the reformulated LS task only involves a few inner products, a single FS for updating the Cholesky factorization matrices, and a single BS for computing the updating direction in each iteration.

For comparison purposes, the same example of the computation breakdown shown in Fig. 4.1 is re-plotted in Fig. 4.3 based upon the reformulated OMP algorithm (Table 4.2). By taking advantage of the algorithm reformulations, the total computation of the LS task gets reduced by almost 17× as compared to the original OMP algorithm in this example. As a result, the LS task now has much reduced computational complexity, that is as low as the

Table 4.3: Computations of the Reformulated OMP at Iteration $t$

| Task | FLOPs [5] |
|---|---|
| AS | $2nm$ |
| LS | $2mt + 2t^2$ |
| EU | $2mt$ |

EU task, and the AS task dominates the total computation. Specifically, the AS, LS, and EU task of the reformulated OMP contributes to 90%, 5%, and 5% of the total computation in this example.

Figure 4.4 illustrates the overall complexity characteristic of the reformulated OMP algorithm. The AS task contributes to over 90% of the total computation with simply inner products, presenting high computational but low operational complexity. Consequently, the AS part of the VLSI architecture design is the potential throughput bottleneck of the system. Sufficient level of parallelization must be applied. Different from the case in Fig. 4.2, the LS task in the reformulated OMP now has low computational and medium operational complexity. This characteristic indicates that the LS part potentially limits the efficiency of the VLSI design. This is because any specialized hardware unit designed for performing the LS task will have a very low utilization rate but contributing leakage power and silicon area. To improve both the area and energy efficiency, it is desired to share as much computing resource for the LS task with the other tasks as possible. Fortunately, the reduced complexity of the LS task resulting from the algorithm reformulation techniques greatly facilitates resource sharing in the architecture design.

## 4.3 Hierarchical AS

In VLSI design, quantization is an important factor that has great impact on almost every aspect of the hardware design. However, this factor is often overlooked at the algorithm

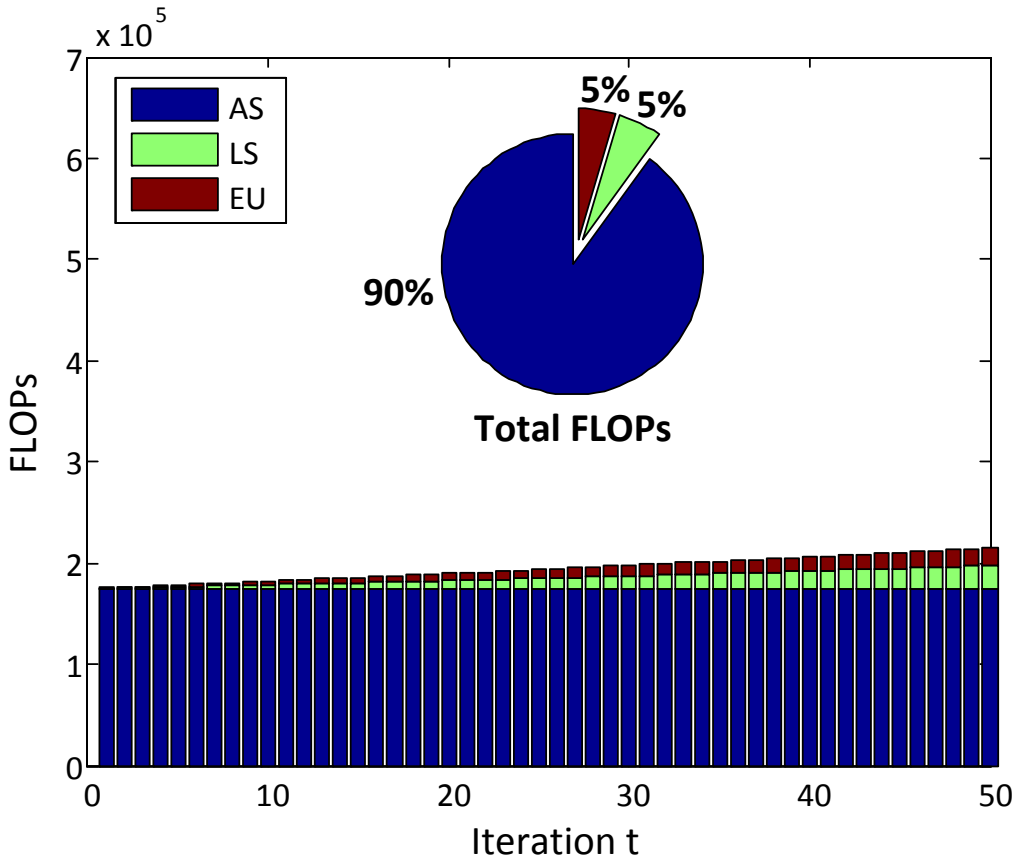



Figure 4.3: Illustration of the computation breakdown of the reformulated OMP ($n = 500, m = 175, k = 50$).

level. To be more specific, quantization noise may have different impact on different operations in the same algorithm. For instance, a comparison operation typically has much higher tolerance to quantization noise than addition or multiplication. This is because in a comparison operation, we only care about the relative magnitude rather than the exact value of the operands.

One should note that although the AS task of the reformulated OMP contributes to over 90% of the total computation, all the information needed from this task is simply which atom has the biggest correlation coefficient with the residue (see Step 3 in Table 4.2). In other words, every inner product computed in the AS task is followed by a magnitude comparison. Therefore, we can infer that the whole AS task is actually insensitive to quantization noise,

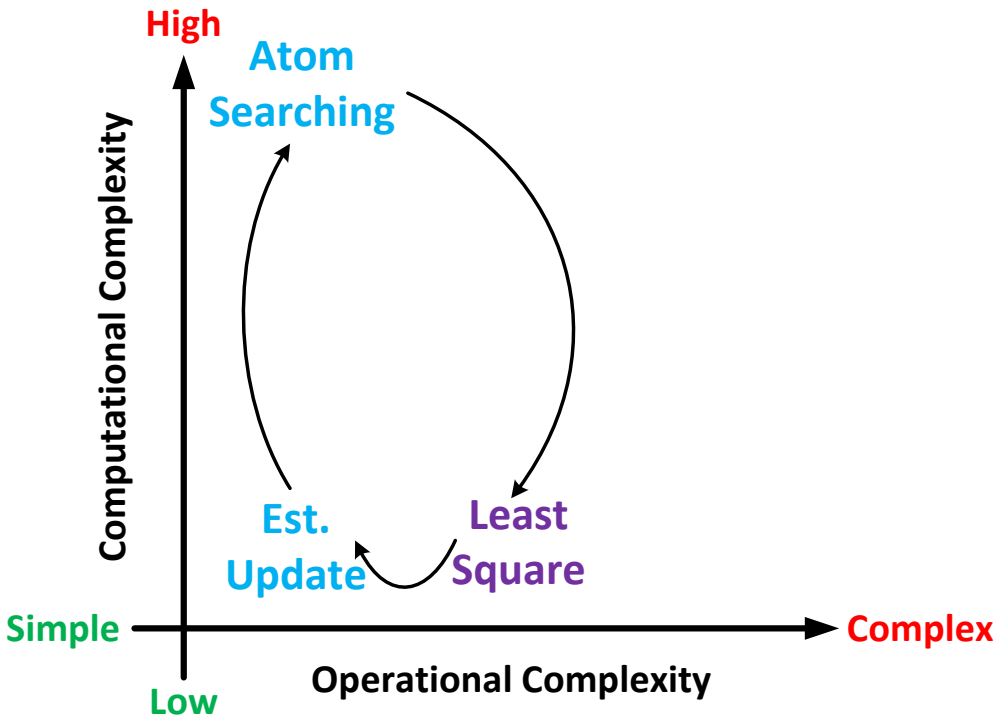



Figure 4.4: Complexity characteristic of the reformulated OMP.

and large level of quantization can be applied to further reduce the computational complexity of the AS task. The secret of quantization is that the magnitude information resides in the most significant bit (MSB) of numbers. As shown by the example of comparing two inner products in Fig. 4.5, by using only the MSB of each number to compute the inner product, we are still able to tell that the top inner product has a bigger magnitude than the bottom one, but of course, with slightly reduced confidence.

Based upon this observation, we propose a hierarchical AS method in two steps. The pseudo-code of the hierarchical AS method is shown in Table 4.4. First, a coarse-grain searching is performed by computing the full list of atoms in $\boldsymbol{A}$ with $w$ MSB only (assuming a fixed-point data format). The coarse-grain searching can effectively reduce the computational complexity of the AS task for an efficient VLSI implementation. In order to guarantee the accuracy, a second round of fine-grain searching is performed by computing a much shorter list of atoms, which is essentially the top ranked $j$ atoms from the coarse-grain searching,

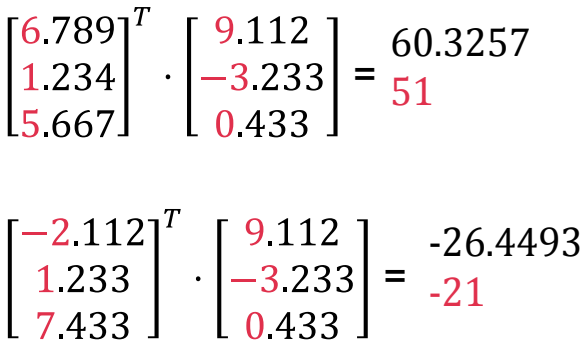



Figure 4.5: Illustration of comparing two inner products using MSB only.

Table 4.4: Pseudo-Code of the Hierarchical AS Method

| Step | Operation |
|---|---|
| 1 | Compute $c = \boldsymbol{A}^T r_{t-1}$ with $w$ MSB only,<br>$c' = sort(c)$, $\Omega = \{$the index set of the top $j$ elements in $c'$ $\}$ |
| 2 | Compute $c = \boldsymbol{A}(\boldsymbol{\Omega})^T r_{t-1}$ with full precision,<br>$\varphi = \arg\max_i \lvert c(i) \rvert$, $\Lambda_t = \Lambda_{t-1} \cup \varphi$. |

with full precision.

Note that the proposed method has two important user parameters. $w$ is the data word-length to use in the coarse-grain searching, and $j$ is the size of the fine-grain searching. For each signal recovery test, there exists a minimum size of $j$ that guarantees the correct searching result for a given value of $w$. Consequently, $j$ can be considered as a random variable for a given value of $w$ in practical applications. For the best VLSI implementation results, Monte Carlo simulations must be conducted by the designer in order to determine the optimal values of $w$ and $j$. Figure 4.6 shows an example of the design space of $w$ and $j$ extracted from the reconstruction test of compressively sampled ECG signals. Figure 4.6 is a box plot that presents the statistics of $j$ that guarantees the correct searching result

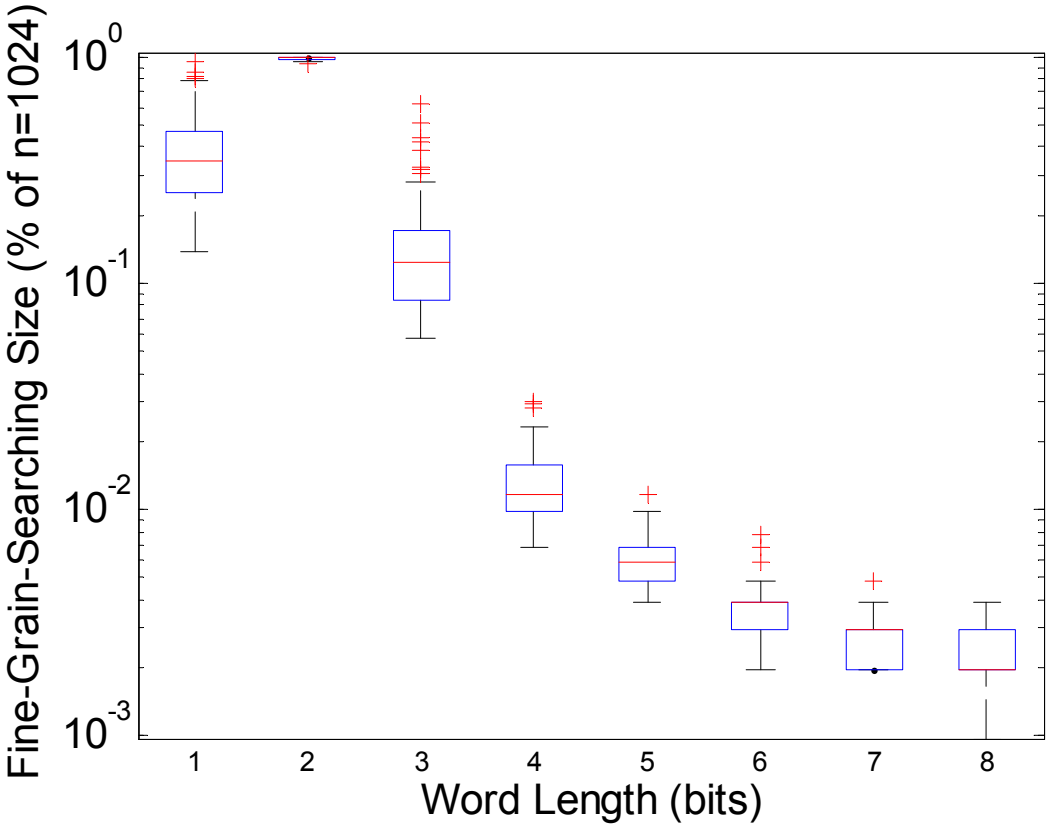



Figure 4.6: The design space of the hierarchical atom searching method extracted from the reconstruction test of compressively sampled ECG signals.

with respect to $w$ when a fixed-point data format is used. Overall, by using only 4 MSB (including 1 sign bit) in the coarse-grain searching, the proposed hierarchical AS method effectively reduces the size of the fine-grain searching to only $< 5\%$ of the total atoms without any degradation in accuracy.

The proposed hierarchical AS method presents two advantages for the VLSI implementation. First, one could improve the throughput of the design by reconfiguring the same arithmetic unit to take more data in parallel (each with MSB only). Alternatively, one could also improve the efficiency of the design by reducing the logic complexity.

# CHAPTER 5

# VLSI Architecture Design

In this chapter, we present a scalable VLSI architecture of a SA engine soft-IP core that can be implemented on reconfigurable logic devices, such as field-programmable gate arrays (FPGAs), or system-on-chips (SoCs) for performing real-time and energy-efficient SA. The soft-IP core supports a floating-point data format and 10 design parameters, providing the necessary flexibility for application-specific customization. Taking advantage of the algorithm-architecture co-design based upon the reformulated OMP algorithm, the proposed VLSI architecture features high parallelism, scalability, and configurability, in which all the computing resources are completely reused for performing the AS, LS, and EU tasks. As a result, the implementation of the SA engine achieves a 100% utilization of the computing resources and high area efficiency.

## 5.1 System Architecture

The system architecture of the SA engine is shown in Fig. 5.1. The computing resources in the system architecture include the vector and scalar processing cores (VC and SC). In order to support a flexible throughput with high energy efficiency, multiple processing elements (PEs) are coordinated in parallel through the interconnect block (IB) in the VC. Increasing the parallelism of PEs allows the designer to trade-off supply voltage in the circuit design to gain energy-efficiency while maintaining the real-time throughput. When the IB is enabled, PEs are configured to perform inner product. Otherwise, PEs can be configured to support element-wise addition, multiplication, multiply-accumulation (MAC), and vector-scalar product. SC supports scalar comparison, addition, accumulation, and

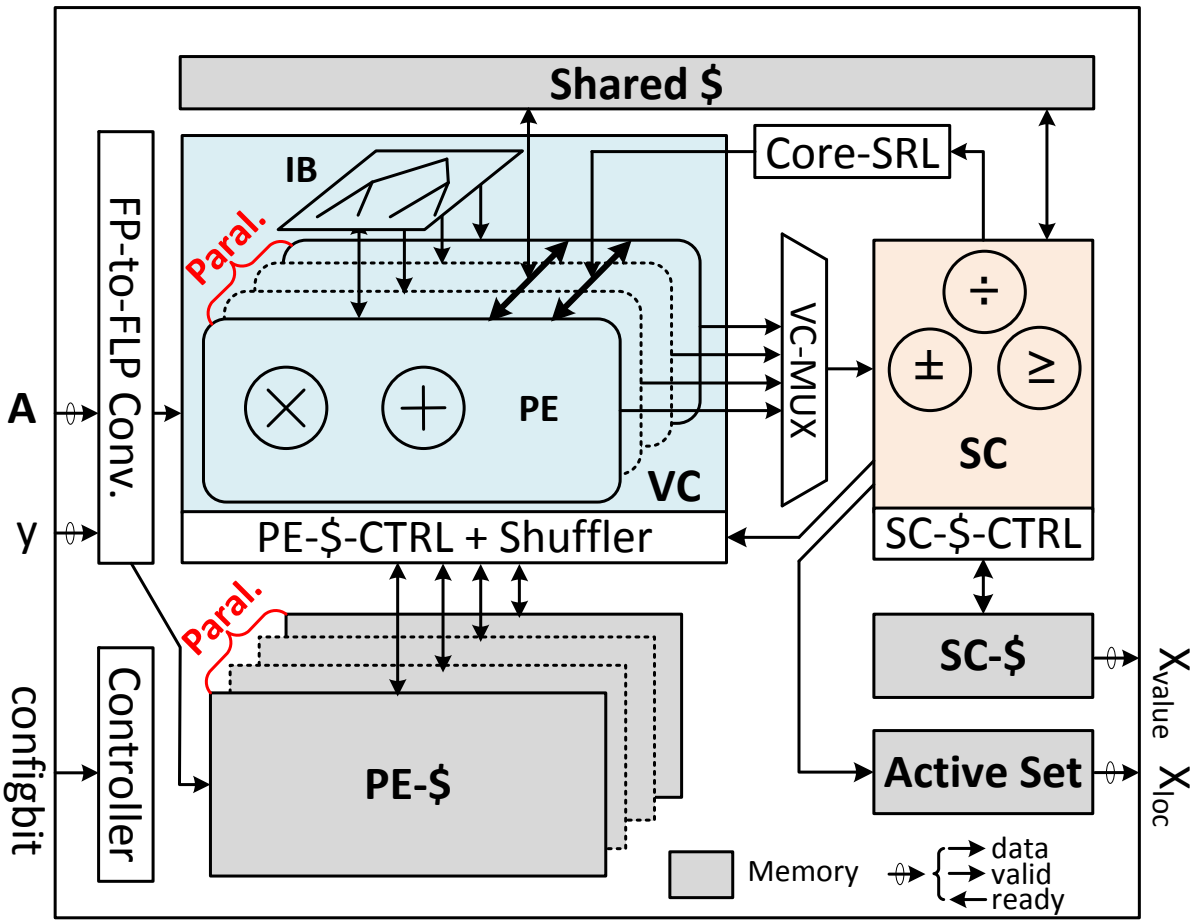



Figure 5.1: System architecture of the SA engine.

division. Depending on the top-level data-path configuration, SC can either post-process 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 that is accessible to all the PEs in the VC and the SC is deployed. In addition, a core-level shift-register logic (SRL) unit (Core-SRL) is used as a shortcut to connect the SC with all the PEs. This customized feedback path effectively reduces the loop latency between VC and SC to 3–12 cycles (depending on the

pipeline stage settings of multipliers and adders), thereby greatly accelerating the iterative BLA operations such as FS and BS. A separate memory unit is dedicated for storing the index of the active set. The controller of this memory unit is also responsible for accessing the data in the sampling matrix $A$ 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. The SA engine uses first-in-first-out (FIFO) interfaces to handle the flow control at the data inputs and outputs. Specifically, a handshaking protocol consisting of a "valid" and a "ready" signal is used to avoid data over-flow. The "valid" signal is generated if the sender's buffer is not (almost) empty, indicating that the data is valid at the output port of the sender. The "ready" signal is generated when the the receiver's buffer is not (almost) full, implying that the data can be captured at the next clock edge. Only when both the signals are asserted, data can be transferred at the next clock edge.

Note that there are several data-paths bridging the VC and SC in the system architecture. These include a feedthrough path realized via the VC-MUX, a feedback path realized using the Core-SRL, and a bidirectional path realized through the shared cache. The complex data flow of the reformulated OMP is enforced by the customized local memory controllers (PE-$-CTRL and SC-$-CTRL), which are coordinated by a top-level finite-state machine (FSM) (Controller). 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.

