EFFICIENT ALGORITHM AND VLSI ARCHITECTURES FOR COMPRESSED SENSING SIGNAL RECOVERY

A thesis submitted
in partial fulfilment for the degree of

Doctor of Philosophy

By

Thomas James Thomas
SC16D019

Department of Avionics
Indian Institute of Space Science and Technology
Thiruvananthapuram, India

December 2022

Certificate

This is to certify that the thesis titled ***Efficient algorithm and VLSI architectures for compressed sensing signal recovery*** submitted by **Thomas James Thomas**, to the Indian Institute of Space Science and Technology, Thiruvananthapuram, in partial fulfillment for the award of the degree of **Doctor of Philosophy** is a bonafide record of the original work carried out by him under my supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Approved by:

Dr. J. Sheeba Rani, Advisor
Associate Professor
Department of Avionics
Indian Institute of Space Science and Technology

Dr. Deepak Mishra
Professor and Head
Department of Avionics
Indian Institute of Space Science and Technology

Date Approved: December , 2022

Declaration

I declare that this thesis titled ***Efficient algorithm and VLSI architectures for compressed sensing signal recovery*** submitted in partial fulfillment for the award of the degree of **Doctor of Philosophy** is a record of the original work carried out by me under the supervision of Dr. J Sheeba Rani, and has not formed the basis for the award of any degree, diploma, associateship, fellowship, or other titles in this or any other Institution or University of higher learning. Keeping with the ethical practice in reporting scientific information, due acknowledgements have been made wherever the findings of others have been cited.

Place: Thiruvananthapuram
Date: December , 2022

Thomas James Thomas
SC16D019

This thesis is dedicated to my parents, family, mentors and my loving wife

ACKNOWLEDGEMENTS

"He who has begun a good work in you will complete it until the day of Jesus Christ (Philippians 1:6). I am grateful to the Almighty for giving me the knowledge, strength and privilege to undertake this research and complete it satisfactorily."

First and foremost, I would like to express my sincere gratitude to Dr. J Sheeba Rani for the opportunity to work under her supervision. Her selfless guidance, dedication, constant encouragement and moral support have enabled me to pursue my research with vigour and passion. She has always motivated me to think freely and observe the highest scientific standards during the course of my research. She has been kind, understanding and emotionally supportive during the course of my research.

I would also like to thank my doctoral committee members: Dr. Deepak Mishra, Dr. Ramarao Nidamuri, Dr. Vanidevi M. and Dr. Kailash Chandra Ray (IIT Patna) for their inspiration, critical reviews and thought-provoking comments throughout the course of my research work. I am also grateful to have worked collaboratively with Dr. Sai Subrahmanyam Gorthi (IIT Tirupati) and for his sound guidance and inspiration.

I am also immensely thankful to Dr. D Sam Dayala Dev, the Director of IIST and Dr. V. K. Dadhwal, the former Director of IIST for providing me with a wonderful opportunity to pursue my PhD. Furthermore, I would also like to deliver my kind gratitude and respect to Dr. Raju K. George, Dean R&D, Dr. Kuruvilla Joseph, Dean Student Activities, Dr. A. Chandrasekar, Dean Academic and Dr. Y.V.N. Krishna Murthy, Registrar for providing me adequate facilities and student support that enabled me to complete my PhD thesis.

I would also like to express my sincere thanks to all the faculties of Department of Avionics and staff members (technical and nontechnical), especially Mr. Nidheesh Ravi and Ms. Preetha, for their kind support during the course of the PhD thesis. This work would not have been possible without the support of staff members from the Department of Avionics, R&D Section, Student Affairs and Finance Section of IIST Trivandrum. I

really appreciate their patience and thank them all for the support. I also acknowledge the Department of Space, Government of India for providing the scholarship.

I am thankful for being able to collaborate and engage in research discussions with my colleagues Bibin, Anupama and Minha. I would also like to thank my dear friends Asif and Krishna for their years of loving friendship and constant support and all my other friends at the Indian Institute of Space Science and Technology.

I thank my parents for their encouragement, patience and support throughout my research and for helping me to believe in myself. I would also like to acknowledge my loving wife Christina for giving me the motivation to push myself beyond the limits and for believing in me.

Abstract

Compressed Sensing (CS) empowers the signal processing landscape to acquire sub-Nyquist measurements of real-world signals by exploiting their underlying sparse nature in suitable domains. Sparse recovery algorithms are integral to CS as they facilitate the reconstruction of higher dimensional signals from such compressed measurements. The recovery speed and hardware complexity has been a major focus of research in the design of such sparse recovery algorithms, with the computational effort for successful signal recovery remaining high, even for moderate sized problems. Dedicated hardware-driven sparse recovery has been pursued in the literature to offer swifter and cost-effective solutions for signal recovery compared to software implementations, but are restricted by the slow convergence or hardware complexity of the underlying algorithm. To solve this problem, the research work described in this thesis focuses on the development of a novel reconstruction algorithm to swiftly and accurately decode CS measurements. Further, this work presents the design of two distinct hardware architectures related to the goals of processing speed improvement and resource minimization respectively by employing suitable hardware-friendly optimizations on the proposed algorithm.

The preliminary work is concentrated on the design of a novel sparsity independent CS reconstruction algorithm that employs parallel index selection and regularization to curtail the number of iterations required to reconstruct the signal and thereby enhance the reconstruction speed at the algorithmic level. A restricted isometry property (RIP) based analysis is provided to guarantee the exact recovery of k-sparse signals. A rigorous experimental evaluation of the proposed algorithm is carried out in high-dimensional, sparsity blind and noisy scenarios. The proposed algorithm is found to achieve a significant speed-up with respect to the state-of-art while maintaining the reconstruction accuracy.

Subsequently, our next focus of research has been to exploit the achieved algorithmic speed-up on dedicated hardware. Complexity reduction techniques are adopted to opti-

mize the proposed algorithm with the specific goal of maximizing the reconstruction speed. The reformulated algorithm incorporates a cheaper regularization strategy and a modified Gram-Schmidt (MGS) based incremental QR decomposition (QRD) approach to augment the support set. The proposed design incorporates an iterative QRD architecture with feedback circuitry to exploit the parallelism of the multi-atom support augmentation step at increased hardware costs. Additionally, a fast inverse square root block circumvents the need for parallel divider blocks giving considerable hardware and latency savings. The design reuses the iterative QRD block to implement the interdependent computations of the algorithm by sophisticated scheduling techniques. The proposed design is found to accelerate sparse reconstructions respective to the state-of-art for similar problem sizes by a factor of 2 at least. Based on the achieved reconstruction speed for 36-sparse vectors, the proposed hardware is able to attain a processing throughput of roughly 13100 Vectors/second or equivalently 13.4 million samples per second (MSPS). The proposed implementation does not require prior knowledge of k for termination or signal estimation such that the reconstruction process can remain unmonitored for signals of varying k. It is observed that for lower sparsity levels, reconstruction speed is greatly improved without significant change in RSNR.