## 5.2 Computation Cores

### 5.2.1 Vector Core (VC)

The block diagram of the PE in the VC is shown in Fig. 5.2. 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

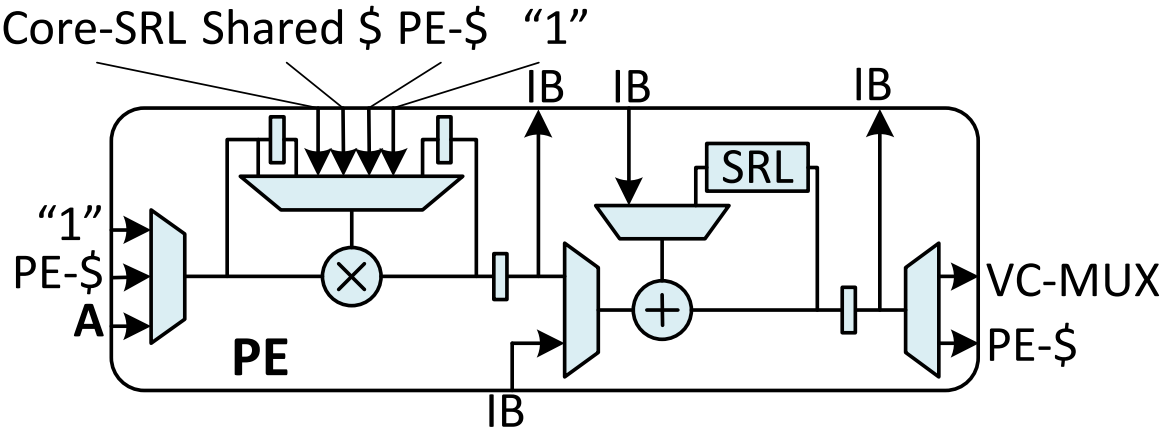



Figure 5.2: Block diagram of the PE in the VC.

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 input to 1, and the adder can be bypassed by resetting the SRL unit to output 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, multiple PEs are configured to support vector operations, such us vector addition, element-wise multiplication, element-wise MAC, and vector-scalar product, in a single-instruction-multiple-data (SIMD) fashion. To enable folded processing of long vectors, a SRL unit is inserted in the feedback path of each PE. The folding factor of the processing can be controlled by the latency of the SRL unit. In addition, the PEs can be coordinated through the interconnects inside the IB for computing vector inner products as illustrated in Fig. 5.3. Basically, the IB utilizes the built-in registers and multiplexers inside the PEs to connect the adders in different PEs into a pipelined adder tree, so that the element-wise products can be added up to a scalar. Note that 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. Different from the other supported vector operations, computing the folded inner product of long vectors needs an additional accumulator at the output of VC-MUX, which is integrated into the SC.

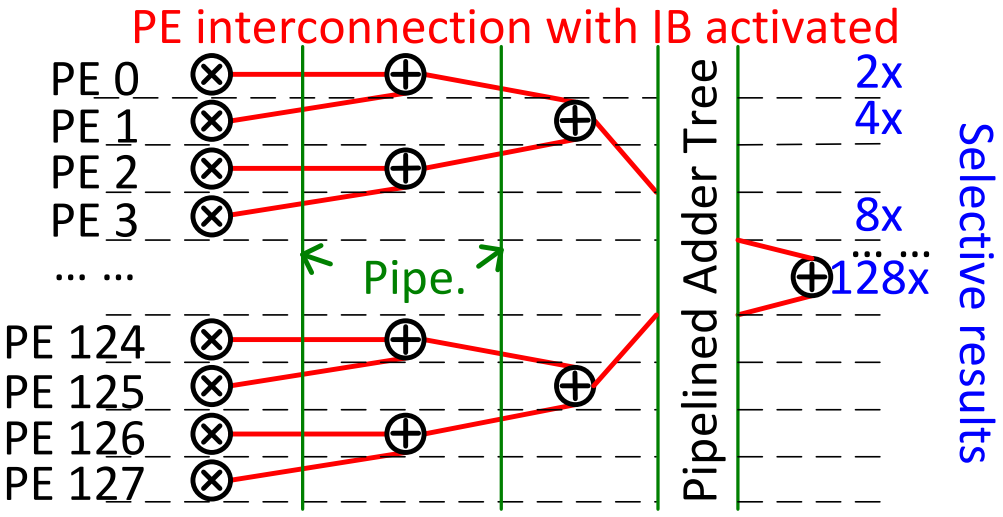



Figure 5.3: Illustration of the interconnects of 128 PEs for computing vector inner products.

### 5.2.2 Scalar Core (SC)

The block diagram of the SC is shown in Fig. 5.4. 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 tasks through the control bits of the multiplexers. When the SC is cascaded to the VC as a processing group, complex operations, such as correlation sorting, FS, and BS can be executed.

The adder at the first stage that connects to the VC-MUX plays critical roles in two tasks. First, it accumulates the result from the VC for performing folded inner product. Second, it adds the RHS of a linear equation to the LHS for performing FS and BS. The arithmetic units at the second stage are mainly used to post-process the results from the VC. For example, the sequential divider can be used to handle the inverse of a diagonal matrix as in (4.7). The comparator can be used to perform sorting tasks, such as sorting the correlation coefficients in the (hierarchical) AS task. The other adder is responsible for updating the results of folded inner product as in the Step 6a of Table 4.2, or to update independent data as in (4.23).

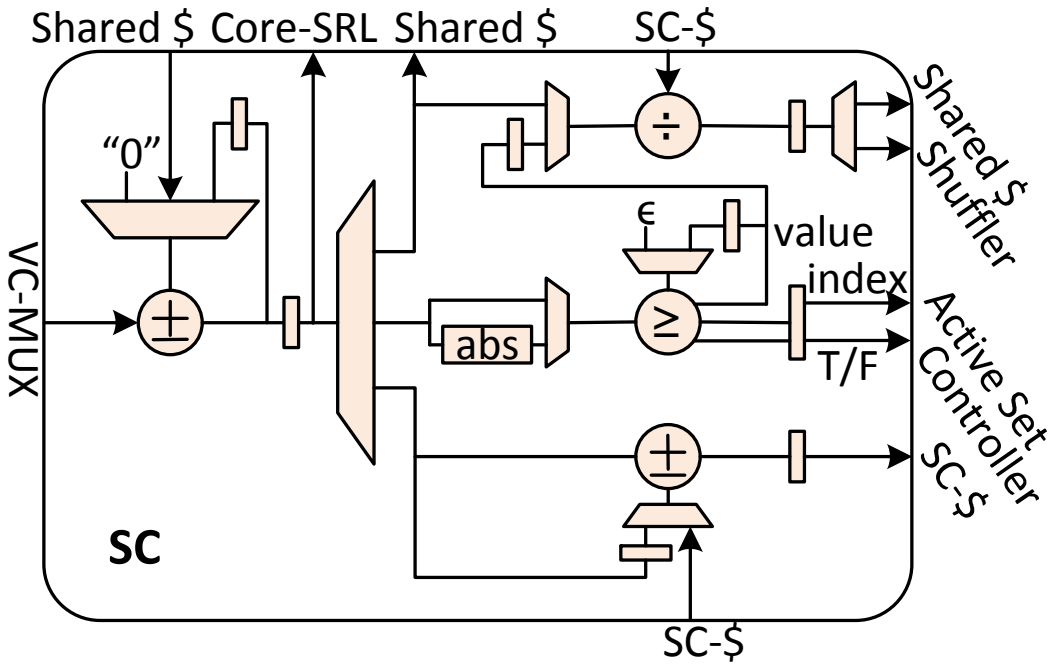



Figure 5.4: Block diagram of the SC.

## 5.3 Data Memory

For the CS reconstruction of different bio-medical signals, the sampling matrix $\boldsymbol{A}$ is often different due to the variety of their sparse domain. However, $\boldsymbol{A}$ serves as fixed coefficients and needs not to be updated during the reconstruction for the same problem. In order to accommodate the CS reconstruction on different basis, $\boldsymbol{A}$ must be explicitly stored in either on-chip or external memory. Note that all the elements in $\boldsymbol{A}$ need to be fully accessed once during the AS task in every iteration. This leads to a high memory access intensity. Specifically, one data has to be accessed per two FLOPs on average. Consequently, the memory bandwidth $B$ required by the VC for parallel processing can be characterized as

$$B = f_{PE} \cdot \frac{N_{PE}}{2} \cdot W \cdot p, \tag{5.1}$$

where $f_{PE}$ is the operating frequency of PEs, $N_{PE}$ is the number of FLOPs performed by a single PE in each clock cycle ($N_{PE} = 2$ in the reformulated OMP), $W$ is the data word-length, and $p$ is the parallelism (number of PE) of the VC. Note that the system throughput

will become memory-bounded if the RHS of (5.1) is greater than the LHS. In this case, further increasing $f_{PE}$ or $p$ will not boost the throughput but only degrade the power and area efficiency of the design. On the other hand, when the LHS of (5.1) is greater, the available memory bandwidth in the system is not fully utilized. Further speed-up can be achieved through more parallelization of PE or more pipelining if applicable. Therefore, it is critical to balance the memory bandwidth to match the system throughput following (5.1) if on-chip memories are used. In our FPGA implementations, we utilize the abundant block RAM (BRAM) resources on chip to realize a balanced system performance for parallel processing.

Different from $\boldsymbol{A}$, all the other variables in the reformulated OMP algorithm need to be updated in every iteration. At iteration $t$, the residual $r_t$, the active set $\Lambda_t$, the estimation $x_t$, and the Cholesky factorization matrices $\boldsymbol{L}$, $\boldsymbol{D}$ are updated based upon their values from iteration $t-1$. Consequently, these variables cannot share the same memory space through time-multiplexing as they all have to be carried over to the next iteration. In our design, $r_t$ and $\boldsymbol{L}$ are stored in the PE caches since their parallel access is required. $x_t$ and $\Lambda_t$ is stored in the SC cache and the active set memory, respectively. Note that $\boldsymbol{D}$ is a special case because its sequential access is needed for (4.13), while the parallel access is required by (4.14). Therefore, the diagonal elements of $\boldsymbol{D}$ are stored both across PE caches and in SC cache. Differently, the remaining variables in the reformulated OMP are all temporary. To maximize the memory utilization, they are buffered as intermediate results that share the same memory space in the shared cache.

## 5.4 Memory Control Scheme for Handling Cholesky Factorization

In the LS task of the reformulated OMP algorithm, a FS and a BS (see (4.6) and (4.8)) need to be performed in each iteration for realizing Cholesky factorization. Note that column and row access of a triangular matrix $\boldsymbol{L} \in \mathbb{R}^{k \times k}$ is required for performing FS and BS, respectively. 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 $\boldsymbol{L}^T$ as a regular square matrix as

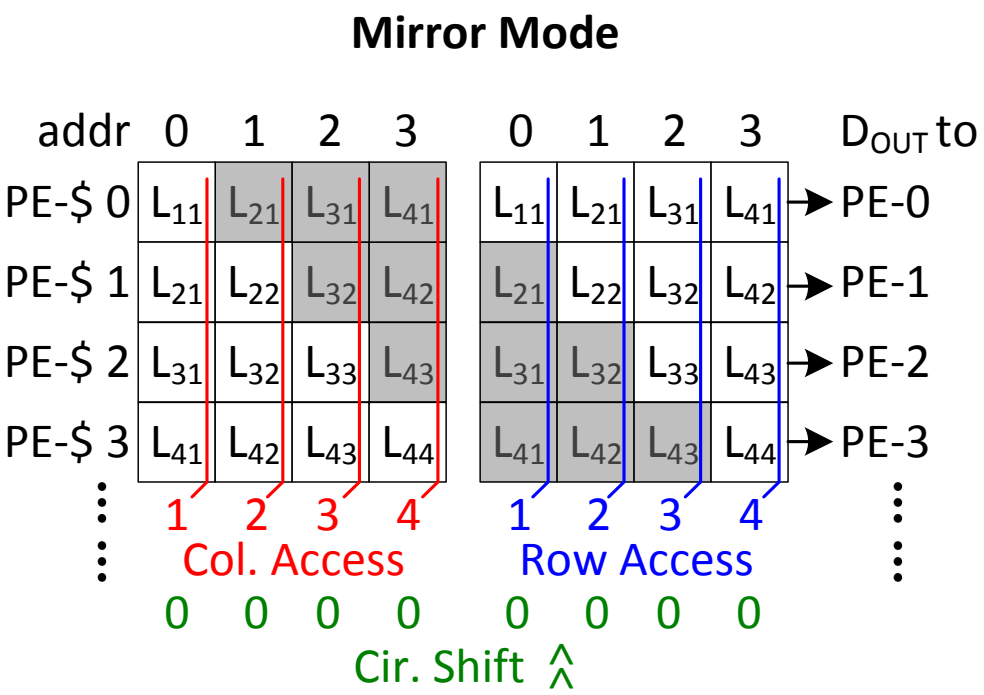



Figure 5.5: Data mapping scheme of PE caches in the mirror mode for handling Cholesky factorization.

illustrated in Fig. 5.5.

We refer to this memory 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 the instance in Fig. 5.5, the column vectors $l_1$, $l_2$, and $l_3$ can be accessed in parallel by reading the data at address 0, 1, and 2 of each PE cache, respectively. Also, the row vector $l_1^T$, $l_2^T$, and $l_3^T$ 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 it allows a larger square matrix to fold into smaller $p \times p$ blocks so that a larger-size ($k > p$) Cholesky factorization can be computed in a folded fashion with the help of the PE-SRL and the Core-SRL units. An example folding scheme in the mirror mode is illustrated in Fig. 5.6, where a folding factor of 1.5 is presented with $p = 128$ and $k = 192$.

However, this merit is enjoyed at the cost of doubled storage space for $\boldsymbol{L}$. For the low-

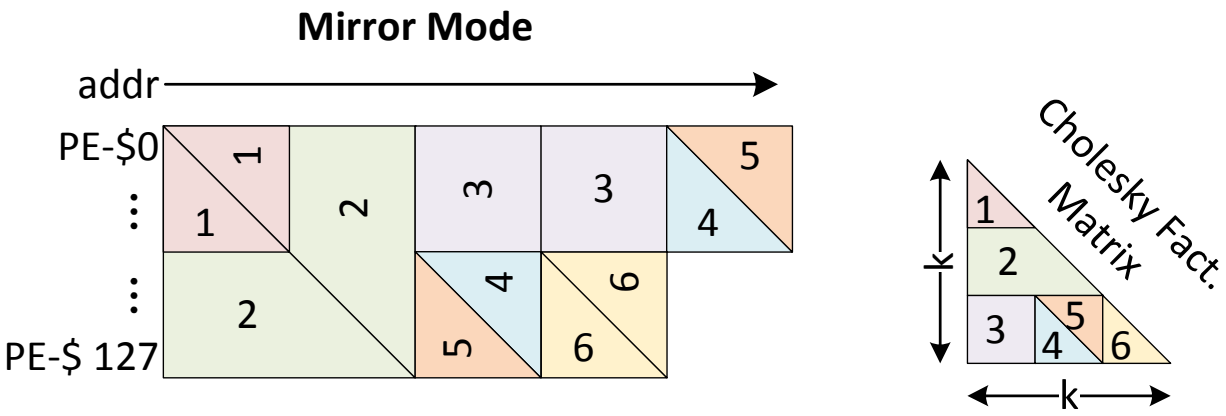



Figure 5.6: Data folding scheme of PE caches in the mirror mode for handling Cholesky factorization, where $p = 128$ and $k = 192$.

power implementation of the SA engine, where memory leakage dominates the total power consumption, a data shuffling scheme that is more efficient in utilizing memory space should be adopted. In the shuffle mode, each row of $L$ is stored in a shuffled order across adjacent PE-caches as illustrated in Fig. 5.7. As a result, the rows and columns of $L$ can be accessed at the same and at a different address of each PE-cache, respectively. Note that a circular position shift must be performed to recover the data order correctly. For the example in Fig. 5.7, the column vectorsS $l_1$, $l_2$, and $l_3$ can be accessed in parallel by reading the data at address 0, 1, and 2 of each PE cache with an up-shift in position by 0, 1, and 2, respectively. Differently, the row vectors $l_1^T$, $l_2^T$, and $l_3^T$ can be accessed in parallel by reading the data at the address set $[A_{PE-\$0},\ A_{PE-\$1},\ A_{PE-\$2},\ A_{PE-\$3}]$ of $[0,\ 1,\ 2,\ 3]$, $[X,\ 0,\ 1,\ 2]$, and $[X,\ X,\ 0,\ 1]$ with an up-shift in position by 0, 1, and 2, respectively.

Compared to the mirror-mode, a 2× memory size reduction can be achieved by adopting the shuffle mode. Note that special folding scheme has be realized when the folded processing of larger Cholesky factorization ($k > p$) is needed. An example folding scheme of the same case as in Fig. 5.6 is illustrated in Fig. 5.8.

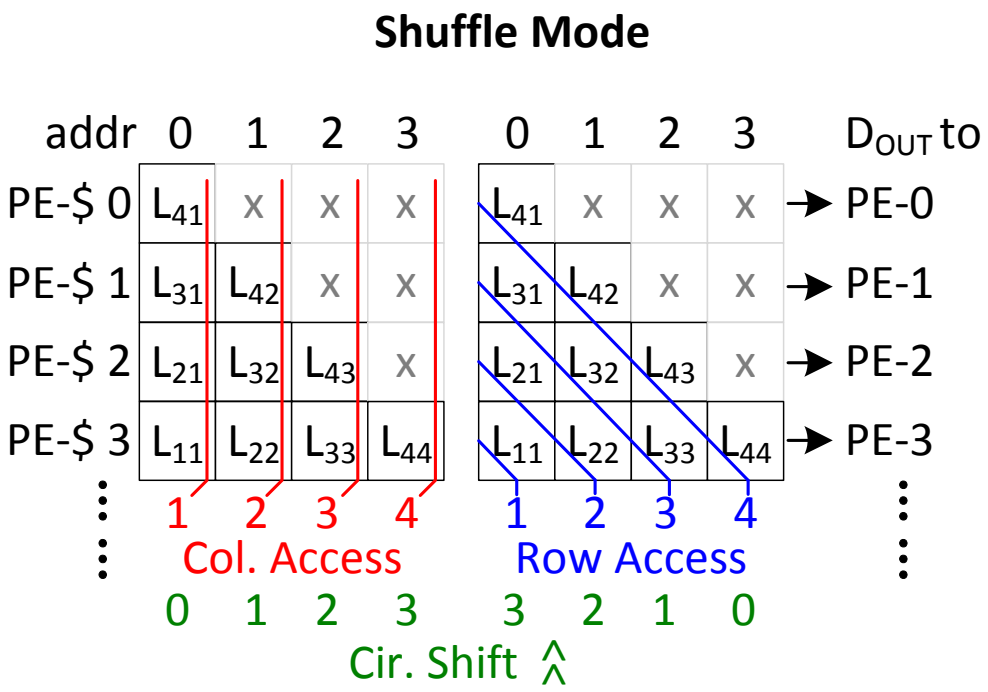



Figure 5.7: Data mapping scheme of PE caches in the shuffle mode for handling Cholesky factorization.

## 5.5 Dynamic Configuration of System Architecture

Taking advantages of the reformulated OMP algorithm that has a much simplified LS task, we manage to reuse the same computing resources to perform all the tree tasks by dynamically configuring the system architecture. Due to the intrinsic data dependency between the 6 BLA operations in Table 4.2, such a resource sharing scheme maximizes the hardware utilization rate and area efficiency without introducing throughput overhead.

Figure 5.9 illustrates the dynamic configuration of the system architecture in the AS task. 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 sampling matrix and the PE caches, respectively. The PEs are configured to compute their inner product as $c(i) = \boldsymbol{a_i}^T r_{t-1}$. 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 bigger value and the associated column index is buffered for the

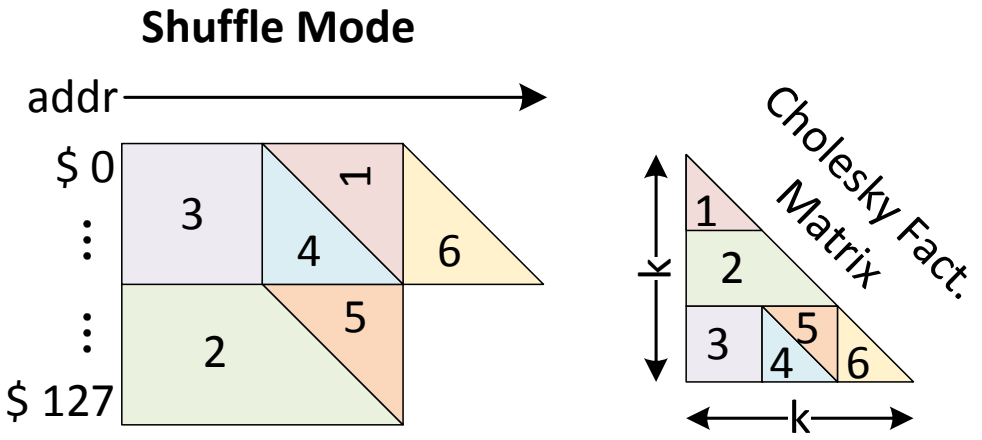



Figure 5.8: Data folding scheme of PE caches in the shuffle mode for handling Cholesky factorization, where $p = 128$ and $k = 192$.

next comparison. After all the correlation coefficients are compared, the column index of the maximum component is written into the active set memory.