In order to significantly curtail hardware consumption by allowing a relaxation in reconstruction time, we subsequently focus on developing an improved version of the proposed algorithm where a gradient descent inspired least-mean-squares (LMS) approach is proposed to replace the complex least squares (LS). The proposed architecture bypasses matrix transpose requirement for parallel access to each row of the measurement sub-matrix by interleaving row write operations with the pipelined LMS process. The proposed implementation is found to curtail DSP and logic slices by $3\times$ and $2\times$ respective to the implementation in the state-of-art possessing comparable reconstruction time. An alternate row based LMS update scheme is proposed to reduce the latency cost and leads to a 28% improvement in the recovery time with a 12% degradation in the RSNR with respect to the

full LMS update.

Table of Contents

List of Tables

List of Figures

LIST OF ALGORITHMS

Chapter 1

Introduction

1.1 Motivation

The digital revolution driving the present technological age was heralded by the seminal *Shannon-Nyquist* sampling theorem [1, 2] which demonstrated the recovery of any kind of data from uniformly spaced samples acquired at a rate twice that of the highest frequency present in the data under consideration. The bulk of signal processing has hence switched from the analog to the digital domain and has been democratised by the increase in computing efficiency described by *Moore's law*. This has led to the development of robust, inexpensive and flexible acquisition and processing systems capable of realizing sophisticated functionalities.

The advent of big data and artificial intelligence has drastically increased the amount of data generated by sensors owing to the exceptionally high Nyquist rates, resulting in a data deluge. To address the challenges associated with transmitting and storing such high dimensional data, various compression schemes have been devised to represent the data with fewer dimensions in appropriate transform domains. Most natural signals can be adequately approximated by applying a threshold on their corresponding representations in a suitable domain. For instance, the image of the ISRO GSLV-MKIII spacecraft shown in Fig. 1.1 can be transformed to the wavelet domain where a major portion of the coefficients are relatively small and a simple thresholding operation can be employed to obtain a sparse representation. The practice of sampling at such high Nyquist rates and then discarding a major share of the coefficients seems wasteful of the sensing resources.

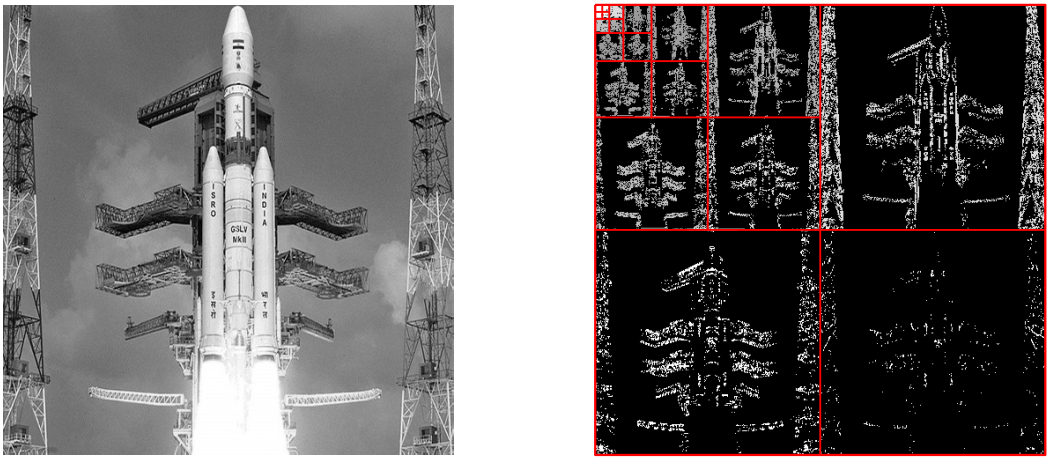


Figure 1.1: Image representation via multi-scale wavelet transform (a) Original image (b) Wavelet representation with white pixels depicting large coefficients and dark pixels depicting small coefficients

Moreover, the prospect of sampling at very high Nyquist rates for certain applications becomes too expensive or infeasible [3]. In other cases like magnetic resonance imaging (MRI), the duration of the acquisition process in order to satisfy the Nyquist rate is too long, which renders it susceptible to motion artefacts and prevents wider patient coverage.

Compressed Sensing presents a paradigm shift in signal processing that aims to *sample* and *compress* simultaneously in order to exploit the inherent redundancy in the Nyquist strategy. The goal of CS is to facilitate the precise or approximate reconstruction of high dimensional data from much fewer linearly acquired non-adaptive samples. It is well known from linear algebra that this formulation is an ill-posed problem with an infinite number of solutions. However, the sparse or compressible nature of real world signals raises the possibility of recovery from such incomplete measurements. The task of identifying the indices of the non-zero or significant components of the signal given its sparsity in a particular basis is not trivial. The basis pursuit (BP) algorithm [4] has been widely adopted to decode the compressed measurements of the CS framework.

1.2 Sparse recovery formulation

An N-dimensional signal $\boldsymbol{b}$ is said to have a sparsity level k provided it has k or fewer non-zero terms

$$\boldsymbol{b} \in \mathbb{R}^N, \quad \|\boldsymbol{b}\|_0 = |\text{supp}(\boldsymbol{b})| \leq k \ll N \tag{1.1}$$

where $|\text{supp}(\boldsymbol{b})|$ denotes the cardinality of $\boldsymbol{b}$'s support and the $\|\cdot\|_0$ quantity represents the number of non-zero components in $\boldsymbol{b}$. CS measures sparse or compressible signals as a set of non-adaptive linear samples with dimension $M < N$, where each measured sample is the inner product between the signal $\boldsymbol{b} \in \mathbb{R}^N$ and the sensing vector $\boldsymbol{\phi}_i \in \mathbb{R}^N, i = 1, \cdots, M$. The linear samples collected in this manner can be constructed as the product of an $M \times N$ sensing matrix whose rows are effectively the sensing vectors $\boldsymbol{\phi}_i$'s and the signal of interest to yield the measurement vector $\boldsymbol{y} = \boldsymbol{\Phi b} \in \mathbb{R}^M$. While reconstruction of signals from such incomplete measurements seems impossible, the l_0 minimization problem can be theoretically used to arrive at the sparsest solution satisfying $\boldsymbol{y} = \boldsymbol{\Phi b}$ as follows

$$\min_{b \in \mathbb{R}^N} \|b\|_0 \text{subject to} \boldsymbol{\Phi b} = \boldsymbol{y} \tag{1.2}$$

Though the concept of leveraging l_0 minimization theoretically works to perfectly reconstruct $\boldsymbol{b}$, solving it is computationally NP-Hard [5].