Figure 5.10 illustrates the dynamic configuration of the system architecture in the LS task. In the LS task, a series of matrix-vector multiplications, FS, divisions, and BS need to be executed as shown in Step 4–7 in Table 4.2. For computing matrix-vector multiplications in Step 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 folding capability for computing large-size FS and BS. Figure 5.11 illustrates the detail data-path configuration of the VC and SC for computing the FS shown in (4.6). 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 iterative process that has loop-carried data dependency, and (2) the LS task is not the throughput bottleneck in the reformulated OMP as shown in Fig. 4.4. 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 (4.6) is executed iteratively using the configuration shown in Fig. 5.11, the subsequent divisions can be then scheduled to the SC and executed by the

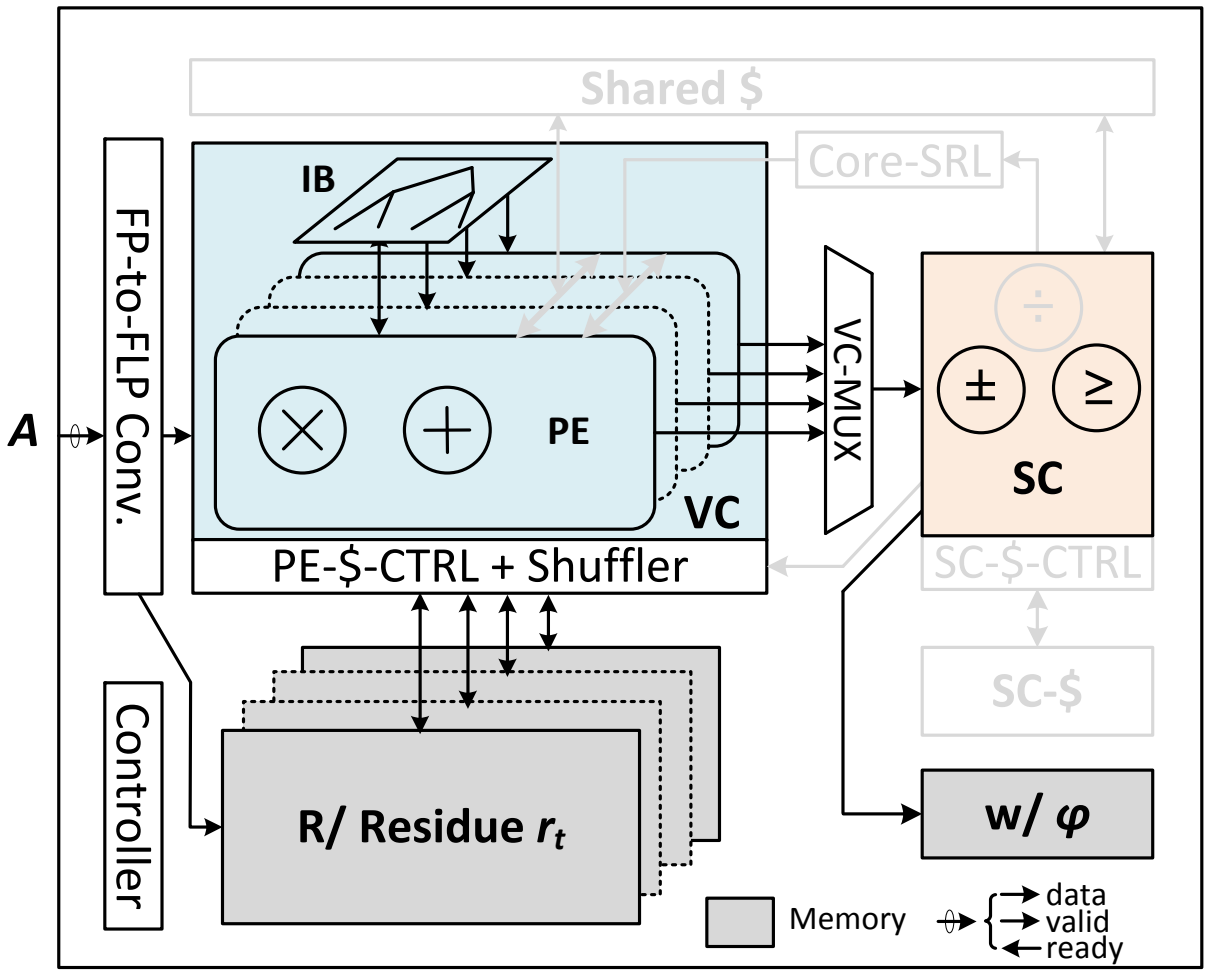



Figure 5.9: Dynamic configuration of the system architecture in the AS task.

sequential divider in pipeline.

Figure 5.12 illustrates the dynamic configuration of the system architecture in the EU task. In the EU task, the two computation cores are configured to update the estimation results separately. The VC accesses $\boldsymbol{A}_{\Lambda_t}$ and $d(\Lambda_t)$ from the sampling matrix memory and the shared cache, respectively. Note that the active atoms of $\boldsymbol{A}$ 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 $v = \boldsymbol{A}_{\Lambda_t}d(\Lambda_t)$ in the EU task are computed in a column-wise fashion by configuring the PEs to execute element-wise MAC. Each clock cycle, one column of $\boldsymbol{A}_{\Lambda_t}$ and a single element of $d(\Lambda_t)$ are multiplied, and the

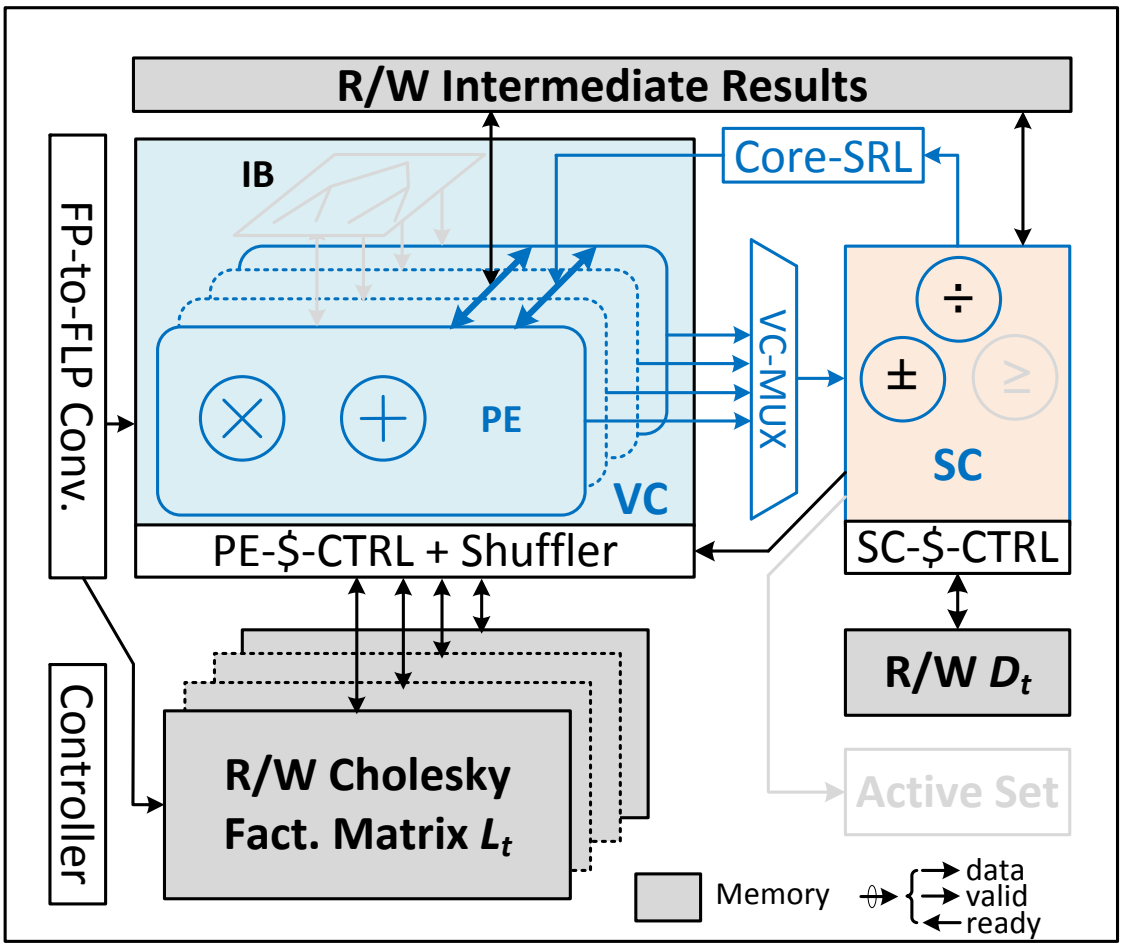



Figure 5.10: Dynamic configuration of the system architecture in the LS task.

results are accumulated element-wise in the SRL units of each PE. 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 $r_t = r_{t-1} + v$. In the 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.

Overall, the dynamic configuration scheme of the system architecture is summarized in Fig. 5.13.

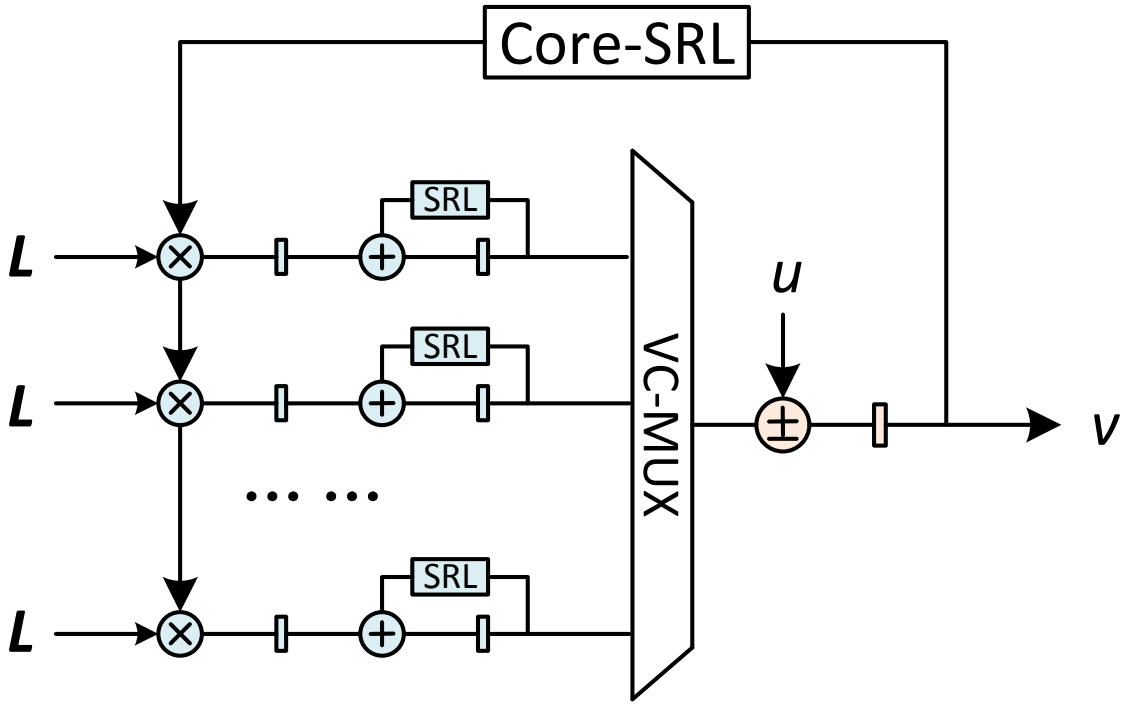



Figure 5.11: Data-path configuration for computing FS and BS.

## 5.6 Data Format for Preserving Software-Level Accuracy

To make the soft-IP core more suitable for the intended application (see Section 1.2), we are interested in applying the data format that can preserve the software-level accuracy for recovering different bio-medical signals. Therefore, we investigate the dynamic range requirement of each arithmetic unit in our design based upon the statistics extracted from double-precision MATLAB simulations. In the simulations, different biomedical signals are emulated by artificially generating their sparse coefficients on the selective orthogonal basis. These include canonical basis, discrete wavelet transform (DWT) basis, discrete cosine transform (DCT) basis, joint DWT-DCT basis, etc. Note that the sparse coefficients are also scaled such that they follow the power-law decay described by (2.8). The generated signals are sampled by Bernoulli random matrices and contaminated by AWGN. Then, the dynamic range statistics in each step of the reformulated OMP algorithm is extracted from the subsequent signal reconstructions. Note that the emulation method used in this

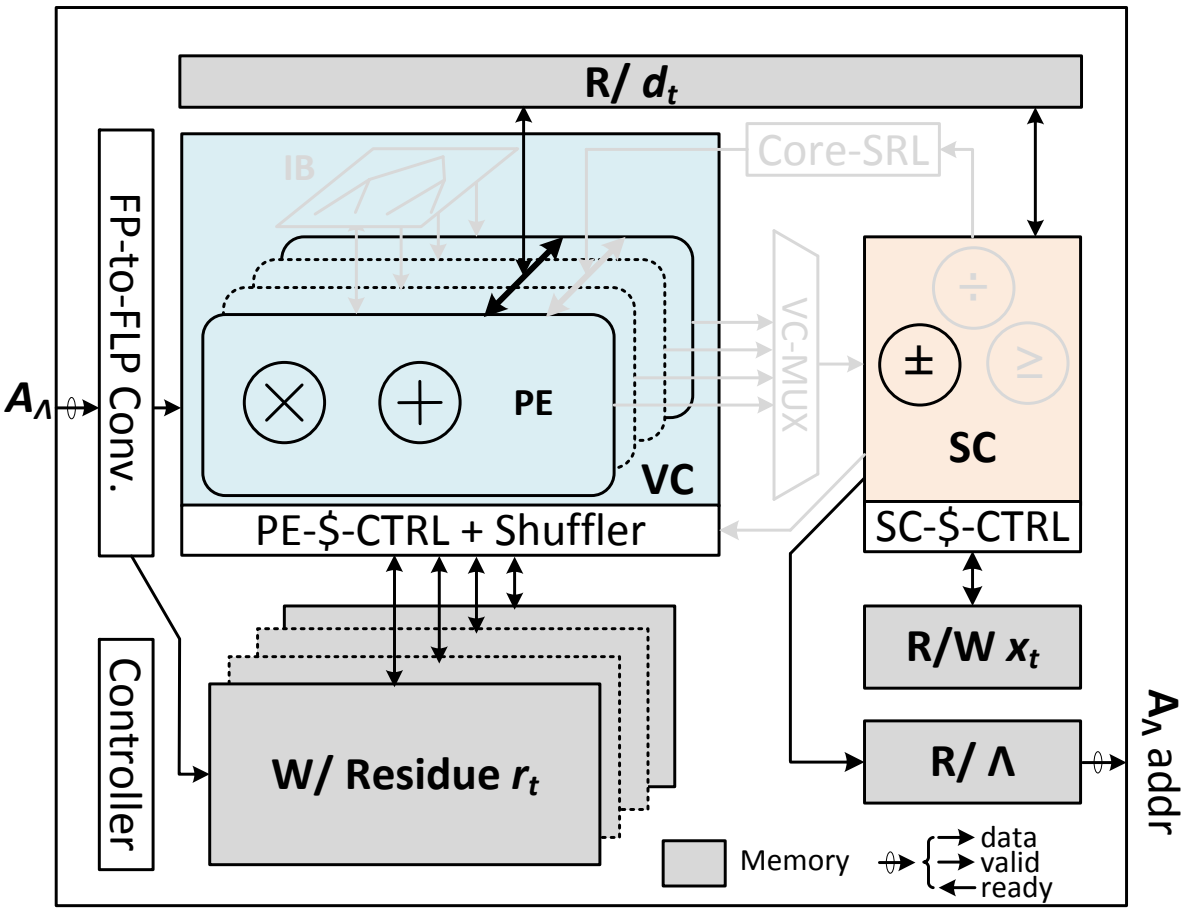



Figure 5.12: Dynamic configuration of the system architecture in the EU task.

study is a first-order approach that estimates the worst-case dynamic range requirement for recovering different bio-medical signals. To determine the optimized word-length for a specific signal type, Monte Carlo simulations needs to be performed with more representative signal samples.

Figure 5.14 shows the worst-case dynamic range required by each arithmetic unit for preserving the software-level accuracy. When a fixed-point data format is used, the word-length required by the MAC unit in the VC, and the adder, the divider, the comparator unit in the SC is 78, 35, 73, and 39 bits, respectively. The large dynamic range requirement is largely due to the solution searching characteristic of OMP: the scales of most variables are gradually scaled down as the residual approaches zero. Additionally, sharing the computing resources in different tasks also have negative impact on the dynamic range requirements.

| Task | | PE | PE-SRL | IB | VC-MUX | Core-SRL | SC | PE-$ | SC-$ | Share $ | Active Set |
|---|---|---|---|---|---|---|---|---|---|---|---|
| AS | | Inner prod. | Off | On | Static | Off | ACC, CMP | R/ Residue | Off | Off | W/ Index |
| LS | Op 1 | Inner prod. | Off | On | Static | Off | ACC | Off | Off | W/ IR | R/ Index |
| | Op 2 | MAC | On | Off | Dyn. | On | ACC, DIV | R/W L | R/ D | R/W IR | Off |
| | Op 3 | Mult | Off | On | Static | Off | ACC | R/ L | W/ D | R IR | Off |
| | Op 4 | MAC | On | Off | Dyn. | On | ACC, DIV | R/ $L^T$ | Off | W/ IR | Off |
| EU | | MAC | On | Off | Off | Off | ACC, ADD | R/W residue | R/W x | R/ IR | R/ Index |

IR = Intermediate Result, R/W = read/write, x = sparse coefficient

Figure 5.13: Summary of dynamic configuration scheme of the system architecture.

This is because the shared units must cover the worst case of all the three tasks. The results in Fig. 5.14 indicate that a fixed-point implementation would incur large area overhead mainly due to the need for building wide data memories. Alternatively, a floating-point data format is able to provide the required dynamic range with much reduced word-length, leading to significant area saving primarily from the data memories.

Nonetheless, the sampling matrix $\boldsymbol{A}$ is still stored in a fixed-point data format in our design. This is because $\boldsymbol{A}$ serves as fixed coefficients and does not need to be updated during the operations for the same problem. In addition, $\boldsymbol{A}$ often has higher tolerance to quantization noise, especially when it is a random matrix. Therefore, $\boldsymbol{A}$ typically can be well represented with a limited dynamic range. Note that for a different accuracy requirement, the choice of data format should be reinvestigated accordingly.