1.2.1 Recovery methods

Candes and Tao [6] demonstrated the potential of randomly encoding naturally sparse information resulting in sampling rates significantly lower than the Nyquist criterion. The seminal work in [6] demonstrated the effectiveness of a relaxed l_1 minimization to perfectly reconstruct the sparse information, as shown below

$$\min_{b \in \mathbb{R}^N} \|b\|_1 \text{subject to} \quad \boldsymbol{\Phi b} = \boldsymbol{y} \tag{1.3}$$

Over the years, there has been significant research towards practically implementing CS in various fields of applied sciences and mathematics. The random encoding ensures the preservation of the sparse information irrespective of the basis upon which it lies. Thus, the CS framework facilitates acquisitions in cost and power sensitive applications by directly sensing data in a compressed manner.

The reduction in sensing effort is compromised by the complexity of signal recovery which conventionally requires to solve optimization based basis pursuit problems to construct the sparsest vector from the randomly encoded measurements. The reconstruction task in CS involves high computational effort and memory access intensity which deems any software based approach inefficient for applications with time, cost and energy constraints. There exists a plethora of CS algorithms to reconstruct the original sparse information that vary in their hardware complexity, reconstruction speed and dependence on sparsity prior. The basis pursuit strategies based on the traditional l_1 norm minimization have proven reconstruction capabilities with theoretical guarantees and approach the optimal solution over multiple iterations. The computational effort of these methods is significantly high, and the required precision levels as well as complex structures make them infeasible for dedicated hardware implementations.

A new class of CS reconstruction algorithms termed *greedy pursuits* emerged to minimize the hardware complexity of the recovery task by iteratively approaching sub-optimal solutions. Of the existing greedy pursuits, the orthogonal matching pursuit (OMP) [7] has been widely targeted for hardware design in the literature owing to the simple regular structure of the algorithm. The state-of-art designs have investigated distinct strategies for the least squares step coupled with architectural optimizations to improve the achievable recovery speed. However, the OMP is inherently slow as it augments the support set by one column in each iteration. Sophisticated parallel greedy pursuits capable of performing multiple column updates in order to reduce the number of iterations have been proposed, but have very high associated hardware complexity.

1.3 Applications

1.3.1 Compressive Imaging

Digital image acquisition is naturally a good candidate for employing compressed sensing as the images of interest are generally sparse over some domain. The digital cameras in existence acquire images by deploying a 1:1 ratio of sensors to pixels, followed by compressing the images acquired in this way, which is definitely wasteful. Through compressive imaging, random measurements can be acquired by the sensors at a much lower sensor to pixel ratio which have been demonstrated through the design of a single pixel camera in works like [8]. It consists of a lens which focuses light on to a digital micro-mirror device (DMD) where each mirror corresponding to every pixel can be configured to contribute or not to the current measurement. The light is combined by the lens and measured by focusing onto the photon detector. This is similar to the inner product operations described earlier to yield the measurement vector $\boldsymbol{y}$ successively. The measurements obtained this way are transmitted to a dedicated system that performs the reconstruction. The advantage of the ingenious single pixel camera is that it can operate over a broader spectrum than conventional cameras built on silicon.

One of the prominent application areas of CS has been magnetic resonance imaging (MRI) where the acquisition process in the two-dimensional Fourier domain is inherently slow. While certain MR images are sparse in their original representation like angiograms, others are sparse over some other basis such as the wavelet basis or learned over-complete dictionaries [9]. The application of CS in MRI can afford to reduce the number of measurements without sacrificing the quality of the image. This in turn reduces scan times and costs by being available for a wider patient group. Further, CS can also be extended to improve the diagnostic quality of the images for the same number of measurements. There has been considerable research in the field of compressive MRI for various body imaging

modalities [10] which has led to the development of real CS based MRI machines.

1.3.2 Radar Signal Processing

Radars operate by transmitting a properly designed radar pulse and the received signal is correlated with the pulse and a time-frequency grid is constructed. Radar frameworks are ideal for CS applications as the number of targets in the discretized time-frequency grids will be small enough to employ CS techniques to recover the target scene [11, 12, 13]. Assuming there are W targets in a discretized $N \times N$ time-frequency grid with their range velocities unknown and an Alltop sequence is being sent by the transmitter, the received signal can be expressed in the form of a CS acquisition. If $W \ll N^2$, the original scene can be reconstructed using CS recovery even when targets are collocated.

Recently, super-resolution techniques have been employed in high-resolution synthetic aperture radar (SAR) and inverse SAR imaging applications to alleviate the resolution problem of traditional Fourier-based methods [13]. These techniques exploit the sparsely distributed nature of scatterers, using CS-based recovery to suppress sidelobes and noise.

1.3.3 Wireless Sensor Networks

CS can offload the local computation and sensor volume transmission burden in wireless sensor networks (WSNs) by performing random sampling at the sensor nodes [14]. The CS-WSN framework has been shown to deliver similar performance to traditional data compression mechanisms at significantly lower data volume that dominate the received signal and computations [15, 16].

Biomedical telemetry is an evolving application area that tremendously benefits from data acquisitions modelling the CS framework. The transmission energy costs in such biomedical sensor nodes are far more significant than the other functionalities enabled by the circuits. CS based data reduction strategies are found to be more efficient than data compression and filtering techniques in minimizing the energy cost of the sensor nodes

[14]. The signal structure of many biophysical signals enable the CS-based sampling at rates relative to the information content of the signals instead of the frequency contents. Another key advantage to leveraging CS-based sampling at the sensor nodes is avoiding the requirement of decision-making, instead transferring the complexity of signal recovery to centralized servers.

1.3.4 Biomedical Signal Processing

Wireless health monitoring is gaining importance in the age of internet-of-things (IoT) [17] and low power sensing devices are critical to ensuring sustainability. Real-time and continuous monitoring of the biological signals in patients with increased risk of cardiovascular diseases can aid in early detection. Wearable devices have gained recent traction for such constant monitoring scenarios, which acquire and transmit the signals for further analyses. Traditional sampling hardware significantly reduces the lifetime of such wearable devices and motivates the need for low cost acquisition devices. CS promises solutions for low power wearable devices that compressively acquire biomedical signals and perform the recovery on dedicated reconstruction engines [18, 19, 20].

1.4 Important challenges

The field of compressed sensing is still evolving with considerable research being undertaken towards three principal aspects of the CS framework. First, the design of efficient matrices that can produce the compressed measurements at low costs as well as guarantee recovery at the receiver is challenging. Second, reconstruction algorithms that can effectively decode the information at low computational costs and satisfactory recovery guarantees without requiring any prior knowledge of the nature of the signal. Third, the application of CS in a low cost, energy efficient and resource constrained regime dictates the meticulous design of hardware architectures targeting the suitable reconstruction algo-

rithms which vary in their complexity.

1.4.1 Construction of explicit matrices

Probabilistic sensing matrices like the random Gaussian, Bernoulli or partial Fourier matrices have demonstrated empirical recovery guarantees, but it is known to be computationally NP-Hard to determine if any given matrix satisfies the criteria for uniform recovery guarantees of all K-sparse signals. In recent years, much research has been involved in the design of deterministic sensing matrices that can meet the performance guarantees of random sensing matrices while also considerably reducing the underlying complexity of incoherent sampling.

1.4.2 Decoding algorithms with provable guarantees

In the compressed sensing literature, the basis pursuit (BP) algorithm which employs a relaxed l_1 minimization has garnered much attention due to its strong theoretical guarantees and stability. However, the optimization strategies used to implement the basis pursuit like linear programming are generally slow and the time complexity grows cubically in the signal dimension. Software solvers based on such methods are not time, energy and cost efficient, which highlights the need for faster algorithms. Greedy pursuits were developed with the intention to speed up the reconstruction by probing for sub-optimal solutions which often corresponds to exact or accurate versions of the signal. But many of these methods have weak theoretical performance guarantees. Advanced greedy algorithms possessing stronger recovery bounds do not work well in practical settings. This signifies a key research gap in the greedy pursuit literature that this thesis will seek to address.

1.4.3 Dedicated hardware architectures for the pursuit process

There has also been a significant effort in the direction of developing low cost and energy efficient hardware solutions in contrast to the CPU or DSP based software solvers. Despite a plethora of CS reconstruction algorithms in existence, greedy pursuits have attracted considerable attention due to their relative ease of implementation in hardware compared to optimization based basis pursuit strategies. Greedy pursuits characteristically consist of three main operations: inner product between $\mathbf{\Phi}$ and b, support set augmentation and residual update.

Orthogonal Matching Pursuit (OMP) has been prominent among the greedy pursuits because it offers a simple and regular structure from the hardware perspective. Albeit, it requires K iterations in order to arrive at the sparse solution. There have been research efforts towards accelerating these methods on CPU, GPU and DSP platforms [21, 22] achieving reconstructions in few tenths of a second for 1024-dimensional signals. OMP has also been used in retrieval mechanisms of big data analytics systems where the signal dimensions are huge [23]. Despite using GPUs to speed up the OMP, it takes about 1 second for finding the top K entries in data vectors sized tens of thousands. The need for accelerating OMP reconstruction by at least two orders of magnitude and the emergence of power-constrained CS applications in wireless medical telemetry has driven very large scale integration (VLSI) designs of OMP based on field programmable gate arrays (FPGAs) or application specific integrated circuits (ASICs). However, the serial nature of OMP prohibits further acceleration that is desired in applications like radar detection [24] where timely decisions based on the reconstructed information needs to be taken. This drawback of OMP has led to more advanced algorithms that typically require fewer iterations than the signal sparsity by employing varied support augmentation strategies. However, this significantly adds to the hardware complexity costs to realize the complex mechanisms involved. This motivates the need for designing efficient architectures in tandem with the design of fast decoding

algorithms in order to realize the higher speed-ups mandated by realistic CS applications.

1.5 Scope and objectives

The scope of the work is on the design of a novel fast decoding algorithm for CS which is also independent of the sparsity level and reconstructs signals reasonably well with respect to the state-of-art. It further covers the design of efficient architectures for the novel pursuit devised for achieving two distinct design goals. The first goal is to develop an architecture for a hardware friendly adaptation of the proposed decoding algorithm that trades off hardware resources in order to achieve significant improvement in reconstruction speed compared to the existing works. The problems tackled in this research work are the multi-atom support augmentation and residual update operations that are handled by a complicated and well-designed scheduling mechanism. The second design goal is to devise a hardware friendly adaptation that sacrifices reconstruction speed for achieving considerable reduction in hardware resources.

The objectives of the work are to realize high throughput and low complexity architectures for designing CS reconstruction engines respective to the desired design goals, which decode the compressed measurements to yield vectors with a sufficient level of accuracy.

1.6 Contributions of this thesis

The research work focuses on the development of a sparsity independent fast CS reconstruction algorithm and proposing two hardware architectures targeting high recovery speed and low resource utilization respectively. The major contributions of the work are the following

- A novel sparsity independent CS reconstruction algorithm is proposed that employs parallel index selection and regularization to curtail the number of iterations required

to reconstruct the signal and thereby enhance the reconstruction speed. A restricted isometry property (RIP) based analysis is provided to guarantee the exact recovery of k-sparse signals. A rigorous experimental evaluation of the proposed algorithm is carried out with the state-of-art.

- A novel reformulation of the SIRP algorithm from the hardware perspective, incorporating a cheaper regularization strategy and a modified Gram-Schmidt (MGS) based incremental QR decomposition (QRD) approach. The proposed design incorporates an iterative QRD architecture with feedback circuitry to exploit the parallelism of the triangularization step in MGS. Additionally, a fast inverse square block circumvents the need for parallel divider blocks giving considerable hardware and latency savings. The design reuses the iterative QRD block to implement the interdependent computations of the algorithm by sophisticated scheduling techniques.

- An improved version of the sparsity independent regularized pursuit (SIRP) is proposed that improves reconstruction speed in addition to being sparsity independent and hardware-feasible. A pipelined gradient descent inspired least-mean-squares (LMS) architecture is proposed to replace the complex least squares (LS) step in SIRP and incorporates hardware-sharing for the interdependent steps of the algorithm. Further, an alternate row LMS scheme reduces the cycles per iteration to improve reconstruction speed by trading off the reconstruction quality.

1.7 Thesis outline

This thesis investigates the compressive sensing reconstruction framework with a focus on improving the recovery speed and further striving to achieve a balance between the resource utilization and recovery speed. The remainder of the thesis is structured as follows.

Chapter 2 discusses the CS framework in depth, with specific attention to the reconstruction algorithms and the spectrum of existing hardware implementations targeting the

prominent pursuit

Chapter 3 draws attention to the formulation of a novel pursuit algorithm which employs parallel index selection and a methodical regularization strategy to curtail the number of iterations in CS reconstruction, which is discussed in depth with theoretical guarantees and rigorous experimental analyses.

Chapter 4 investigates a novel adaptation of the sparsity independent regularized pursuit favouring accelerated reconstructions. The proposed design incorporating a cheaper regularization strategy and a modified Gram-Schmidt (MGS) based incremental QR decomposition (QRD) approach is prototyped on FPGA. An ASIC design of the improved algorithm in UMC 65 nm technology is presented.

Chapter 5 investigates a distinct adaptation of the sparsity independent regularized pursuit that trades off the reconstruction speed in order to significantly curtail the hardware resources. A pipelined gradient descent inspired least-mean-squares (LMS) architecture is proposed to replace the complex least squares (LS) step in the proposed algorithm.

Chapter 6 presents the conclusions of the research work and the future perspectives.