## 5.7 Compile-Time Scalability and Run-Time Configurability

The soft-IP core developed in Verilog-HDL is parameterized and features great scalability at compile time. Table 5.1 summarizes all the user-defined parameters supported in the soft-IP core. At the circuit level, the user can specify the word-length ($W_M,\ W_E$) and the pipeline stages ($S_{ADD}, S_{MULT}, S_{DIV}, S_{CMP}$) of each arithmetic unit to meet different precision and performance requirements. At the architecture level, the user can tune the parallelism of

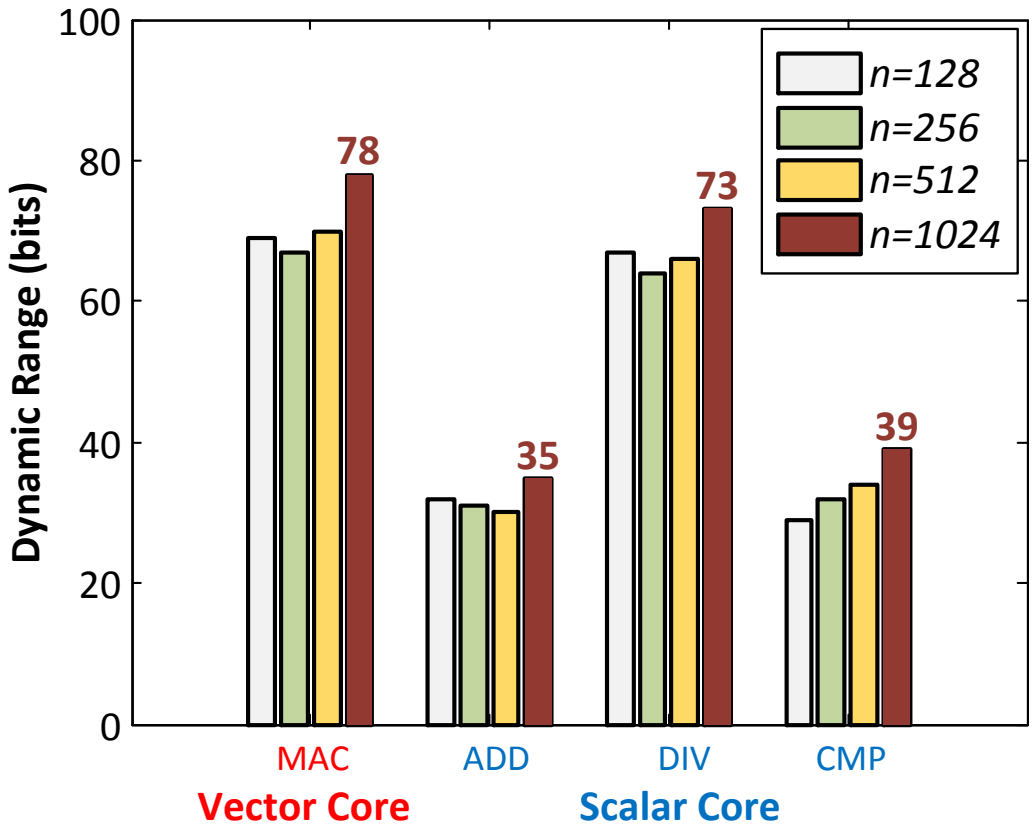



Figure 5.14: Worst-case dynamic range required by hardware units to preserve the software-level accuracy for varying problem size $n$.

PE ($P$) to adjust the scale of vector processing. A higher parallelism of PE can either improve system throughput or allow the designer to scale down the supply voltage for the same throughput to gain energy efficiency at the cost of increased area. In addition, the user can specify the maximum problem size ($N$, $M$, $K$) to be supported by the SA engine. Since the data memories are separated from the computation cores, the scaling of one part has little dependency on the other given that the timing difference in control signals needs to be handled by adjusting the FSM in local controllers. Therefore, the size of data memories and the timing of controllers are adjusted coherently, according to the architecture parameters ($P$, $N$, $M$, $K$) defined by the user. By customizing the parameters in Table 5.1, different Verilog descriptions of the SA engine can be customized at compile time for driving different applications.

Table 5.1: User-defined Parameters in the SA Engine Soft-IP

| | Parameter | Description |
|---|---|---|
| Circuit | $\{W_M,\ W_E\}$ | Word-length of mantissa and exponent |
| | $\{S_{ADD}, S_{MULT}, S_{DIV}, S_{CMP}\}$ | Pipeline stages of arithmetic units |
| Architecture | $P$ | Parallelism of PEs in the VC |
| | $N$ | Maximum signal dimension $n$ |
| | $M$ | Maximum measurement dimension $m$ |
| | $K$ | Maximum signal sparsity level $k$ |

The VLSI architecture is also configurable at run time. Specifically, every compiled instance of the SA engine can be configured through the dynamic control bits at the input to handle flexible problem settings on the fly. The input control signals include signal and measurement dimensions ($m \leq M$ and $n \leq K$), signal sparsity level ($k \leq K$), and error tolerance ($\epsilon$). These control inputs can be reconfigured during the data loading and result unloading at the end of each SA run. But, they must remain static during the same SA run. In addition, the reconstruction on the different basis can be performed by loading different coefficients from the sampling matrix memory. Note that changing the control bits will not affect the computing throughput (FLOPs/s) of the VC but can change the system throughput (Samples/s) due to the resulting difference in total computational complexity.

# CHAPTER 6

# FPGA Evaluation

In this chapter, we elaborate on the FPGA evaluation of the developed soft-IP core. Specifically, the FPGA implementation results in comparison to prior designs is first reported. Additionally, the accuracy and performance benchmarking results in comparison to an Intel Core i7-4700MQ mobile processor are discussed.

## 6.1 Xilinx KC705 Hardware Evaluation Platform

The SA engine soft-IP is first implemented and tested on a Xilinx KC705 evaluation platform as shown in Fig. 6.1 [Xil12]. This evaluation platform is equipped with a Kintex-7 XC7K325T-2FFG900C FPGA and a number of off-the-shell components for enhancing the system capabilities in memory storage, connectivity, networking, etc.

Specifically, the hardware resources available on the XC7K325T-2FFG900C FPGA include:

- 50,950 slices (equivalently 326,080 logic cells)
- 840 DSP slices
- 16,020 Kb block RAMs (BRAMs)
- 10 clock management tiles (CMTs)
- 1 PCI express (PCIe) module
- 16 Giga-bit transceivers (GTXs)

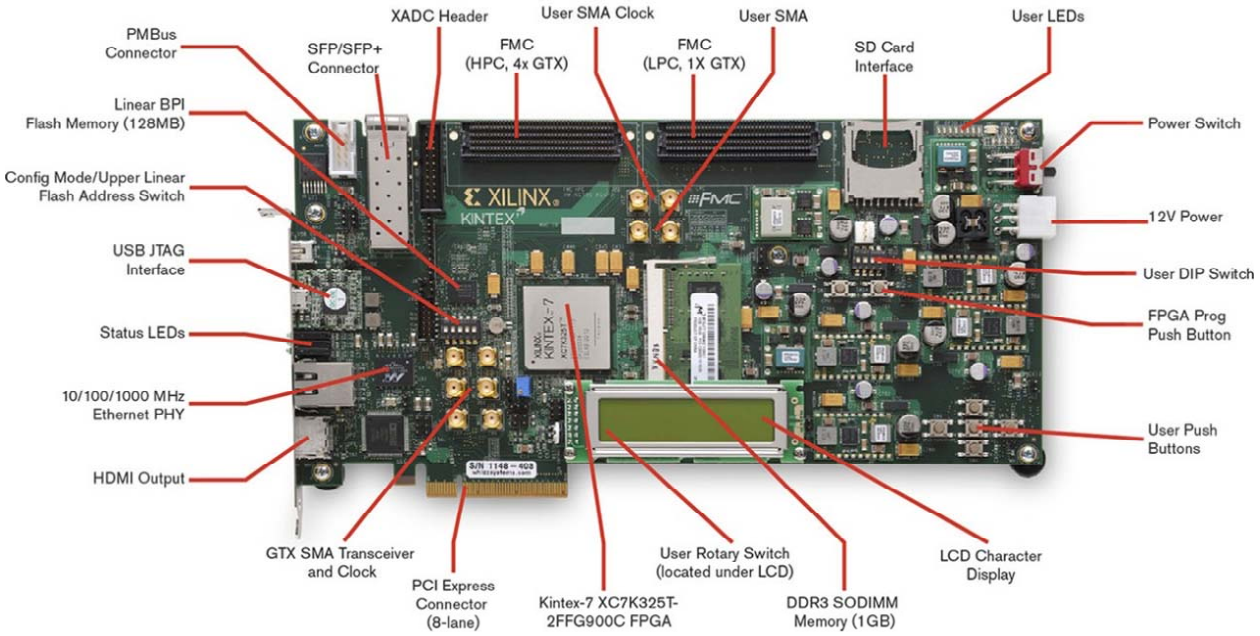



Figure 6.1: Xilinx KC705 evaluation board (courtesy of Xilinx, Inc.).

- 1 Xilinx analog-to-digital converter (XADC)
- 10 input/output (I/O) banks with 500 maximum user I/Os

The memory resources available on the KC705 board include:

- 1GB DDR3 SODIMM 800MHz / 1600Mbps
- 128MB (1024Mb) Linear BPI Flash for PCIe Configuration
- 16MB (128Mb) Quad SPI Flash
- 8Kb IIC EEPROM
- SD Card Slot

The glue logic resources for communication and networking include:

- Gigabit Ethernet GMII, RGMII and SGMII
- SFP / SFP+ cage

- GTX port (TX, RX) with four SMA connectors
- UART to USB Bridge
- PCI Express x8 edge connector

The expansion connectors for connecting customized PCB boards include:

- FMC-HPC (Partial Population) connector (4 GTX Transceiver, 116 single-ended or 58 differential (34 LA and 24 HA) user defined signals)
- FMC-LPC connector (1 GTX Transceiver, 68 single-ended or 34 differential user defined signals)
- Supported I/O voltages of 1.8V, 2.5V, or 3.3V
- Dedicated IIC pins

The miscellaneous peripherals on the KC705 board include:

- Onboard JTAG configuration circuitry to enable configuration over USB
- JTAG header provided for use with Xilinx download cables
- 128MB (1024Mb) Linear BPI Flash for PCIe Configuration
- 16MB (128Mb) Quad SPI Flash
- HDMI Video output
- External Phy/codec device driving an HDMI Connector
- 2x16 LCD display
- 8x LEDs
- XADC header
- 5X Push Buttons

- 4X DIP Switches
- Diff Pair I/O (1 SMA pair)
- AMS FAN Header (2 I/O)
- 7 I/O pins available through LCD header

## 6.2 Computer-Aided Design (CAD) Flow

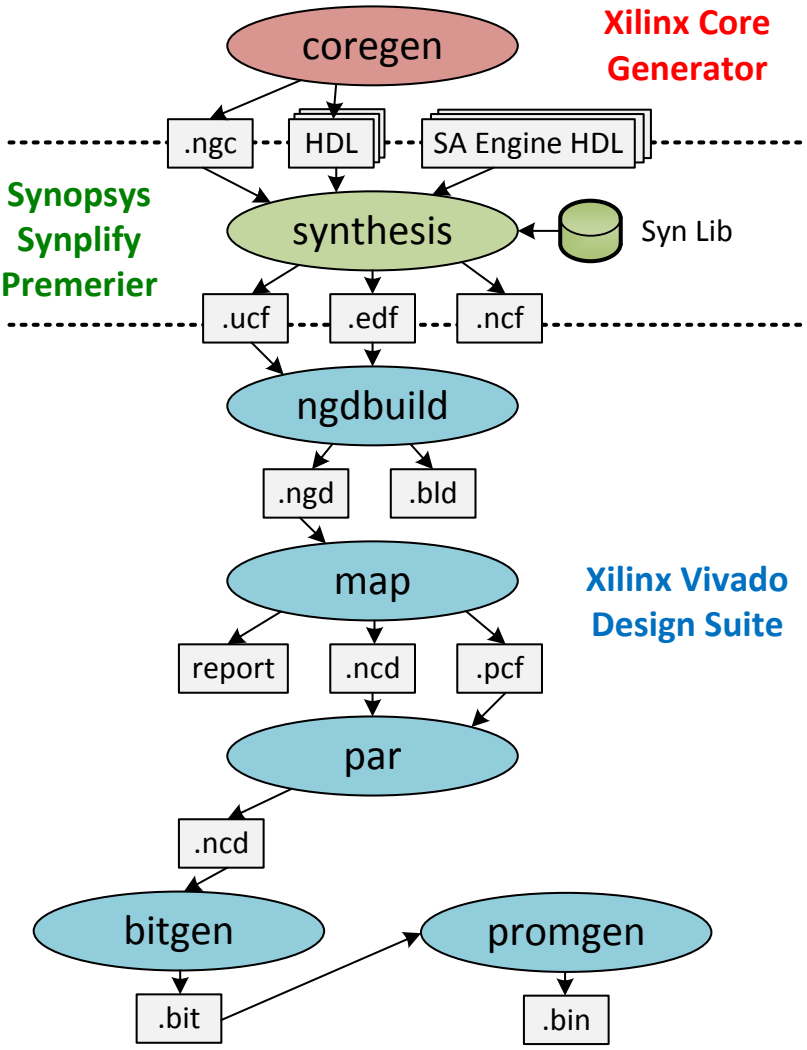



Figure 6.2: CAD flows for implementing the SA engine on the KC705 platform.

The CAD flows for implementing the SA engine on the KC705 Platform is illustrated in Fig. 6.2. The Xilinx CORE Generator System is first used to generate the data memory macros that are mapped to the BRAM resources on the FPGA. The generated NGC files contain both the design netlist and the constraints, while the generated Verilog files describe the port definitions. Then, these files together with the RTL codes of the SA engine are loaded to Synplify Premier for logic synthesis. Note that the floating-point arithmetic units used in our design are from the Synopsys DesignWare library. The EDF file stores the gate-level netlist in an electronic data interchange format (EDIF), and the UCF file contains user-defined design constraints. Next, the generated files are passed to Xilinx Vivado Design Suite to perform physical design and bit file generations. Specifically, the "ngbbuild" command reads in the netlist in EDIF format and creates a native generic database (NGD) file that contains a logical description of the design reduced to Xilinx NGD primitives and a description of the original design hierarchy. The "map" command takes the NGD file, maps the logic design to a specific Xilinx FPGA, and outputs the results to a native circuit description (NCD) file. The "par" command takes the NCD file, places and routes the design, and produces a new NCD file, which is then used by the "bitgen" command for generating the bit file for FPGA programming. For a faster programming in the case of large design, the "promgen" command can be used to convert the bit file into a binary format.

## 6.3 Implementation Results

By efficiently utilizing the resources on the FPGA, we are able to deploy 128 single-precision PEs in the VC, and the reconstruction engine can support a flexible problem size of $n \leq 1680$, $m \leq 640$. In addition, our implementation can support up to $k \leq 300$ sparse coefficients in signal reconstruction, which is 10 times more than prior work [Sa10, Sa12, Ba12]. Supporting more coefficients in reconstruction is equivalent to adding finer details to the recovered signal. This is critical to achieving high reconstruction accuracy, especially when the signal is not ideally sparse on the selective basis.

[1]DM: data-path memory

Table 6.1: Implementation Results on FPGA

| | | Usage Breakdown | | |
|---|---|---|---|---|
| | System | VC | SC | DM [1] |
| Frequency | 53.7 MHz | | | |
| LUTs | 186,799 (91%) | 90% | 3% | 7% |
| DSP48s | 258 (31%) | 99% | 0% | 1% |
| BRAMs | 435 (98%) | 0% | 0% | 100% |

Table 6.1 summarizes the system performance and reports the resource utilization of the FPGA implementation. After inserting 1, 1, 1, and 6 pipeline stages into the adder, multiplier, comparator, and sequential divider, respectively, and global retiming, a balanced data-path delay distribution is observed. The critical paths are found to be part of the floating-point multiplier, which is massively distributed in the VC. As a result, the whole system achieves an operating frequency of 53.7 MHz. The implementation utilizes 91% of the LUTs, 31% of the DSP48 slices, and 98% of the BRAMs on the FPGA. All the BRAMs are used for building the data memories, where 65% of the usage is dedicated for storing $\boldsymbol{A}$ in a 10-bit fixed-point data format. Two DSP48 slices are utilized in each floating-point multiplier for performance improvement. Thus, a total of 256 DSP48 slices are used in the realization of 128 PEs. Note that among the total LUT usage, 90% is contributed to building the VC. This means the proposed architecture has a high area efficiency, which results from the algorithm reformulation and resource sharing efforts. As a result, our design is able to utilize 90% of the logic resources to parallelize the VC for achieving higher processing throughput. A layout view of the FPGA with the SA engine fully implemented is shown in Fig. 6.3.

In order to validate the flexibility of the soft IP, we implement multiple instances of the SA engine for handling different problem settings. These implementations are evaluated based on different FPGA platforms for comparison purposes. Table II summarizes the evaluation

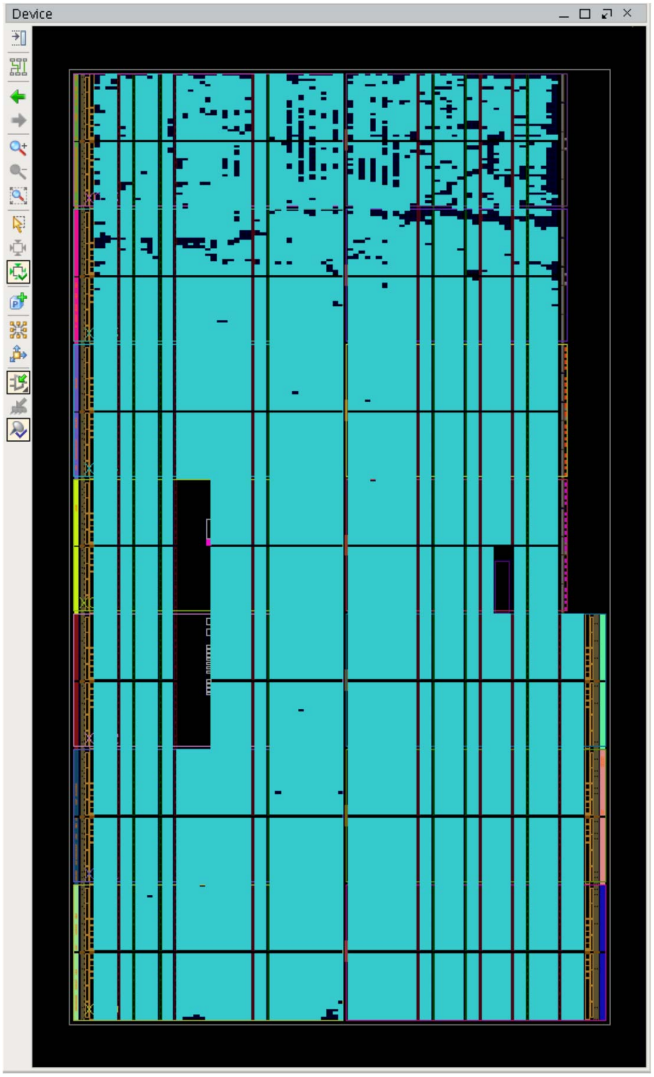



Figure 6.3: Layout view of the FPGA with the SA engine implemented.

results in comparison to prior FPGA designs [Sa10, Ma12, Ba12]. Taking advantage of the flexibility of the soft-IP core, we are able to explore the design space on the FPGA by comparing the mapping results of different parameter settings. Then, the one with the maximal performance and resource utilization is selected for the final implementation. As a result of the design space exploration, our implementations are able to outperform the prior work with up to 30% higher throughput while providing larger dynamic range capability and better flexibility.

| Design | [Sa10] | Our Work | [Ba12] | Our Work | [Ma12] | Our Work |
|---|---|---|---|---|---|---|
| Platforms | Virtex-5 | | Virtex-6 | | Spartan-6 | |
| N,M,K,P | 128,32,5,32 | | 1024,256,36,256 | | 1024,512,64*,32 | |
| P | 32 | 32 | 256 | 256 | 32 | 32 |
| Data Format† | FP (32) | FLP (8,23) | FP (25) | FLP (8,16) | FP (30) | FLP (8,21) |
| Frequency (MHz) | 39 | 59.3 | 100 | 77.6 | 41.2 | 64.4 |
| Slices | N/A | 12,330 | 32,010 | 62,026 | 3,525 | 15,769 |
| DSP48s | N/A | 64 | 261 | 256 | 132 | 98 |
| Dec. Time (µs) | 24 | 18.5 | 630 | 581.6 | 21,378 | 17,611 |
| Throughput# (Ksamples/s) | 5,333 | **6,919** | 1,625 | **1,761** | 47.9 | **58.1** |

Figure 6.4: Implementation results in comparison to prior work.