Chapter 2

Literature review of algorithms and architectures

This chapter presents a brief introduction to the reconstruction method in compressed sensing with emphasis on greedy pursuits. The persistent challenges in the state-of-art are clearly discussed which drives us to identify potential modifications that can achieve much better tradeoffs between hardware complexity and reconstruction speed.

2.1 Preliminary

2.1.1 Notation

The conventions mentioned below apply to the notations that will be used in this thesis. An $M \times N$ matrix is denoted as an upper case bold letter (e.g. $\mathbf{\Psi}$) whereas a vector is denoted as a lower case letter (e.g. ψ). The notation ψ_i implies the i^{th} element of an arbitrary vector ψ, while its bold version $\boldsymbol{\psi_i}$ implies the i^{th} column of matrix $\mathbf{\Psi}$. A set of indices is denoted by an upper case Greek or English letter (e.g. Λ), which when used as a subscript of a matrix $\mathbf{\Psi_\Lambda}$ represents the set of columns in $\mathbf{\Psi}$ indexed by Λ. $|\Lambda|$ denotes the cardinality of the set Λ, while Λ^C refers to the complement of Λ.

2.1.2 l_p norm

Given a vector $b \in \mathbb{R}^n$ residing in an n-dimensional space, its l_2 norm is defined as

$$\|b\|_2 = (\sum_{i=1}^{n} |b(i)|^2)^{1/2} \tag{2.1}$$

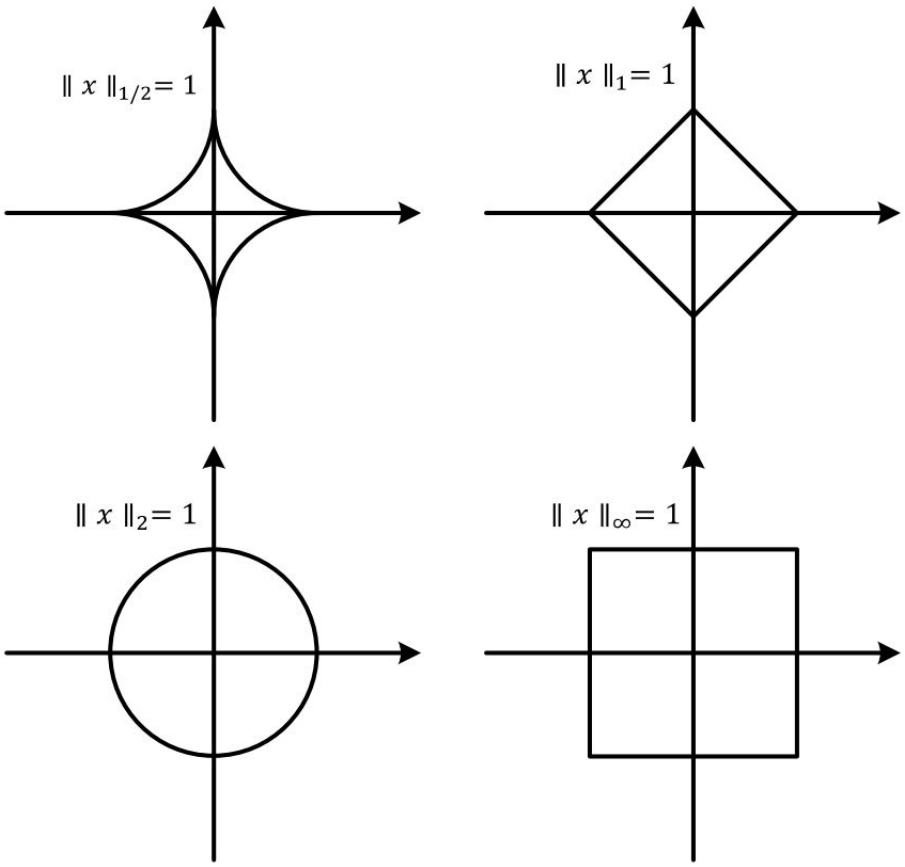


Figure 2.1: Examples of l_p norms for the unit circle

which is representative of the Euclidean length of b. The l_p norm can then be interpreted as the length of the vector in an l_p space. The l_p norm of a given vector $b \in \mathbb{R}^n$ is defined for all real numbers $p \geq 1$ as

$$\|b\|_p = (\sum_{i=1}^{n} |b(i)|^p)^{1/p} \tag{2.2}$$

As a special case, the infinite norm l_∞ is defined as the limit of the l_p norm when p tends to ∞,

$$\|b\|_\infty = \max_{i=1,2,..,n} |b(i)| \tag{2.3}$$

It should be noted that for all $p \geq 1$, the triangular inequality holds and certain examples are shown in Fig. 2.1 for unit circles in two-dimensional space. When $0 < p < 1$, the inequality no longer holds which results in concave functions instead of convex.

2.1.3 Sparsity

A signal is said to be sparse when most of its coefficients are null or negligible. The sparsity of a signal is defined by the cardinality of its support set, which simply means the number of non-zero indices that support the signal in the $\mathbb{R}^n$ space. The l_0 pseudo-norm can be used to indicate the sparsity level of the signal K by

$$\|b\|_0 \leq K \tag{2.4}$$

without providing any information on its energy. In practical signals of interest, the acquired samples typically are not perfectly sparse on any basis owing to the inherent perturbations in the acquisition process. The sorted samples $b^{'} = sort(b)$ demonstrate a power-law decay property in a suitable domain, defined by

$$|b^{'}(i)| \leq C \cdot i^{-q} \tag{2.5}$$

with a constant C and $q \geq 0$. Equation 2.5 implies that a significant part of the signal's energy is held by a few coefficients. Natural signals are often compressible in the sense that discarding a huge percentage of the coefficients contributes to a minimal information loss. This suggests that any natural signal can be approximated by a sparse vector in a suitable domain under bounded error constraints.

2.2 Compressive Sensing framework

2.2.1 Compressed measurements

The CS framework enables signals to be sensed at reduced sampling rates and still performs reconstructions perfectly. The sensing mechanism is realized in such a way that every measurement would contain information about the entire signal.

The measurements in CS are obtained by projecting the signal on to vectors $\{\phi_i\}_{i=1}^{M}$ resulting in the measurements vector $\boldsymbol{y} = \{y_i\}_{i=1}^{M}$ computed as the inner product of the corresponding columns of $\boldsymbol{\Phi} = [\boldsymbol{\phi}_1^T \boldsymbol{\phi}_2^T \cdots \boldsymbol{\phi}_M^T]$, which can be mathematically represented as

$$\boldsymbol{y} = \boldsymbol{\Phi b} \tag{2.6}$$

Assuming a sparse representation for $\boldsymbol{b}$ in an orthogonal basis $\boldsymbol{\Omega}$ of size $N \times N$ corresponding to the length of the signal,

$$\boldsymbol{y} = \boldsymbol{\Phi b} = \boldsymbol{\Phi \Omega a} = \boldsymbol{\Psi a} \tag{2.7}$$

$\boldsymbol{\Phi}$ corresponds to the measurement or sensing matrix which relates each non-zero term in $\boldsymbol{a}$ to a particular measurements pattern. The basis vectors $\boldsymbol{\psi}_i$'s of $\boldsymbol{\Psi}$ are referred to as dictionary elements or atoms. Fig. 2.2 compares the traditional Nyquist and CS acquisitions in terms of matrix equations. Since the measurement scheme is independent of the signal under consideration, the measurements in the CS framework are typically non-adaptive.

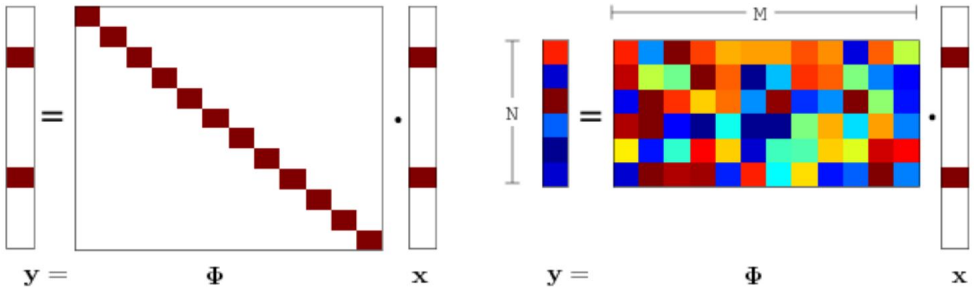


Figure 2.2: Comparison of traditional and CS acquisitions

The design of measurement matrices that can guarantee reconstruction of $\boldsymbol{a}$ in the CS framework will be discussed in the following subsection.

2.2.2 Recovery criteria

The reconstruction of $\boldsymbol{a}$ from compressed measurements is an ill-posed problem as it corresponds to an under-determined linear system with infinitely many solutions. However, enforcing the condition of sparsity on the signal can lead to a unique solution. The objective of sparse recovery is to produce an accurate representation of the signal in either the identity basis or the sparsifying orthogonal basis $\boldsymbol{\Omega}$ in 2.7. This recovery optimization problem can be mathematically formulated as

$$\hat{\boldsymbol{b}} = \min_{\boldsymbol{b}} \|\boldsymbol{b}\|_0 \quad \text{subject to} \quad \boldsymbol{y} = \boldsymbol{\Phi b} \tag{2.8}$$

The naive realization of the l_0 minimization problem requires exhaustively scouring through ${}^{N}C_{K}$ sparse combinations which is infeasible for even small problem sizes as it is NP-hard. There has been a considerable priority to devising feasible alternatives to realize the viability of CS in real-world applications.

The major breakthrough in CS research was the work of Candes and Tao [6] that proved the convex l_1 minimization can reproduce the sparse signal with high probability by enforcing certain conditions on the measurement matrix. This seminal formulation of the sparse recovery problem has been known as the basis pursuit (BP)

$$\hat{\boldsymbol{b}} = \min_{\boldsymbol{b}} \|\boldsymbol{b}\|_1 \quad \text{subject to} \quad \boldsymbol{y} = \boldsymbol{\Phi b} \tag{2.9}$$

Fig. 2.3 depicts the reconstruction of a signal residing in the x-y plane with a sparsity of 1 which implies it lies on one of the axes. It can be seen that minimizing the l_2 norm is equivalent to growing the l_1 sphere till it meets the constraint line. Similarly, minimizing the l_1 norm can be equated to growing the l_1 rhombus till it touches the constraint line. Of the two, it can be observed that the l_1 minimization is able to meet the constraint line at the axes which is where the underlying signal resides. This visual representation demonstrates

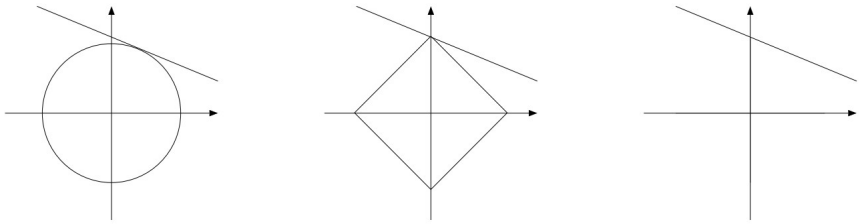


Figure 2.3: Representation of different l_p norms

the ability of the basis pursuit strategy to successfully reconstruct the sparse signal and has considerably less computational complexity compared to the l_0 minimization.

Considering the impact of noise in the measurement process, the basis pursuit has been refined to account for noisy measurements as follows

$$\hat{\boldsymbol{b}} = \min_{\boldsymbol{b}} \|\boldsymbol{b}\|_1 \quad \text{subject to} \quad \|\boldsymbol{\Phi b} - \boldsymbol{y}\|_2 \leq \epsilon \tag{2.10}$$

The basis pursuit denoising (BPDN) refinement ensures near accurate signal representations with theoretical bounds on the error.

2.2.2 Null Space Property

The null space of the measurement matrix can be defined as the set of all vectors $\boldsymbol{b}$ that can be mapped to 0.

$$\mathbf{N}(\boldsymbol{\Phi}) = \{\boldsymbol{b} : \boldsymbol{\Phi b} = 0\} \tag{2.11}$$

This can be used to fashion some criteria on $\boldsymbol{\Phi}$ for achieving successful recovery. Firstly, it can be said that two distinct signals of sparsity K cannot produce identical measurement vector, or mathematically as follows

$$\boldsymbol{\Phi}(\boldsymbol{b} - \boldsymbol{b'}) \neq 0 \Leftrightarrow \boldsymbol{b} - \boldsymbol{b'} \notin \mathbf{N}(\boldsymbol{\Phi}) \tag{2.12}$$

Secondly, subtracting a K-sparse vector from another results in a vector that is atmost $2K$-sparse. If $\mathbf{N}(\boldsymbol{\Phi})$ does not contain any $2K$-sparse vector, this would imply that $\boldsymbol{b}$ is

unique and results in a lower bound on the measurement vector size that would facilitate reconstruction, i.e., $M \geq 2K$.