## 6.4 Benchmarking Study

### 6.4.1 Testing Environment

Figure 6.5 illustrates the testing setup for the benchmarking study. The evaluation platform is connected with a PC through both USB and Ethernet cables. The USB connection is used to program the FPGA and monitor the chip status via the ChipScope IPs deployed. The Ethernet connection supports high-speed data transfer between the PC and the evaluation platform. In order to bridge the SA engine to Ethernet, several Xilinx LogiCORE IPs are integrated on the FPGA as glue logic. Specifically, a tri-mode Ethernet media access control (TEMAC) module is instantiated to drive the 10/100/1000 MHz Ethernet PHY device equipped on the board through the Gigabit media independent interface (GMII). In addition, two asynchronous FIFOs are used to link the SA engine and the TEMAC module across different clock domains. To deal with the Ethernet frame format, customized FIFO controllers are used to perform decapsulation and encapsulation for the incoming and the

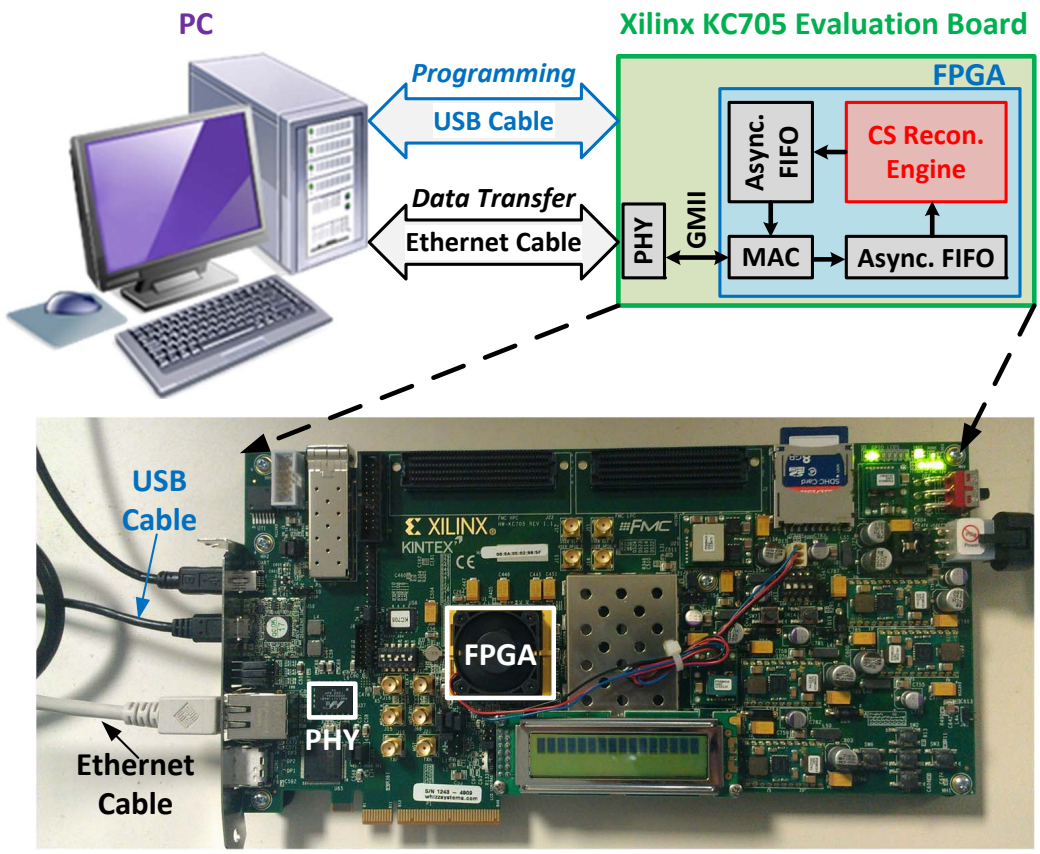



Figure 6.5: Testing environment on the Xilinx KC705 evaluation platform.

outgoing data streams, respectively. Overall, a full-duplex data communication channel that has a data transfer rate of 1 Gbps is established between the SA engine and the PC through the Ethernet cable. By utilizing the network infrastructure, the same setup can be also used for remote testing. When the evaluation platform is connected to Ethernet, test vectors can be sent to the SA engine in Ethernet frames from any terminal user belonging to the same local area network (LAN) by specifying the MAC address of the TEMAC as the destination address.

### 6.4.2 Accuracy Benchmarking

In order to evaluate the reconstruction accuracy, we use the ECG data from the MIT-BIH database [Ma01] to perform CS sampling and reconstruction in the benchmarking study. In

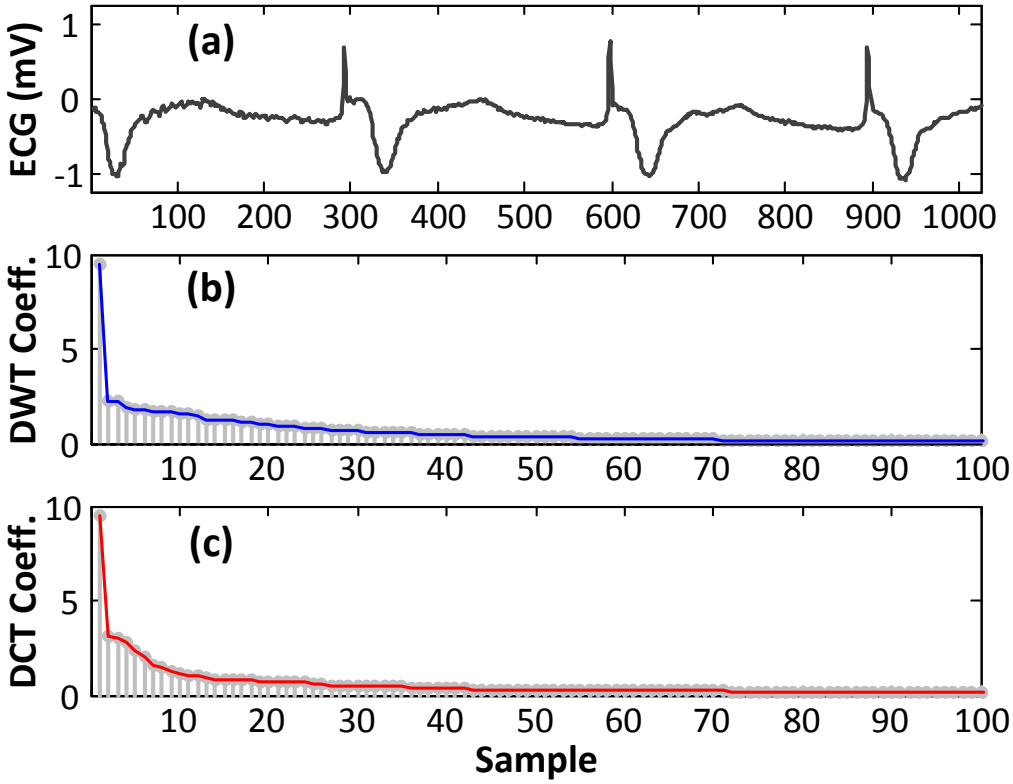



Figure 6.6: An example of (a) the ECG signal from MIT-BIH database and its top 100 sorted (b) DWT and (c) DCT coefficients.

the experiments, the ECG signals are segmented with a fixed window size of $n = 1024$ and sampled by a Bernoulli random matrix with different undersampling ($m/n$) ratios. Recall from the CS theorem that the signals can be uniformly sampled without specifying sparse domain. However, such prior knowledge must be provided in the signal reconstruction.

Figure 6.6 shows an example of the ECG signal and its top 100 coefficients on the Haar DWT and the DCT basis, respectively. Note that the coefficients follow a slightly faster power-law decay on the DWT basis. Although the coefficients seems to follow a power-law decay on the DCT basis also, the DCT basis is not suitable for ECG reconstruction. Figure 6.7 illustrates the ECG signal reconstructed on the DCT, the DWT, and a DWT-DCT joint basis, respectively. Because the spike component in the ECG signal has a wide band in the frequency domain, the algorithm fails to reconstruct the spike component that contains

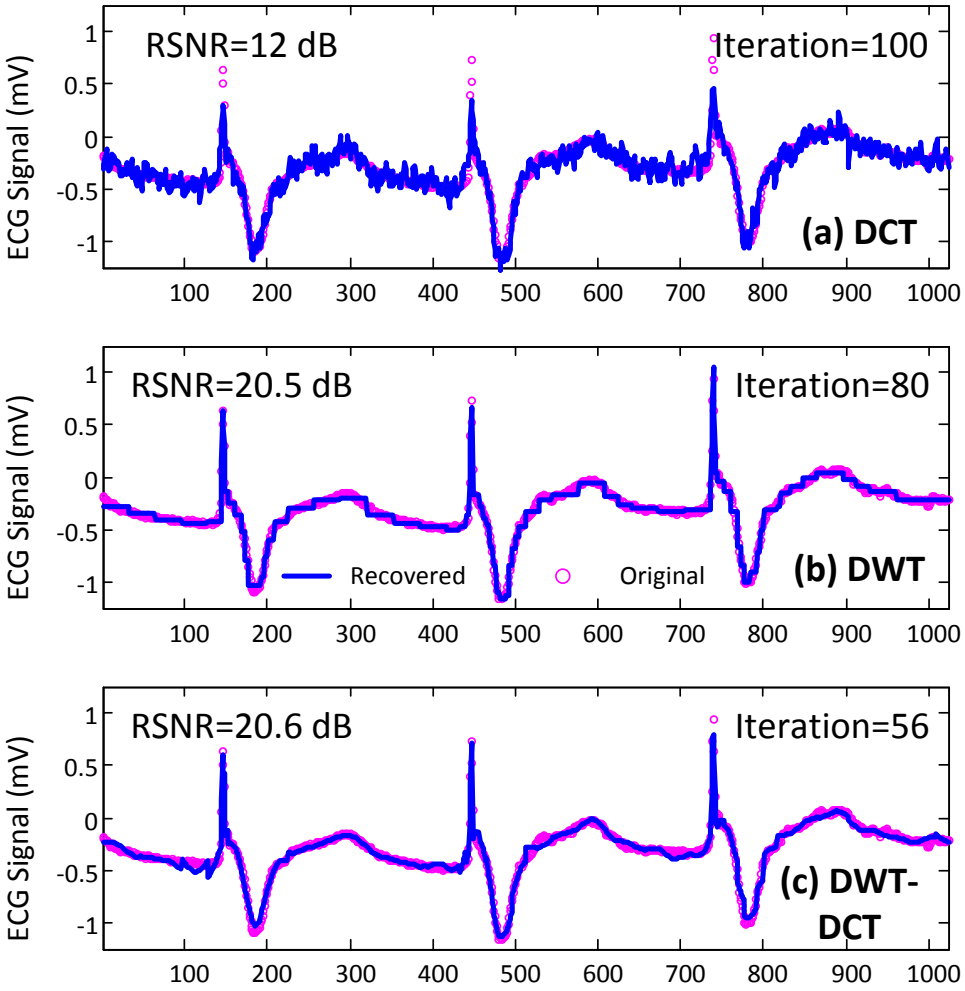



Figure 6.7: ECG signal reconstructed on (a) DCT, (b) DWT, and (c) DWT-DCT joint basis, respectively.

critical information by pursuing the sparsest representation on the DCT basis. Differently, on the DWT basis that is good at representing singularity, the spike component can be reconstructed nicely. By performing the reconstruction on the DWT basis, a 20.5 dB RSNR can be achieved in 80 iterations. In fact, an over-complete basis jointly composed by both the DCT and DWT basis can better reconstruct the ECG signal with the fewer coefficients. Since the periodic and the spike component has a sparse representation on the DCT and the DWT basis, respectively, the ECG signal can be well reconstructed on the DWT-DCT joint basis in much fewer iterations (with much fewer atoms). By performing the reconstruction

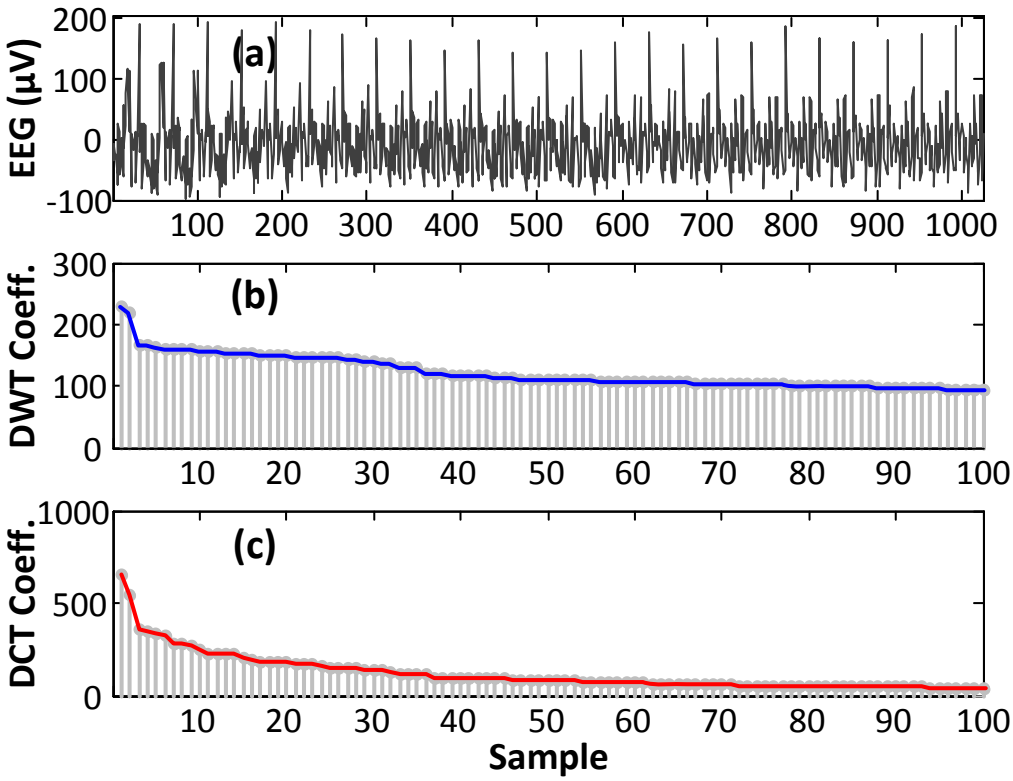



Figure 6.8: An example of (a) the EEG signal from UCSD-EEGLAB database and its top 100 sorted (b) DWT and (c) DCT coefficients.

on the DWT-DCT joint basis, a 20.6 dB RSNR can be achieved in just 56 iterations.

Fig. 6.6 plots an example of the EEG signal from the UCSD-EEGLAB database [Da04]. In this case, the DCT coefficient of the EEG signal is much sparser than its DWT counterpart. The two examples in Fig. 6.6 and 6.8 illustrate that different bio-medical signals often exhibit the best sparsity on different basis. Therefore, it is crucial to keep a generic architecture in hardware design so that different sampling matrix can be adopted for the reconstruction of different bio-medical signals.

For comparison purposes, we reconstruct the ECG signals on the Haar DWT basis using both C program and the SA engine on the FPGA. The reconstruction accuracy is measured by RSNR defined in (3.16). Note that RSNR characterizes the energy ratio of original signal and the reconstruction error in dB.

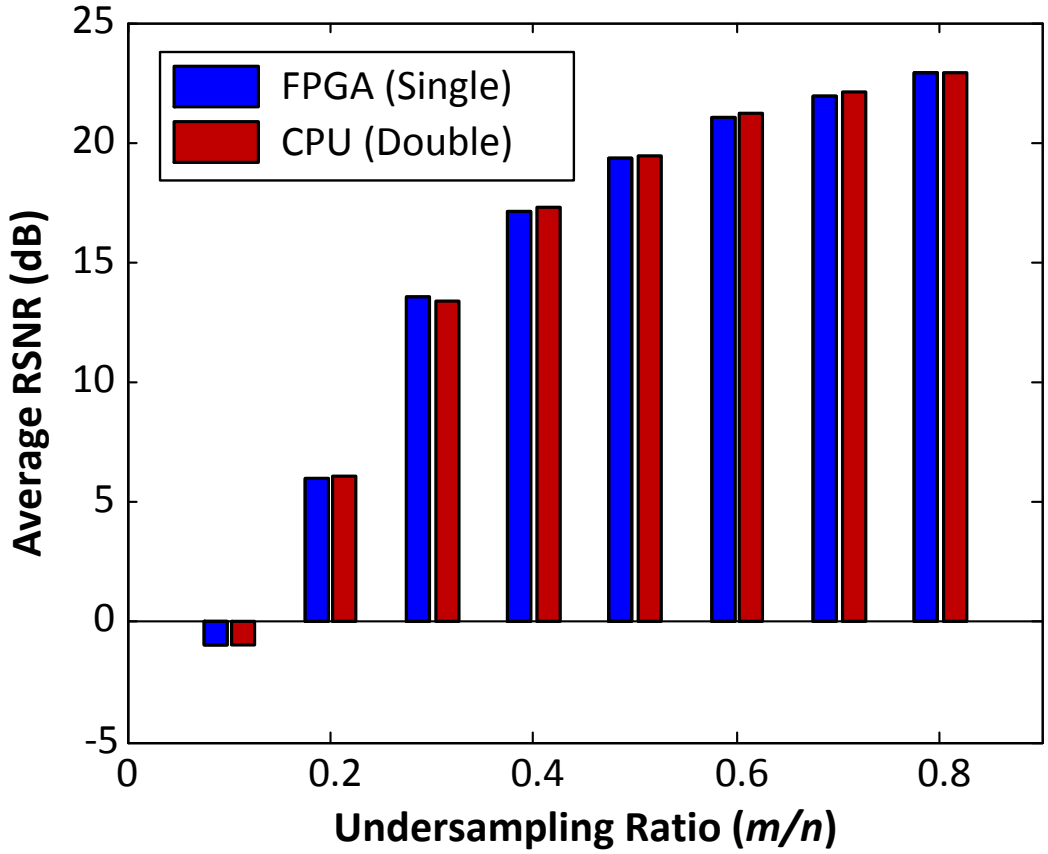



Figure 6.9: Average RSNR performance measured from the ECG reconstruction in C program and on the FPGA at different undersampling ratio ($m/n$).

Figure 6.9 shows the average RSNR performance measured from the double-precision C program and the FPGA reconstruction. As a floating-point data format is used, the SA engine on the FPGA is able to achieve the same level of accuracy as the software solver running on the CPU. As the undersampling ratio increases from 0.2 to 0.3, the RSNR performance is improved by over 2×. Beyond that, the RSNR gradually saturates to around 23 dB as $m$ increases. On average, a RSNR of 15 dB can be achieved at the undersampling ratio of 0.3 in the ECG reconstruction test.

### 6.4.3 Performance Benchmarking

In this benchmarking study, we compare the performance of the FPGA reconstruction with its software counterpart. To simplify the comparison, we use ideally sparse signals in the reconstruction tests. First, $k$-sparse signals $x \in S_k^n$ are generated with different problem size $n$ and sparsity level $k$. Specifically, for each $n$ and $k$, 1000 signals are randomly generated based on a Normal distribution $\mathcal{N}(0, 1)$. Note that the index of the nonzero elements in each signal is also randomly selected. The generated signals are sampled by Gaussian random matrices $\boldsymbol{A} \in \mathbb{R}^{m \times n}$, where an empirical value of $C = 1.8$ is used for evaluating the required measurement size $m$ according to (2.18). Then, the sparse signals are directly recovered using the SA engine on the FPGA and the C program running on a 2.4 GHz Intel Core i7-4700MQ mobile processor, respectively.

Figure 6.10 presents the averaged FPGA reconstruction time in comparison to the averaged CPU run time. Note that the performance of the software solver does not scale as well as the FPGA implementation. In general-purpose computing platforms, accessing data from the main memory has a relatively long latency in the case of cache misses. Such long latency will become a dominating factor on the overall performance when the queue of data access is short. Consequently, as $n$ decreases, the reduction in CPU run time slows down gradually. As shown in Fig. 6.10 (a), the top curve starts to flatten out as $n \leq 800$. In contrast, the SA engine on the FPGA has a better data locality since all the data memories can be accessed locally. As a result, the memory access latency is minimized regardless of the access pattern. As shown in Fig. 6.10, the FPGA reconstruction time drops at a constant rate as $n$ decreases.