The null space concentration is measured by the null space property (NSP) which is satisfied when

$$\|\boldsymbol{d}_G\|_2 \leq \gamma \|\boldsymbol{d}_{G^C}\|_1, \forall \boldsymbol{d} \in \mathbf{N}(\boldsymbol{\Phi}) \quad \text{and} \quad \gamma \in (0,1) \tag{2.13}$$

for sets $G \subset 1, .., N$ having a cardinality of K. A necessary criteria for perfect reconstruction by basis pursuit is satisfying the NSP which can be shown as a reconstruction error upper bound for approximate K-sparse signals as in [25]

$$\|\boldsymbol{b} - \hat{\boldsymbol{b}}\|_1 \leq 2\frac{1+\gamma}{1-\gamma}\sigma_K(\boldsymbol{b})_1 \tag{2.14}$$

where $\sigma_K(\boldsymbol{b})_p$ is the error corresponding to the ideal K-sparse representation

$$\sigma_K(\boldsymbol{b})_p = \min_{\|\boldsymbol{b}\|_0 == K} \|\boldsymbol{b} - \hat{\boldsymbol{b}}\|_p \tag{2.15}$$

2.2.2 Restricted Isometry Property

Also known as the uniform uncertainty principle, restricted isometry property has been devised to evaluate the robustness of CS in noisy and approximate sparse scenarios [5]. The necessary criteria for a matrix $\boldsymbol{\Phi}$ to fulfil the RIP with an isometry constant δ_k is

$$1 - \delta_k \leq \frac{\|\boldsymbol{\Phi}\boldsymbol{b}\|_2}{\|\boldsymbol{b}\|_2} \leq 1 + \delta_k \tag{2.16}$$

Candes et al.[26] provided a theoretical bound for BPDN reconstruction of eq. (2.10) under bounded error and isometry constant conditions ($\delta_{3K} + 3\delta_{4K} < 2$)

$$\|\boldsymbol{b} - \hat{\boldsymbol{b}}\|_2 \leq C_1\epsilon + C2\sigma_K(\boldsymbol{b})_1/\sqrt{K} \tag{2.17}$$

The above equation has two terms related to the measurement noise ϵ and imperfect sparsity $\sigma(\boldsymbol{b})_1$ which cannot be avoided in realistic settings. Though this guarantees exact sparse recovery when $\delta_{2k} < \sqrt{2} - 1$, finding the RIP constant is itself an NP-hard problem.

2.2.2 Mutual incoherence

Another condition that can be enforced on the measurement matrix is the measure of coherence between $\mathbf{\Phi}$ and the sparsifying basis $\mathbf{\Gamma}$ which is defined as follows

$$\mu(\mathbf{\Phi}, \mathbf{\Gamma}) = \max_{i \in [1,M], j \in [1,N]} | < \phi_i, \gamma_j > | \tag{2.18}$$

It would also suffice to measure the coherence of the effective measurement matrix given by $\mathbf{\Psi}$

$$\mu(\mathbf{\Psi}) = \max_{i \in [1,M], j \in [1,N]} | < \psi_i, \psi_j > | \tag{2.19}$$

Rauhut et al. established a relation to the restricted isometry constant in [27]:

$$\delta_k \leq (K-1)\mu \tag{2.20}$$

It is desired to have lesser coherence between the bases to facilitate better reconstructions. An outcome of this property is that signals sparse in one basis cannot be sparse in the other basis. Candes and Tao [6] demonstrated the effectiveness of random matrices in reconstructing sparse signals with high probability. A few good examples are the random Gaussian, Bernoulli and Fourier matrices which permit CS reconstructions with varying bounds on the number of required measurements.

2.2.3 Recovery algorithms

The CS reconstruction arena has garnered significant attention in the last decade with a plethora of algorithms introduced to reconstruct $\boldsymbol{b}$ from compressive measurements $\boldsymbol{y}$ by

solving the optimal BP problem (2.9) or sub-optimal variations. In this section, a broad class of CS recovery algorithms are discussed elaborating their merits and demerits in order to propose novel algorithms with a prime focus on suitability for hardware implementation. The major features essential for efficient realization on hardware are fixed-point suitability, regular structure, simplified flow and low memory requirements.

2.2.3 Convex optimization

Convex optimization algorithms were deployed immediately to solve the BP problem (2.9) and are well-developed [28]. The earliest works used iterative approaches like simplex and interior-point methods to solve the convex formulations. Many approaches solve the Lagrangian form of the BPDN

$$\hat{\boldsymbol{b}} = \min_{\boldsymbol{b}} \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{\Phi b}\|_2^2 + \lambda\|\boldsymbol{b}\|_1 \tag{2.21}$$

Prominent methods solving this formulation are primal-dual interior-point [28] and fixed-point continuation [29]. While the class of convex relaxation algorithms are known to produce optimal solutions for moderate to large sized problems, they are associated with very high computational complexity and sophisticated structures that prohibit their mapping on custom hardware. The computational complexity is generally of the order of $O(N^3)$ [25] which results in slower recovery when signal dimensions N, m and k grow.

2.2.3 Greedy Approaches

Greedy approaches have received significant attention in the literature due to their speed and ease of implementation to design hardware that can potentially realize CS recovery in real world settings. Greedy algorithms iteratively approach the sub-optimal solution with significant reduction of the computational complexity. Due to this, they typically possess weaker theoretical guarantees than l_1 minimizers. Few greedy approaches use sophisticated

techniques to match the theoretical performance of the basis pursuit methods.

Greedy algorithms generally function in an iterative approach and consist of three main operations:

- Support set augmentation
- Signal estimation
- Residual update

The first operation involves selecting the column(s) of the measurement matrix Φ that most likely contribute to the measured vector $\boldsymbol{y}$. The selected column(s) are augmented to the estimated support set which is then used to estimate the signal corresponding to the support in the estimation step. In the final residual update step, the contributions of the estimated signal are removed from the measured vector $\boldsymbol{y}$ to identify the remaining columns of the true support set.

2.2.3 Matching Pursuit (MP)

MP proposed in [30] was one of the first algorithms employed to solve the sparse recovery problem due to its very simple structure. MP selects the column possessing highest correlation with $\boldsymbol{y}$ in each iteration and updates the corresponding signal coefficient with the correlation product itself. Since only the current element is updated and it is typically non-orthogonal to previous elements, multiple updates of the same coefficient might occur resulting in poor performance. The algorithm pseudo-code is shown below Due to the very simple update procedure, MP is well-suited for hardware optimizations and implementations. However, the non-orthogonality of the selected indices yields slow convergence and offers no guarantees of reconstruction. Due to the possibility of repeatedly selecting the same element, the iteration count can render it inefficient.