Note that more iterations of the OMP algorithm need to be performed for recovering $x \in S_k^n$ with higher $k$. In addition, the computational complexity of the LS task increases quadratically with $k$ (Table 4.3). Figure 6.10 shows that as $k$ increases, the FPGA acceleration becomes less effective when $n$ is small, but more effective when $n$ is large. For $n = 100$, an average speed-up of 71, 51, and 47 times can be achieved at the $k/n$ ratio of 0.1, 0.2, and 0.3, respectively. When $n = 1600$, the corresponding speed-up is 48, 76, and

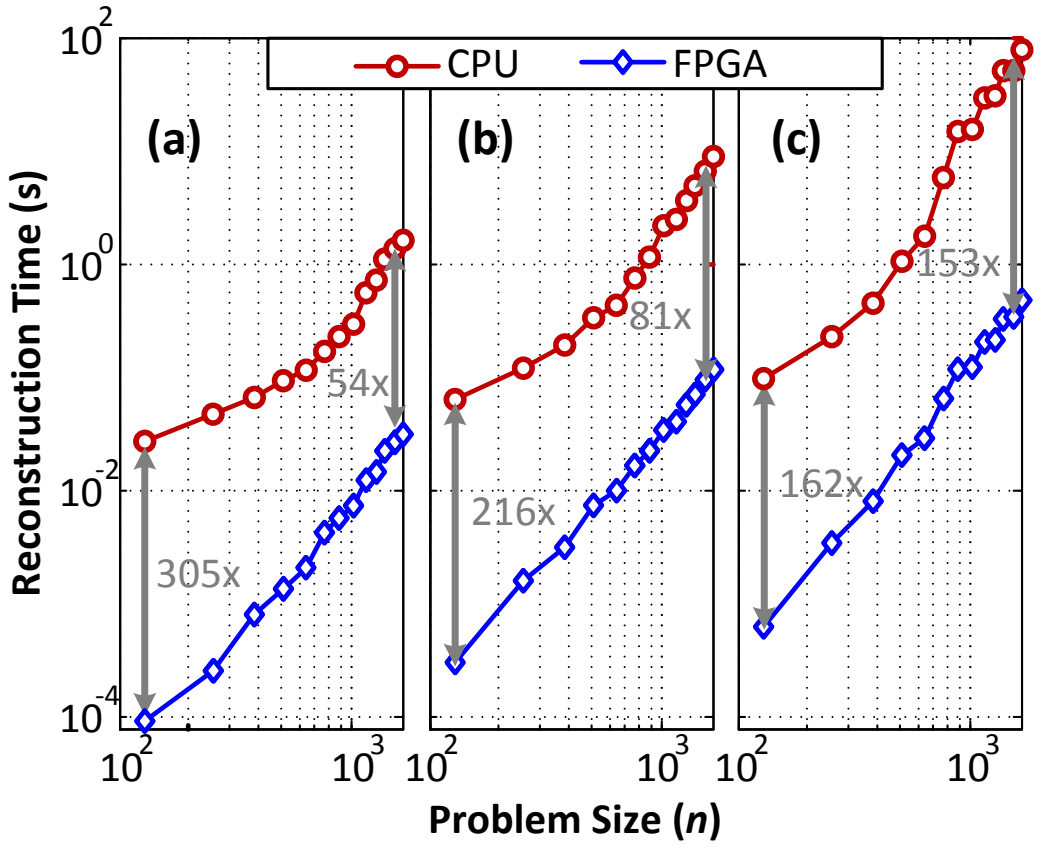



Figure 6.10: Averaged FPGA reconstruction time versus CPU run time measured from the experiments at different problem size $n$. The reconstructed signal has a sparsity ratio of (a) $k/n = 0.1$, (b) $k/n = 0.2$, (c) $k/n = 0.3$.

147 times, respectively. The difference comes from how the memory access latency affects the overall performance. When $k$ is small, the complexity of the LS task is limited. The memory access latency affects the overall performance as a fixed overhead. Therefore, it becomes less significant as the complexity of the LS task increases. Differently, when $k$ is large, the memory access latency expands the execution time as a scaling factor. Due to the loop-carried data dependency of the FS and BS computation, the total execution time of the LS task is proportional to the loop latency. In the CPU systems, the loop structure is realized by reading from and writing back to the memory. Consequently, the memory access latency becomes the bottleneck of the loop latency. Alternatively, the SA engine has a low-latency loop structure as shown in Fig. 5.11 when computing the FS and BS, where

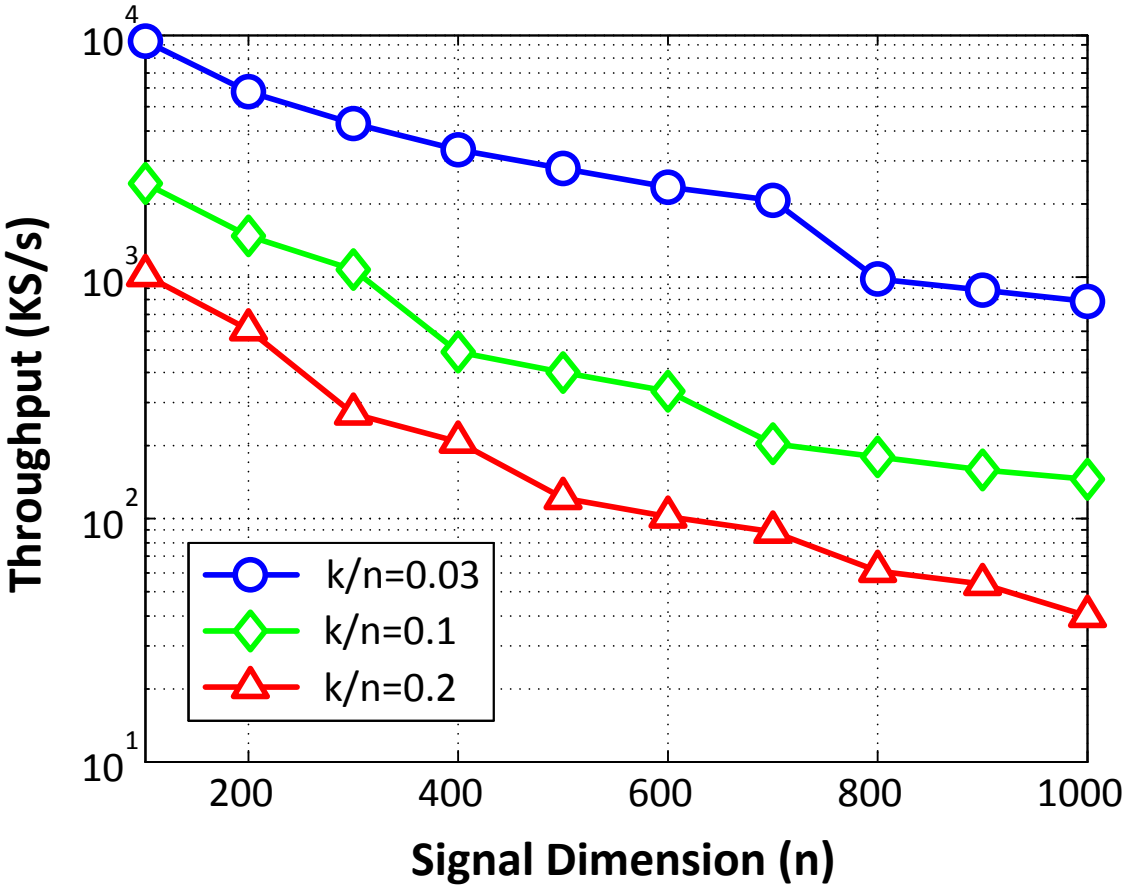



Figure 6.11: Reconstruction throughput of the FPGA implementation.

no memory access is involved. Therefore, the FPGA acceleration becomes more effective as the size of the LS task increases at large $k$.

Figure 6.11 shows the measured reconstruction throughput of the FPGA implementation. Operating at 53.7 MHz with 128 PEs in parallel, the SA engine achieves the reconstruction throughput of 796–9,368, 144–2,418, and 40–1,021 KS/s for the signal dimension ($n$) range of 100–1000 at the high ($k/n = 0.03$), medium($k/n = 0.1$), and low ($k/n = 0.2$) signal sparsity ratio, respectively.

# CHAPTER 7

# A SA Engine Chip for Mobile ExG Data Aggregation

Compressive sensing (CS) is a promising solution for low-power on-body sensors for 24/7 wireless health monitoring [Ca12]. In such application, a mobile data aggregator performing real-time signal reconstruction is desired for timely prediction and proactive prevention. However, CS reconstruction requires solving a sparse approximation (SA) problem. Its high computational complexity makes software solvers, consuming 2–50 W on CPUs, very energy-inefficient for real-time processing. This chapter presents a 12-to-237 KS/s 12.8 mW SA engine chip integrated in 5.13 $mm^2$ in 40-nm CMOS for energy-efficient mobile data aggregation from compressively sampled biomedical signals. By using configurable architecture, a 100% utilization of computing resources is achieved. An efficient data shuffling scheme is implemented to reduce memory leakage by 40%. At the minimum energy point (MEP), the SA engine chip achieves a real-time throughput for reconstructing 61–237 channels of ECG, EMG, and EEG (collectively referred to as ExG) signals simultaneously with $< 1\%$ of a mobile device's 2W power budget, which is 76–350$\times$ more energy-efficient than prior hardware designs.

## 7.1 Chip Design

For achieving the best quality of results for ExG signal reconstructions, the SA engine must be able to handle (1) a large dynamic range, (2) configurable problem setting at run time, and (3) reconstructions on different basis. In addition, to support the real-time reconstruction of multi-channel ExG data on a mobile platform without moving its energy needle, the SA engine must meet the design specification of achieving a $> 50$ KS/s throughput with a power

budget of $< 20$ mW ($< 1\%$ of a mobile device's 2W power budget). Taking advantage of the compile-time scalability and run-time configurability of our soft-IP design, a SA engine chip that meets the above-mentioned requirements and specifications is realized in a 40-nm 1P8M CMOS process.

The RTL codes of SA engine are compiled with the following parameter settings (see Table 5.1):

- $\{W_M = 8,\ W_E = 23\}$:
  The single-precision data format is used to support a large dynamic range.

- $\{S_{ADD} = 1, S_{MULT} = 2, S_{DIV} = 5, S_{CMP} = 1\}$:
  A short pipeline stage that meets the timing specification is used to limit the loop-latency of the SA engine (see Fig. 5.11) to 4–8 cycles, greatly accelerating the iterative FS/BS computation.

- $P = 128$:
  A PE parallelism of 128× allows the SA engine to operate at a scaled supply voltage and frequency for additional energy efficiency gain while maintaining the target throughput.

- $N = 1024$:
  A signal dimension of up to 1024 is supported.

- $M = 512$:
  A measurement dimension of up to 512 (undersampling ratio $\geq 0.5$) is supported.

- $K = 192$:
  A signal sparsity level of up to 192 (sparsity ratio $\geq 0.1875$) is supported.

The compiled RTL codes are synthesized in Synopsys Design Compiler using a standard-cell based design flow. To achieve the target throughput, a setup time of 60 ns (16.7 MHz) evaluated at the worst case process, voltage, and temperature (PVT) corner is used throughout the chip implementation. Taking into account the interconnect overhead incurred by the subsequent physical design, a 22% timing slack is used during the synthesis.

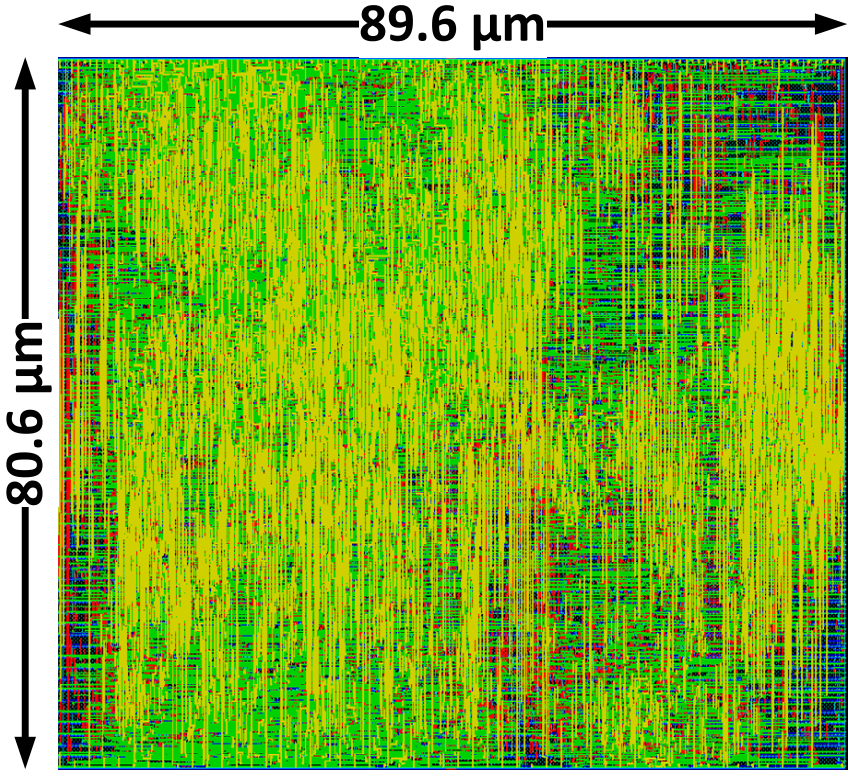



Figure 7.1: Layout view of the PE block.

Specifically, the SA engine is synthesized at $16.7/(1-0.22) = 21.4$ MHz. To reduce the leakage power, the design is first synthesized using high-threshold (HVT) standard cells only. Then, standard-threshold (SVT) standard cells are selectively inserted to the critical paths for timing improvement. The PE, SC, and shared cache in the SA engine has a memory size of 1.2 KB, 1.5 KB, and 768 B, respectively. Note that there are total 128 instances of the PE caches in the design. To save area cost, the PE caches are realized using dual-port SRAM hard macros. Differently, as only a single instance the SC and shared caches is needed, these two data memories are realized using synthesized RAMs, which can be flattened during the physical design to facilitate the floorplaning. For voltage scaling purposes, the SA engine is split into two voltage 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.

The physical design of the SA engine is performed in Cadence Encounter. To reduce the

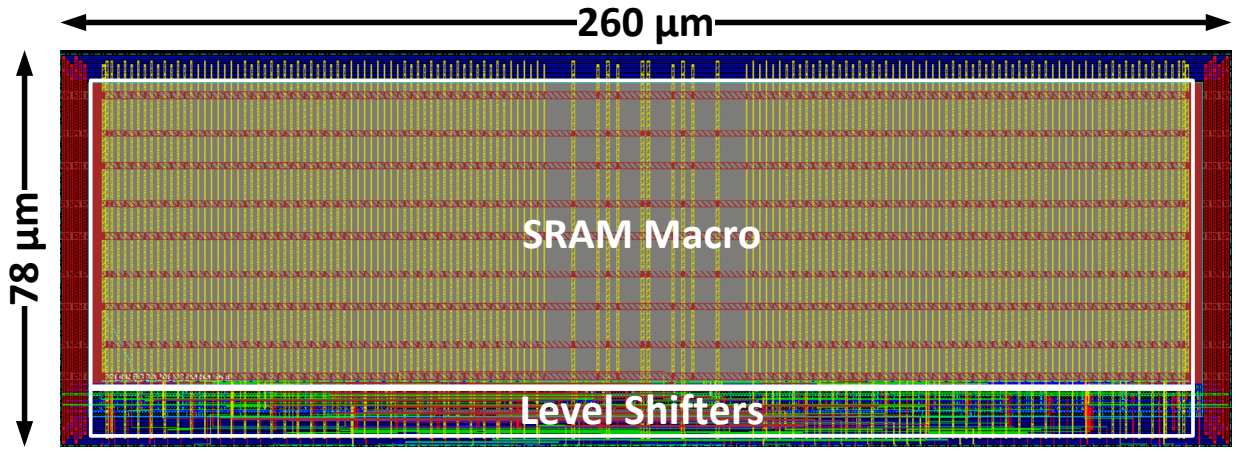



Figure 7.2: Layout view of the PE cache block.

run time, a bottom-up hierarchical design method is adopted. Specifically, the PE and PE cache are first placed and routed separately as sub-blocks at the bottom level. Then, these two blocks are treated as hard macros, and the whole design is flattened and then placed and routed at the top level. Figure 7.1 shows the layout of the PE block. Note that the PE block is routed using M1 to M4 only so that the 128 instances will not block the routing channels of M5 to M8 on the top level. Overall, the placed-and-routed PE block has a macro size of 80.6×89.6 $\mu m^2$.

Figure 7.1 shows the layout of the placed-and-routed PE cache block. Note that the SRAM macro placed in the middle occupies over 90% of the core area. In order to enhance the power delivery, horizontal and vertical power rails are routed over the SRAM macro using M5 and M4, respectively. In addition, level shifters are placed at the bottom of the SRAM macro to handle the voltage transition across different voltage domains. Overall, the placed-and-routed PE cache block has a macro size of 78×260 $\mu m^2$.

The layout view of the whole SA engine chip is shown in Fig. 7.3. 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, 8 pairs of the PE group are placed evenly with a 50 $\mu m$ space in between. To simplify testing, additional memory macros are placed on the top and the bottom slide of the chip for storing the sampling matrices. To enhance power delivery,

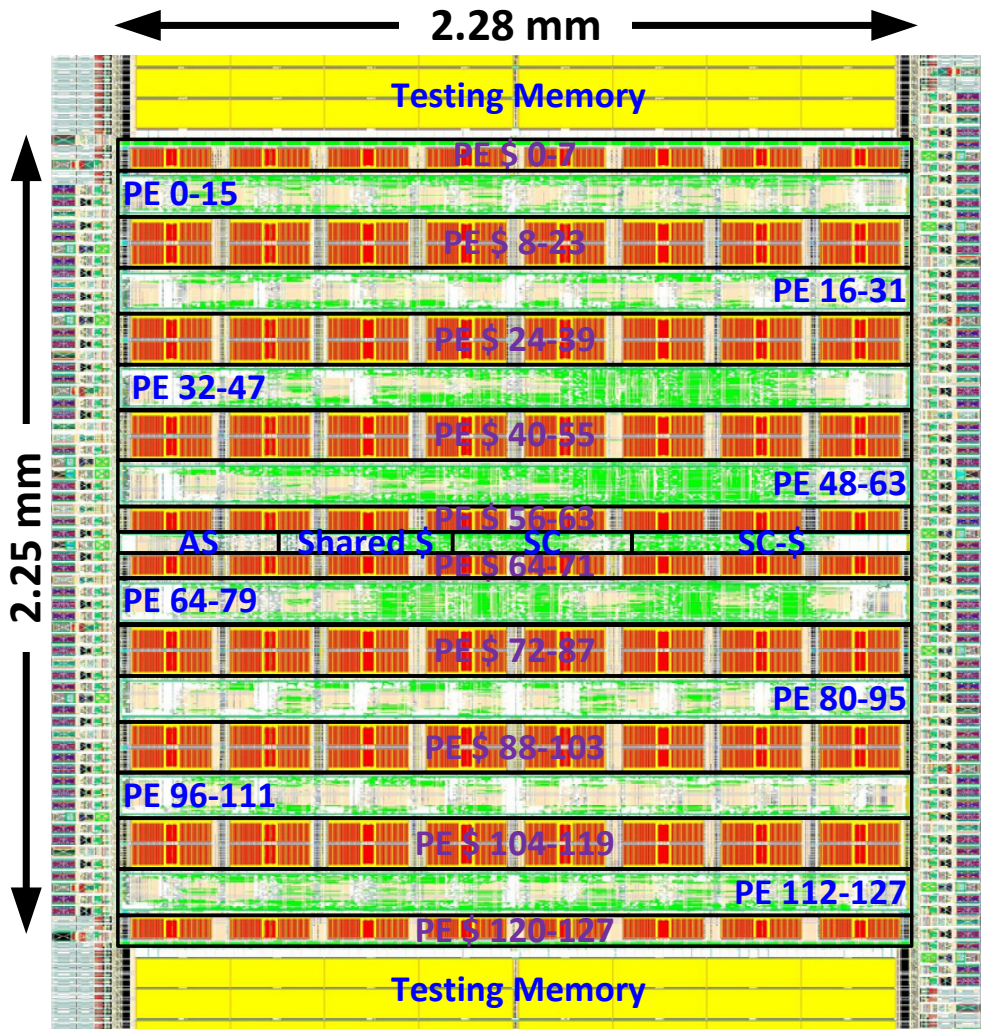



Figure 7.3: Layout view of the SA engine chip.

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 (AP) and the M8 layers that have smaller resistance and higher current density. Overall, the SA engine design occupies a core area of 2.25×2.28 $mm^2$ with an aspect ratio of 0.99.

## 7.2 Chip Testing Environment

The chip testing environment is shown in Fig. 7.4, where a Kintex-7 KC705 evaluation platform is used as the testbed. A customized printed circuit board (PCB) is designed to

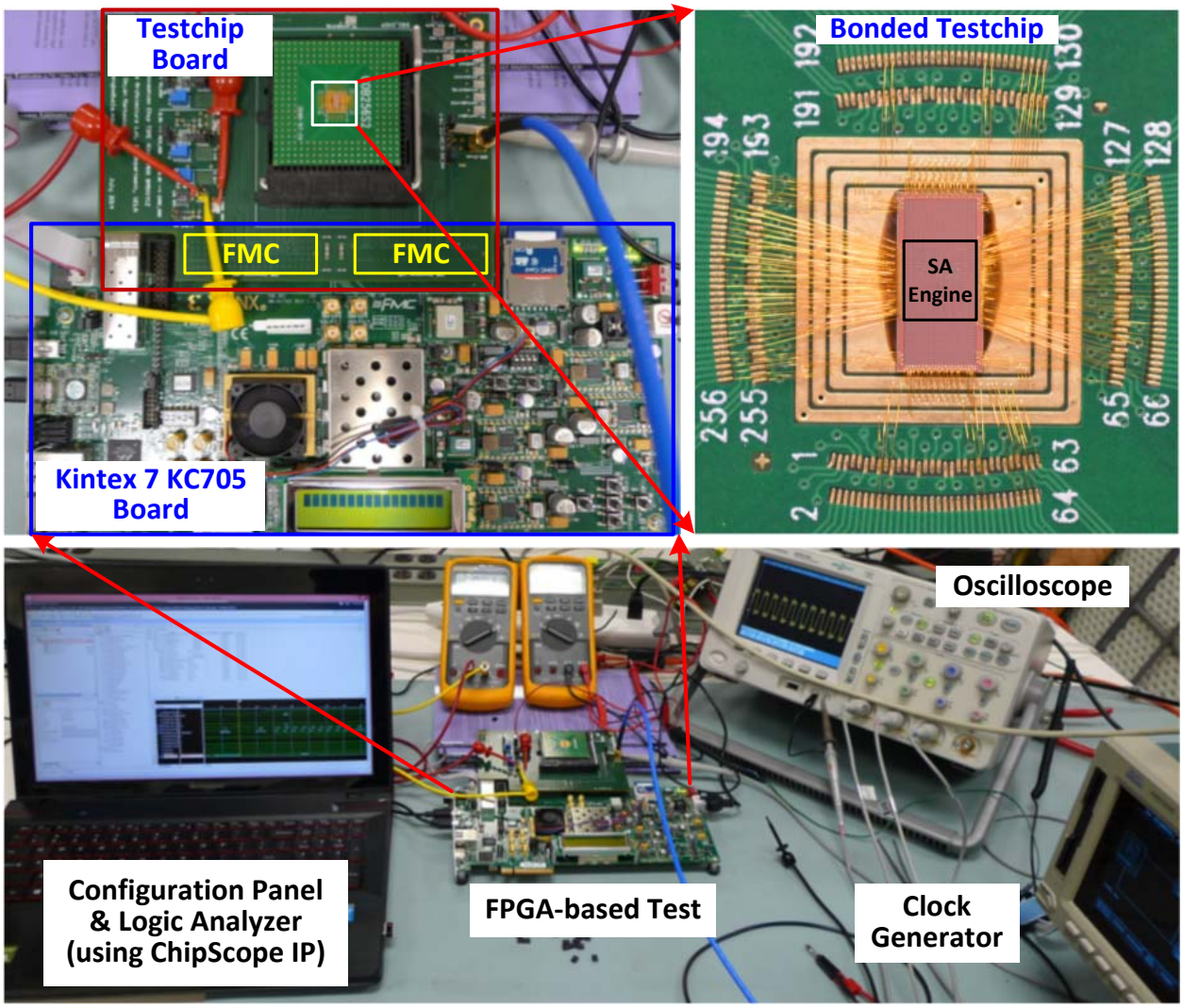



Figure 7.4: Chip testing environment.

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 the 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 a SMA connector and then passed to the FPGA board through the dedicated clock pins on the FMC connector. Note that in this FPGA-based testing environment, one is able to utilize the reconfigurability of the FPGA to map different hardware test benches for testing.

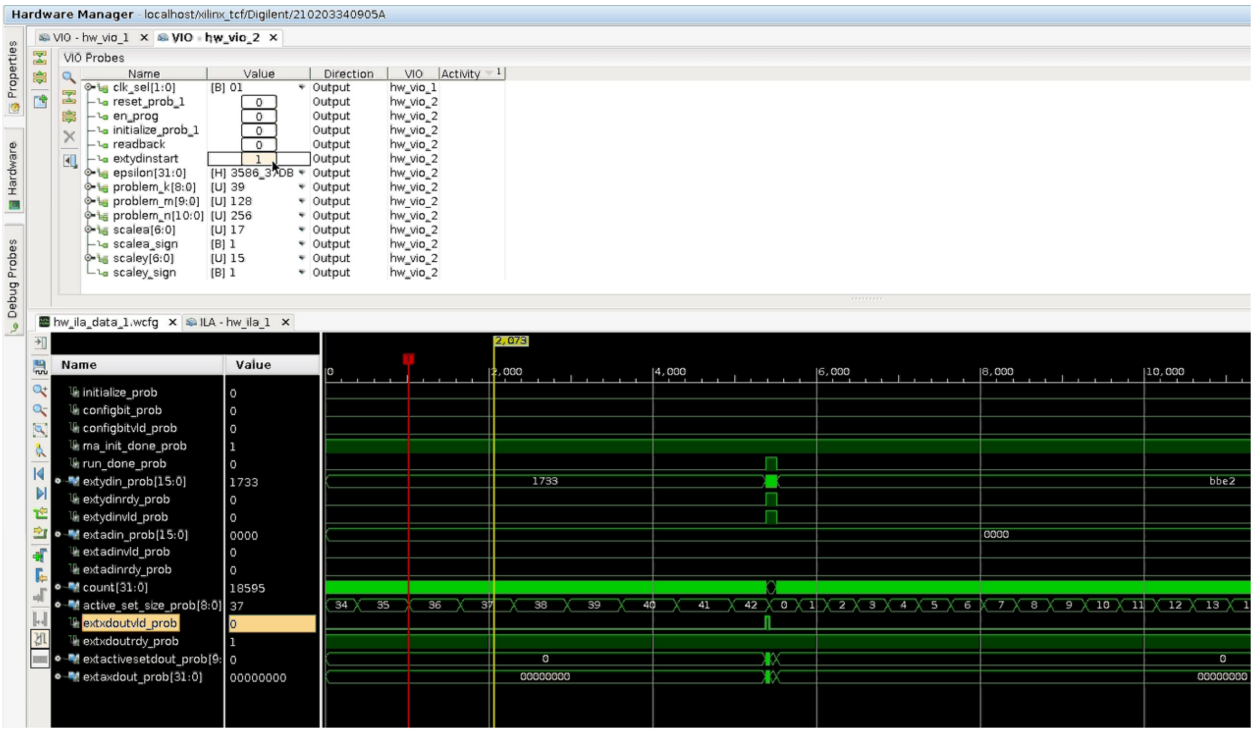
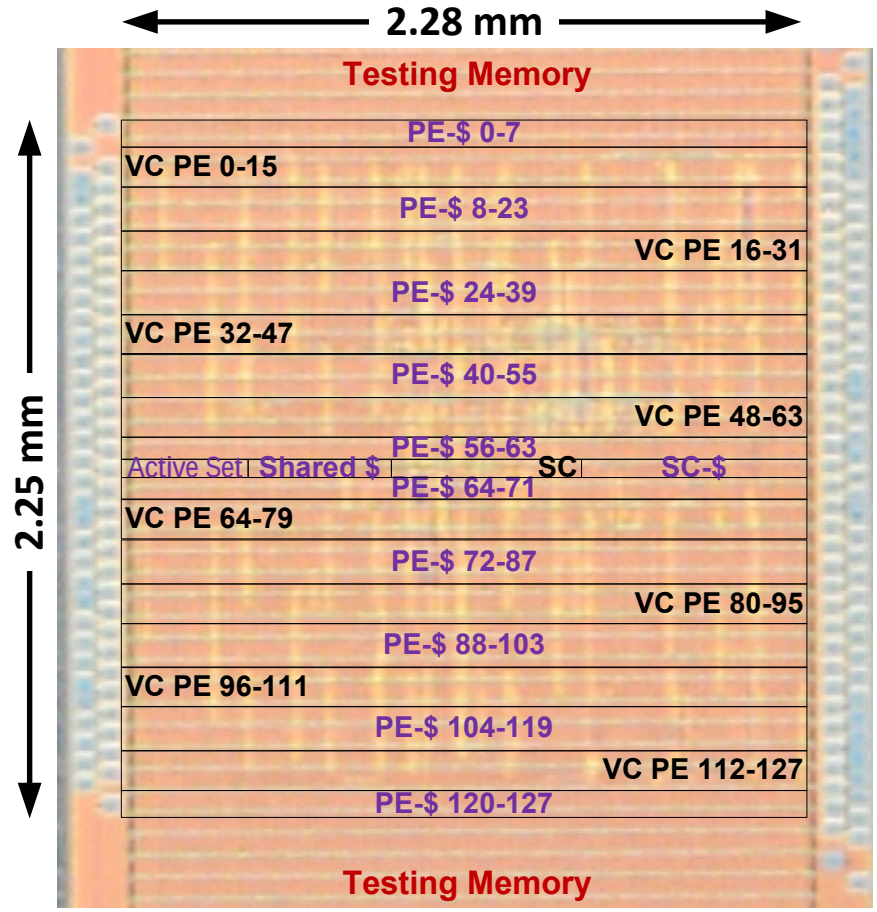



Figure 7.5: Customized control panel of the chip testing in the Xilinx Vivado Design Suite environment.

In our test bench design, the ChipScope vitual I/O (VIO) IP is used as soft registers to control both the SA engine chip and the test bench, while the ChipScope integrated logic analyzer (ILA) IP is used to probe all the I/Os of the SA engine chip. Powered by the Xilinx ChipScope IPs, the customized control panel in the Xilinx Vivado Design Suite environment is illustrated in Fig. 7.5. The top half of the panel lists the soft registers that control the testing process, which is realized by the VIO instances. The user can specify the values of these registers in real-time. Note that the value change is reflected on the FPGA at the JTAG scan rate. The bottom half of the panel shows the signal waveform probed by the ILA instances. The waveform data is sampled at the clock rate of the ILA instances, while the displayed values are updated at the JTAG scan rate.




| | |
|---|---|
| Technology | 40nm 1P8M CMOS FO4 16.3ps (TT) |
| Transistor Flavor | SVT 0.11% HVT 99.89% |
| Transistor Count | 61 M |
| I/Os | Digital 42/58 Power 156 |
| Core $V_{DD}$ | Logic: 0.5 to 1 V Mem: 0.7 to 1V |
| I/O $V_{DD}$ | 2.5 V |
| Core Size | 2.28 x 2.25 mm |

Figure 7.6: Die photo and chip summary.

## 7.3 Chip Measurement Results

The die photo of the SA engine chip is shown in Fig. 7.6. To summarize, the SA engine occupies a core area of 5.13 $mm^2$, integrated with 61 million transistors in a 40-nm 1P8M CMOS process. To reduce the leakage power, high-threshold (HVT) transistors are used in 99.89% of the logic cells. In addition, 0.11% of standard-threshold (SVT) cells are inserted into the critical paths to improve timing. The SA engine chip has a total of 42 digital inputs, 58 digital outputs, and 156 power pads supplying 3 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.

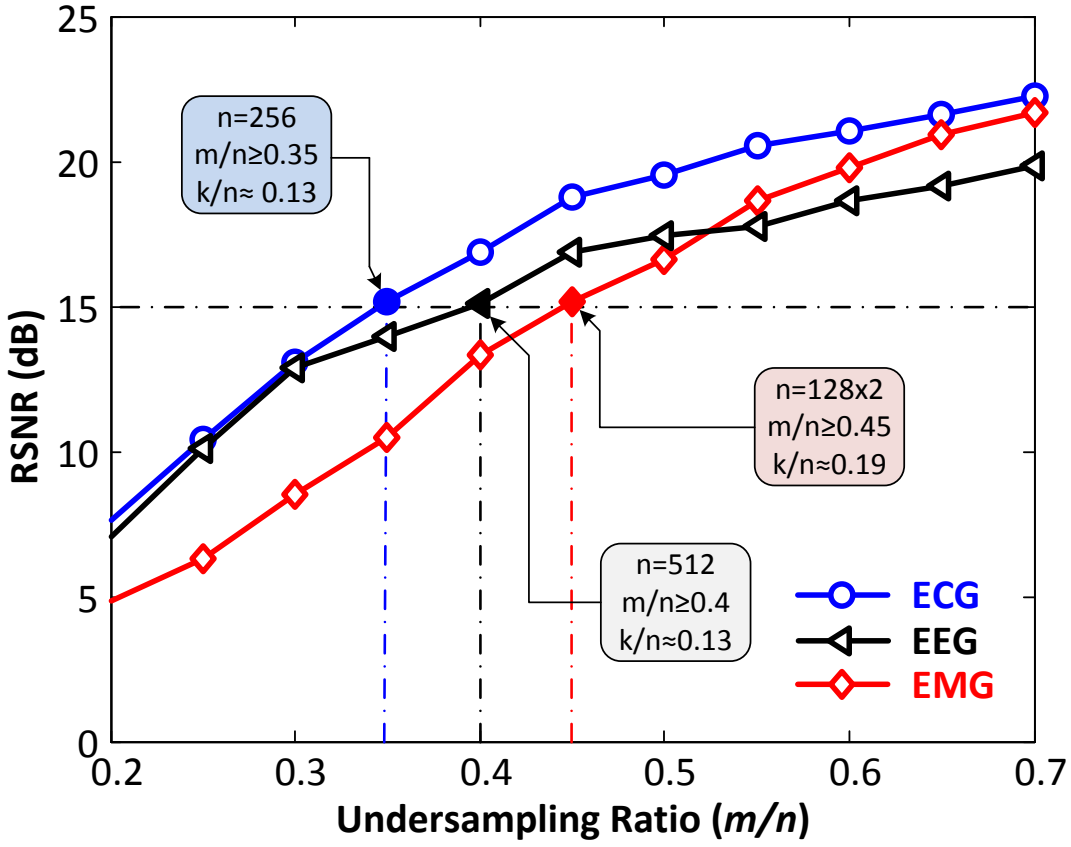



Figure 7.7: Measured RSNR performance of ECG, EMG, and EEG signals reconstructed on DWT, joint DWT-DCT, and DCT basis, respectively.

Several 1-minute recordings of real ExG signals downloaded from the PhysioBank database are used in the signal reconstruction test [Aa13]. Specifically, the digital samples of ExG signals are encoded by random Bernoulli matrices with a 5% overlapping window applied. In order to observe the raw signal sparsity, no thesholding scheme is applied in our test. The RSNR performance are measured on the SA engine chip with different problem settings.

The measured RSNR performance of ExG signal reconstruction is shown in Fig. 7.7. The best orthogonal basis for reconstructing ECG, EEG, and EMG are found to be the Haar DWT, DCT, and DWT-DCT joint basis, respectively. It is also found that the RSNR performance is sensitive to $\epsilon$. Dynamically configuring $\epsilon$ to 3–5% of the energy of each CS

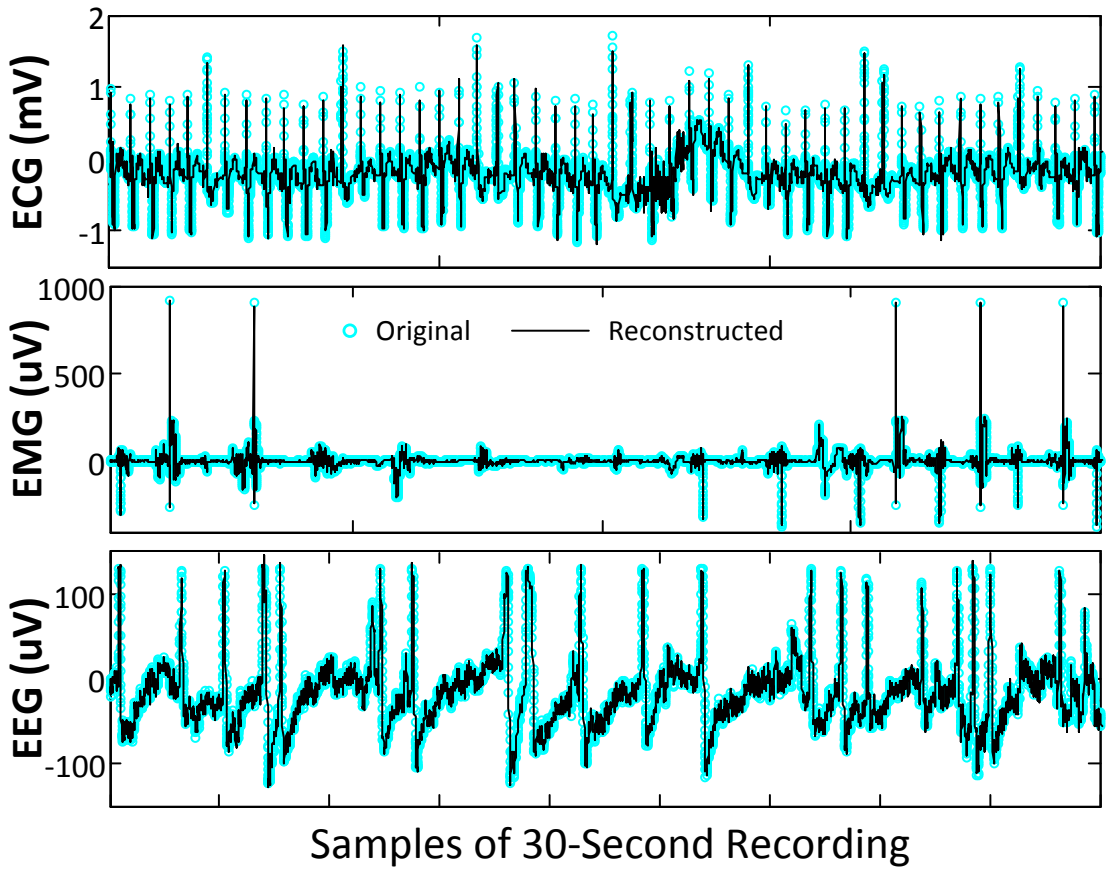



Figure 7.8: Examples of the ExG signals reconstructed on the SA engine chip with a >15 dB RSNR. The ECG, EMG, EEG signals are reconstructed on the DWT, DWT-DCT joint, and DCT basis, respectively.

sample results in the best RSNR performance. In general, higher unsersampling ratio ($m/n$) improves the RSNR performance at the cost of higher sampling rate. In addition, for the same undersampling ratio, using a higher signal dimension ($n$) in reconstruction improves RSNR slightly at the cost of lower throughput and higher energy consumption. Therefore, given a target RSNR, there exists an optimal chip setting for achieving the maximum throughput. For reconstructing the chosen 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. Figure 7.8 presents examples of the reconstructed ExG signals with a >15 dB RSNR.

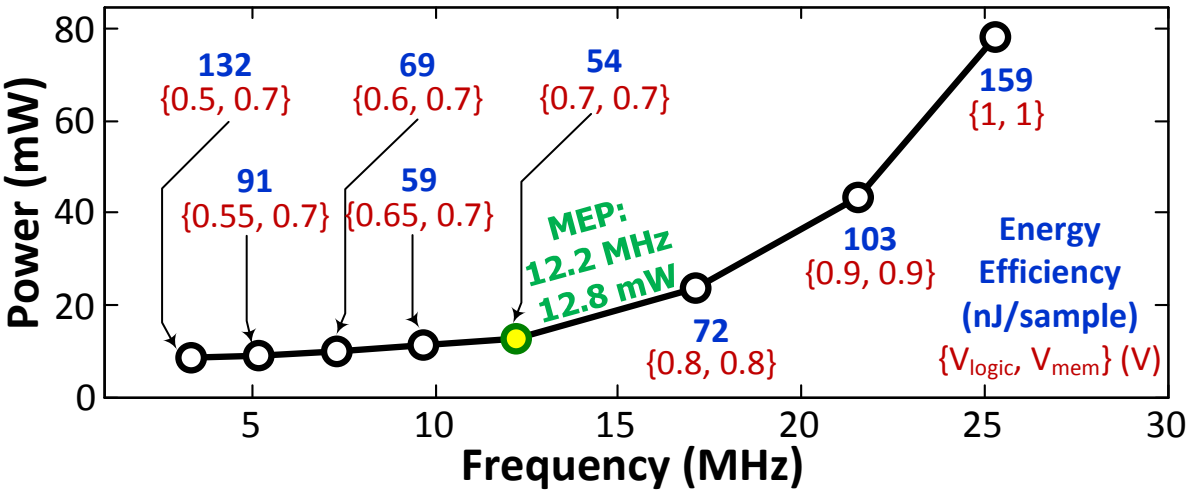



Figure 7.9: Measured power versus frequency at different $V_{DD}$ supplies.

The measured power and operating frequency of the SA engine chip at different supply voltages are shown in Fig. 7.9. The memory and logic domain 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 $V_{DD}$=0.7 V, which is the minimum supply voltage the on-chip memories can operate at. At the MEP, the chip has a operating frequency of 12.2 MHz and a power consumption of 12.8 mW. Note that the SA engine chip is a memory-bounded design, where the memory leakage power dominates the total power consumption. Therefore, lowering the logic power supply below 0.7 V will only reduce the operating frequency without affecting the power much, thereby degrading the energy efficiency. Compared to the MEP, a 2× higher operating frequency can be achieved at the cost of 6× higher power at $V_{DD}$=1 V, which corresponds to a 3× lower energy efficiency.

The measured throughput and energy efficiency of the SA engine chip when operating at the MEP for ExG signal reconstruction are summarized in Fig. 7.10. At the MEP, the SA engine 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 with a >15 dB RSNR, respectively. This throughput performance corresponds to the simultaneous reconstruction of 237, 61, and 132 channels of ECG, EMG, and EEG data, respectively. At

| at 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 |

Figure 7.10: Measured throughput and energy efficiency of the SA engine chip when operating at the MEP for ExG signal reconstruction. The highlighted numbers are the best performance for achieving a $>15$ dB RSNR.

$V_{DD}$=1 V, the maximum operating frequency of the SA engine chip is 25.3 MHz. Compared to MEP, a 2× higher throughput can be achieved at the cost of 3× lower energy efficiency.

The SA engine chip is compared to an Intel Core i7-4700MQ processor and prior chip implementations [Ma10a, Ma12] of generic SA solvers in Fig. 7.11. While the reference designs have a fixed problem setting and a limited dynamic range, our chip handles flexible problem settings on the fly and supports a large dynamic range. Overall, the SA engine chip achieves a 2× higher throughput with up to 14,100× better energy efficiency for ExG signal reconstruction than the software solver running on the CPU. For high-sparsity signal reconstruction, the SA engine is 76–350× more energy-efficient than prior hardware designs. With a <1% power budget of mobile devices, the 5.13mm$^2$ SA engine chip integrated in 40-nm CMOS can enable a 2–3× energy saving at CS-based sensor nodes while providing timely feedback and bringing signal intelligence closer to the user.

| Design | Intel i7-4700MQ | [Ma12] | | | [Ma10a] | This work |
|---|---|---|---|---|---|---|
| Technology | 22nm | 180nm | | | 65nm | 40nm |
| Target app | General | LTE channel estimation | | | Audio | Biomedical sensing |
| Algorithm | OMP/AMP | MP | GP | OMP | AMP | OMP, K-OMP |
| Signal dimension | Large | 256 | | | 512(1) | up to 1024 |
| Measurement dim. | Large | 200 | | | 512 | up to 512 |
| Sparsity level | Large | 50 | 18 | 10 | - | up to 192 |
| Core area ($mm^2$) | 174.4 | 0.73 | 1.21 | 2.42 | 0.629 | 5.13 |
| Dynamic range | High (float-pt) | Low (fix-pt) | | | Low (fix-pt) | High (float-pt) |
| 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 to 98(3) | 2 | | | 397 | 12 to 237(2) |
| Power (mW) | 47,000 | 88 | 209 | 200 | 177.5 | 8.6 to 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 40nm: Delay~1/S, Power~$1/U^2$, where S=L/40nm, U=$V_{DD}$/0.9V.
[1] The supported problem size is 1024x512, 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 measured for the same problem size (m,n,k) as in [Ma12,Ma10a].

Figure 7.11: Comparison with an Intel Core i7-4700MQ processor and state-of-the-art chip implementations of generic SA solvers.

# CHAPTER 8

# Conclusion

This dissertation presents a scalable VLSI architecture of a sparse approximation (SA) engine soft-IP core that can be implemented on FPGAs or SoCs to perform dedicated-hardware-driven SA for supporting the real-time and energy-efficient processing of compressively sampled data in compressive sensing (CS) systems. The soft-IP core supports a floating-point data format with 10 design parameters, providing the high dynamic range and flexibility for application-specific user customizations. Taking advantage of the algorithm-architecture co-optimization based upon the reformulated OMP algorithm, the VLSI architecture of the SA engine features high parallelizability, scalability, and configurability, in which all the computing resources are shared for executing the entire algorithm, leading to a 100% utilization of the computing resources and maximizing area efficiency.

The FPGA evaluation conducted on a Xilinx KC705 evaluation platform shows that our single-precision based implementation can achieve the same level of accuracy as the software solver based on double-precision C programs for the general reconstruction of compressive sampled ECG signals. Operating at the maximum frequency of 53.7 MHz with a PE parallelism of 128× (by fully utilizing the FPGA resources), the FPGA implementation of the SA engine achieves 47–147× higher throughput than the software counterpart running on an Intel Core i7-4700MQ mobile processor.

In order to enable timely prediction and proactive prevention in CS-based 24/7 wireless health monitoring systems, a 12-to-237 KS/s 12.8 mW SA engine is implemented in 40-nm CMOS technology for the mobile data aggregation of compressively sampled biomedical signals. Overall, the SA engine chip achieves a 2× higher throughput with up to 14,100× better energy efficiency for ExG (ECG, EMG, EEG) signal reconstruction than an Intel

Core i7-4700MQ mobile processor. For high-sparsity signal reconstruction, the SA engine is 76–350× more energy-efficient than prior hardware designs. With a $< 1\%$ power budget of mobile devices, the 5.13 $mm^2$ SA engine chip integrated in 40-nm CMOS can provide a 2–3× energy saving at CS-based sensor nodes while providing timely feedback and bringing signal intelligence closer to the user.

## 8.1 Research Contributions

The goal of this research is to provide a VLSI-based computing solution for solving general SA problems, especially for those with high or medium signal sparsity levels. The main target is to develop a VLSI architecture that maintains the scalability, configurability, and flexibility as software solvers, while offering the energy efficiency and performance gains through hardware customizations. To address this goal, this dissertation made the following key contributions.

- Conducted a benchmarking study to compare the potentials of different SA and $\ell_1$-based algorithms for efficient VLSI implementations, including IPM, OMP, Hotomopy, IST, and AMP. The benchmarking results show that OMP has the best complexity-undersampling trade-off for high- and medium-sparsity signals, while AMP is a better choice for low-sparsity signals.

- Analyzed the complexity characteristic of the OMP algorithm to inform architecture design. The LS task that has both high computational and operational complexity must be simplified at the algorithm level for efficient VLSI implementations.

- Reformulated the OMP algorithm to (1) eliminate the square-root operations, (2) reduce the computational complexity of the LS task from $\mathcal{O}(mk^3)$ to $\mathcal{O}(mk^2)$, and (3) simplify the LS task to 4 BLA operations per iteration, which enables the hardware resource sharing in the architecture design.

- Proposed a hierarchical AS method that significantly reduces the computational complexity of the AS task through a coarse-grain searching with high-level quantization,

while maintaining the result accuracy through another round of fine-grain searching with a reduced search space.

- Designed a scalable VLSI architecture that efficiently maps the reformulated OMP algorithm, where all the computing resources are shared for executing the AS, LS, and EU tasks, resulting in a 100% of utilization of the computing resources.

- Proposed two different memory mapping schemes for handling the Cholesky factorization. The mirror-mode-based scheme provides a great scalability for the soft-IP core, as it allows large matrices to fold into smaller square matrices easily. Differently, the shuffle-mode-based scheme is more efficient in utilizing the memory space. In comparison to the mirror mode, the shuffle mode offers 2× memory size reduction, leading to a 40% total power reduction for the chip implementation in 40-nm CMOS.

- Developed a soft-IP core based on the proposed VLSI architecture in Verilog-HDL, which supports a floating-point data format with 10 design parameters, providing the high dynamic range and flexibility for application-specific user customizations. The soft-IP core can be implemented on FPGAs or SoCs to perform dedicated-hardware-driven SA to support the real-time and energy-efficient processing of compressively sampled data in CS systems.

- Evaluated the scalability and performance of the developed soft-IP core on a Xilinx KC705 evaluation platform. Compared to prior FPGA design, our design can achieve up to 30% higher throughput while offering a larger dynamic range capability and better design flexibility. The benchmarking results show that the SA engine implemented on the FPGA achieves the same level of accuracy as the double-precision C programs running on an Intel Core i7-4700MQ mobile processor, while providing 47–147× higher throughput.

- Prototyped a configurable 12-to-237 KS/s 12.8 mW SA engine chip in 40-nm CMOS for mobile data aggregation of compressively sampled biomedical signals, which enables the timely feedback and brings signal intelligence closer to the user in CS-based 24/7

wireless health monitoring systems.

- Demonstrated the real-time throughput for reconstructing 61–237 channels of ExG signals simultaneously with <1% of a mobile device's 2W power budget, which is 14,100× and 76–350× more energy-efficient than the CPU and prior hardware designs, respectively.

## 8.2 Future Work

Stepping upon the results of this research, it will be interesting to investigate the cost of supporting multiple SA or $\ell_1$ based algorithms using a flexible VLSI architecture. The key for achieving a better efficiency-flexility trade-off is the algorithm-architecture co-design rather than optimizing either domain separately. For such a flexible VLSI architecture, an adaptive configuration method should be also developed. The ultimate goal is to adaptively configure the chip to choose the optimal algorithm for archiving the best performance at all times. For instance, a possible scheme is to check the sparsity level of recovered signals periodically to determine the algorithm. By arbitrating against a threshold, OMP and AMP can be deployed when low- and high-sparsity signals are identified, respectively. Such a simple scheme could limit the number of iterations of the executed algorithm, thereby maintaining a high throughput of the recovery regardless of the signal types.

The hardware mapping of complex algorithms is an important research area as such hardware customizations could offer unprecedented application opportunities on two aspects. First, a typical performance boost on the order of 10-100× often enables the real-time processing capability. Second, an energy efficiency gain on the order of 1,000-10,000× are usually achievable, which makes it possible to deploy the whole processing system onto mobile and wearable platforms. Taking these merits into account, a few possible directions for applying smart VLSI customizations are as follows. Future research is needed on building specialized accelerators and kernels for the most commonly used algorithms and computing routines in big data analysis and cloud computing. According to the statistics from the Department of Energy, almost 50% of the energy consumption in data centers are from

the computing operations of server systems. Similar to the dark silicon in smart phone SoCs, these specialized design will be able to offload a significant portion of the computing tasks from general purpose processors and execute them with 3–4 orders of magnitude better energy efficiency. It is of great interest to design domain-specific processors for optimization algorithms. Similar to graphics processing units (GPUs) in computer systems, optimization processing units (OPUs) can be built to serve as the programmable accelerators for speeding-up various real-time optimization tasks in cyber-physical systems. Exciting research opportunities also exist in building application-specific instruction-set processors (ASIPs) for driving deep learning algorithms in real-time applications.

# References


[Aa13] Goldberger A. L. and et al. "PhysioBank, PhysioToolkit, and PhysioNet: Components of a New Research Resource for Complex Physiologic Signals." *Circulation*, **101**(23):e215–e220, 2000 (June 13). Circulation Electronic Pages: http://circ.ahajournals.org/cgi/content/full/101/23/e215 PMID:1085218; doi: 10.1161/01.CIR.101.23.e215.

[Ba04] S. Boyd and et al. *Convex Optimization*, chapter 11. Cambridge University Press, Nov. 2004.

[Ba08] T. Blumensath and et al. "Iterative Thresholding for Sparse Approximations." *Journal of Fourier Analysis and Applications*, **14**(5):629–654, Dec. 2008.

[Ba12] L. Bai and et al. "High-Speed Compressed Sensing Reconstruction on FPGA Using OMP and AMP." In *Proc. 19th Int. Conf. Electronics, Circuits and Systems (ICECS)*, pp. 53–56, Seville, Spain, 2012.

[Bis06] C. Bishop. *Pattern Recognition and Machine Learning*. Springer, New York, NY, 2006.

[Ca98] S. Chen and et al. "Atomic decomposition by basis pursuit." *SIAM J. Sci. Comp.*, **20**(1):33–61, 1998.

[Ca05] E. Candès and et al. "Decoding by Linear Programming." *IEEE Trans. Inf. Theory*, **51**(12):4203C4215, Dec. 2005.

[Ca06] E. Candès and et al. "Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information." *IEEE Transactions on Information Theory*, **52**(2):489 – 509, Feb. 2006.

[Ca08] E. Candès and et al. "An Introduction to Compressive Sampling." *IEEE Signal Process. Mag.*, **25**(2):21–30, Mar. 2008.

[Ca10] J.-F. Cai and et al. "Split Bregman methods and frame based image restoration." *Multi-scale Modeling and Simulation*, **8**(2):337C369, Jan. 2010.

[Ca12] F. Chen and et al. "Design and Analysis of a Hardware-Efficient Compressed Sensing Architecture for Data Compression in Wireless Sensors." *IEEE Journal of Solid-State Circuits*, **47**(3):744–756, Mar. 2012.

[Ca13] G. Caballero and et al. "Paradigm Free Mapping with Sparse Regression Automatically Detects Single-Trial Functional Magnetic Resonance Imaging Blood Oxygenation Level Dependent Responses." *Human Brain Mapping*, **34**(3):501–518, Mar. 2013.

[Da04] A. Delorme and et al. "Interaction of Bottom-Up and Top-Down Processing in The Fast Visual Analysis of Natural Scenes." *Cognitive Brain Research*, **19**(2):103–113, 2004.

[Da06] D. Donoho and et al. "Fast Solution of L1-Norm Minimization Problems When The Solution May Be Sparse." Tech. rep., Inst. Comput. Math. Eng., Stanford Univ., Stanford, CA, 2006. Available: http://www.stanford.edu/~tsaig.

[Da08] M. Duarte and et al. "Single Pixel Imaging via Compressive Sampling." *IEEE Signal Processing Magazine*, **25**(2):83–91, Mar. 2008.

[Da09] D. Donoho and et al. "Message-passing algorithms for compressed sensing." *Proceedings of the National Academy of Sciences of the United States of America*, **106**(45):18914C18919, Sep. 2009.

[Da12] A. Dixon and et al. "Compressed Sensing System Considerations for ECG and EMG Wireless Biosensors." *IEEE Trans. Biomed. Circuits Syst.*, **6**(2):156–166, Apr. 2012.

[Don06] D. Donoho. "Compressed Sensing." *IEEE Transactions on Information Theory*, **52**(4):1289–1306, Apr. 2006.

[E 08] et al. E. Candès. "Highly Robust Error Correction by Convex Programming." *IEEE Trans. on Information Theory*, **50**(7):2829–2840, Jul. 2008.

[Ea04] B. Efron and et al. "Least Angle Regression." *The Annals of Statistics*, **32**(2):407–499, 2004.

[Fa09] M.-J. Fadili and et al. "Image Decomposition and Separation Using Sparse Representations: An Overview." *Proceedings of the IEEE*, **98**(6):983–994, Sep. 2009.

[Ga11] C. Gaudes and et al. "Detection and Characterization of Single-Trial fMRI Bold Responses: Paradigm Free Mapping." *Human Brain Mapping*, **32**(9):1400–1418, Sep. 2011.

[Ka11] K. Karim and et al. "A Real-Time Compressed Sensing-Based Personal Electrocardiogram Monitoring System." In *Proceedings of the IEEE/ACM 2011 Design, Automation and Test in Europe Conference*, pp. 824–829. IEEE/ACM, 2011.

[Kut12] Gitta Kutyniok. *Compressed Sensing: Theory and Applications*, chapter 11, pp. 485–514. Cambridge University Press, Nov. 2012.

[La07] J. Lustig and et al. "Sparse MRI: The Application of Compressed Sensing for Rapid MR Imaging." *Magnetic Resonance in Medicine*, **58**(6):1182–1195, Dec. 2007.

[Ma01] G. Moody and et al. "The impact of the MIT-BIH Arrhythmia Database." *IEEE Eng. in Med. and Biol.*, **20**(3):45–50, May. 2001.

[Ma09] J. Ma and et al. "Deblurring from Highly Incomplete Measurements for Remote Sensing." *IEEE Trans. Geosci. Remote Sens.*, **47**(3):792C802, Mar. 2009.

[Ma10a] P. Maechler and et al. "Implementation of Greedy Algorithms for LTE Sparse Channel Estimation." In *Asilomar Conference on Signals, Systems and Computers*, pp. 400–405, 2010.

[Ma10b] A. Maleki and et al. "Optimally Tuned Iterative Reconstruction Algorithms for Compressed Sensing." *IEEE Journal of Selected Topics in Signal Processing*, **4**(2):330–341, Feb. 2010.

[Ma12] P. Maechler and et al. "VLSI Design of Approximate Message Passing for Signal Restoration and Compressive Sensing." *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, **2**(3):579–590, Sep. 2012.

[Nat95] B. Natarajan. "Sparse Approximate Solutions to Linear Systems." *SIAM J. Comput.*, **24**(2):227C234, May 1995.

[Pa11] J. Pillai and et al. "Secure and Robust Iris Recognition Using Random Projections and Sparse Representations." *IEEE Trans. Pattern Anal. Mach. Intell.*, **33**(8):1877–1893, Sep. 2011.

[Ran11] S. Rangan. "Generalized approximate message passing for estimation with random linear mixing." In *IEEE International Symposium onInformation Theory Proceedings (ISIT)*, pp. 2168–2172, St. Petersburg, July 2011.

[Rom08] J. Romberg. "Imaging via Compressive Sampling." *IEEE Signal Processing Magazine*, **25**(2):14–20, Mar. 2008.

[Sa05] J.-L. Strack and et al. "Image decomposition via the combination of sparse representations and variational approach." *IEEE Trans. Image Process.*, **14**(10):1570C1582, Oct. 2005.

[Sa10] A. Septimus and et al. "Compressive Sampling Hardware Reconstruction." In *Proc. Int. Symp. Circuits and Systems (ISCAS)*, pp. 3316–3319, Paris, Fance, 2010.

[Sa11] P. Sen and et al. "Compressive Rendering: A Rendering Application of Compressed Sensing." *IEEE Trans. Vis. Comput. Graphics*, **17**(4):487–499, Apr. 2011.

[Sa12] J. Stanislaus and et al. "High Performance Compressive Sensing Reconstruction Hardware with QRD Process." In *Proc. Int. Symp. Circuits and Systems (ISCAS)*, pp. 29–32, Seoul, Korea, 2012.

[Ta79] H. Taylor and et al. "Deconvolution with the $\ell_1$ norm." *Geophysics*, **44**(1):39–52, 1979.

[Ta07] J. Tropp and et al. "Signal Recovery from Random Measurements via Orthogonal Matching Pursuit." *IEEE Trans. Inform. Theory*, **53**(12):4655–4666, Dec 2007.

[Tib96] R. Tibshirani. "Regression shinkage and selection via the LASSO." *J. Royal Statist. Soc. B.*, **58**(1):267–288, 1996.

[Van13] L. Vandenberghe. *Applied Numerical Computing*, pp. 87–89. UCLA Academic Publishing, Los Angeles, CA, 2013.

[Var07] U. Varshney. "Pervasive Healthcare and Wireless Health Monitoring." *ACM J. Mobile Networks and Applications*, **12**(2-3):113–127, Mar. 2007.

[Wa09] J. Wright and et al. "Robust Face Recognition via Sparse Representation." *IEEE Trans. Pattern Anal. Mach. Intell.*, **31**(2):210–227, Feb. 2009.

[Xa06] P. Xu and et. al. "A Novel Method Based on Realistic Head Model for EEG Denoising." *Comput. Methods and Programs in Biomed.*, **50**(2):104–110, Mar. 2006.

[Xa12] W. Xu and et al. "Robust Human Activity and Sensor Location Co-Recognition via Sparse Signal Representation." *IEEE Trans. Biomed. Eng.*, **59**(11):3169–3176, Nov. 2012.

[Xa13] W. Xu and et al. "eCushion A Textile Pressure A Textile Pressure Sensor Array Design and Calibration for Sitting Posture Analysis." *IEEE Sensors Journal*, **13**(10):3926–3934, Oct. 2013.

[Xil12] Xilinx Inc., San Jose, CA. *Kintex-7 FPGA KC705 Evaluation Kit Getting Started Guide*, 2012.

[Yan11] D. Yang. *Turbo Bayesian Compressed Sensing*. Ph.d. dissertation, Dept. Elect. Eng., Univ. of Tennessee, Knoxville, Knoxville, TN, 2011.