Algorithm 1: Matching Pursuit (MP)

procedure MP ($\mathbf{y}$, $\mathbf{\Phi}$, $\mathbf{d}$)
Initialization $\hat{\mathbf{x}}^0$=$0_{N\times 1}$, $\mathbf{r}^0 = \mathbf{y}$, $n = 1$, $\mathbf{\Gamma}^0 = \phi$ **while** $n < K$ **do**
$\boldsymbol{u}^n = \mathbf{\Phi}^T \boldsymbol{r}^{n-1}$
$j^n = \arg\max_i |\boldsymbol{u}^n|$ ▷ **Identify**
$\mathbf{\Gamma}^n = \mathbf{\Gamma}^{n-1} \cup j^n$ ▷ **Support Set Augmentation**
$\hat{\mathbf{x}}^n = \hat{\mathbf{x}}^{n-1} + \boldsymbol{e}_{j^n} \boldsymbol{u}^n_{j^n}$ ▷ **Estimate Update**
$\mathbf{r}^n = \mathbf{y} - \mathbf{\Phi}\hat{\mathbf{x}}^n$ ▷ **Residual Update**
$n = n + 1$
end
return Estimated sparse signal $\hat{\mathbf{x}}^n$
end procedure

2.2.3 Orthogonal Matching Pursuit (OMP)

OMP [7] is one of the rudimentary algorithms in arena of sparse recovery and selects the index of $\boldsymbol{u}$ corresponding to the highest correlation magnitude to be part of $\mathbf{\Gamma}^n$ similar to MP. However, it fundamentally differs in the signal estimation step wherein all estimated support set elements of $\mathbf{\Gamma}^n$ are updated in each iteration. The estimation is composed of a least squares (LS) optimization in the span of the currently estimated support set

$$\hat{\mathbf{x}}^n = \min_b \|\mathbf{\Phi}_{\mathbf{\Gamma}^n} \boldsymbol{b} - \boldsymbol{y}\|_2 \tag{2.22}$$

This forces the new estimate to be orthogonal with respect to the residual and avoids repeated selection of a column, unlike MP. This enables OMP to reconstruct the signal in ideally K iterations. The pseudo-code for the OMP algorithm is shown below. The LS update is computationally more complex than the MP update which can be simplified by adopting efficient matrix decomposition methods like QR or Cholesky, which imparts a regular structure to OMP. While this makes it feasible for hardware implementation, running time increases drastically with rising k due to its serial nature. Further, recovery deteriorates under noisy conditions, which necessitates running the algorithm for greater number of iterations than the true sparsity of the signal to reasonably reconstruct it.

Algorithm 2: Orthogonal Matching Pursuit (OMP)

procedure OMP (**y**, $\mathbf{\Phi}$, **d**)
Initialization $\hat{\mathbf{x}}^0$=$0_{N\times 1}$, $\mathbf{r}^0 = \mathbf{y}$, $n = 1$, $\mathbf{\Gamma}^0 = \phi$ **while** $n < K$ **do**
$\boldsymbol{u}^n = \mathbf{\Phi}^T \boldsymbol{r}^{n-1}$
$j^n = \arg\max_i |\boldsymbol{u}^n|$ ▷ **Identify**
$\mathbf{\Gamma}^n = \mathbf{\Gamma}^{n-1} \cup j^n$ ▷ **Support Set Augmentation**
$\hat{\mathbf{x}}^n = \min_b \|\mathbf{\Phi}_{\mathbf{\Gamma}^n} \boldsymbol{b} - \boldsymbol{y}\|_2$ ▷ **Estimate Update**
$\mathbf{r}^n = \mathbf{y} - \mathbf{\Phi}\hat{\mathbf{x}}^n$ ▷ **Residual Update**
$n = n + 1$
end
return Estimated sparse signal $\hat{\mathbf{x}}^n$
end procedure

2.2.3 Regularized Orthogonal Matching Pursuit (ROMP)

There have been concerted efforts to accelerate the serial pursuing process of OMP which have resulted in works like stagewise OMP (StOMP) [31] and stagewise weak OMP (SWOMP) [32]. These try to extend the number of columns selected in each iteration based on a fixed size or threshold. Regularized OMP [33] proposed a novel regularization step on the K largest coefficients of **u** to augment an optimal subset with maximum energy to $\mathbf{\Gamma}$ in each iteration. The pseudo-code of the ROMP algorithm is shown below

Algorithm 3: Regularized Orthogonal Matching Pursuit (ROMP)

procedure ROMP (**y**, $\mathbf{\Phi}$, **d**)
Initialization $\hat{\mathbf{x}}^0$=$0_{N\times 1}$, $\mathbf{r}^0 = \mathbf{y}$, $n = 1$, $\mathbf{\Gamma}^0 = \phi$ **while** $n < K$ **do**
$\boldsymbol{u}^n = \mathbf{\Phi}^T \boldsymbol{r}^{n-1}$
Choose K biggest coordinates of $\boldsymbol{u}$ in$\boldsymbol{J}$
Form disjoint subsets $\mathbf{J}_0 \subset \boldsymbol{J}$ such that
$|\boldsymbol{u}| \geq \frac{1}{2}|\boldsymbol{u}| \ \forall \ a, b \in \mathbf{J}_0$
Choose $\mathbf{J}_0$ with the highest average correlation energy $\sum_{b\in\mathbf{J}_0} |\boldsymbol{u}|^2$ ▷ **Regularize**
$\mathbf{\Gamma}^n = \mathbf{\Gamma}^{n-1} \cup j^n$ ▷ **Support Set Augmentation**
$\hat{\mathbf{x}}^n = \min_b \|\mathbf{\Phi}_{\mathbf{\Gamma}^n} \boldsymbol{b} - \boldsymbol{y}\|_2$ ▷ **Estimate Update**
$\mathbf{r}^n = \mathbf{y} - \mathbf{\Phi}\hat{\mathbf{x}}^n$ ▷ **Residual Update**
$n = n + 1$
end
return Estimated sparse signal $\hat{\mathbf{x}}^n$
end procedure

Since ROMP possesses a similar structure to OMP, it could benefit from many of the simplifications adopted in OMP. However, despite proven theoretical guarantees [33], ROMP exhibits poorer empirical performance than OMP and SWOMP as observed in [32]. Nevertheless, it paved the way for more multi-element pursuit methods to accelerate sparse recovery.

2.2.3 Compressive Sampling Matching Pursuit (CoSaMP)

With a view to considerably speed up the reconstruction process by employing a parallel estimation of the support set based on the sparsity parameter K, CoSaMP [34] selects $2K$ columns in each iteration and augments it to the K best columns after signal estimation in the previous iteration. This amounts to solving an LS problem of size $3K$ which is computationally much more intensive than the previously discussed greedy algorithms. However, it entails a provision to eliminate false choices and continuously refine the support set. The algorithm's pseudo-code is outlined below:

Algorithm 4: Compressive Sampling Matching Pursuit (CoSaMP)

procedure CoSaMP (**y**, **Φ**, **d**)
  Initialization $\hat{\mathbf{x}}^0$=$0_{N\times 1}$, $\mathbf{r}^0 = \mathbf{y}$, $n = 1$, $\mathbf{\Gamma}^0 = \phi$ **while** $n < K$ **do**
  $\boldsymbol{u}^n = \mathbf{\Phi}^T \boldsymbol{r}^{n-1}$
  $\mathbf{\Gamma}^n = \mathbf{\Gamma}^{n-1} \cup$ LargestIndices$(|\boldsymbol{u}^n|^2, 2K)$ ▷ **Support Set Augmentation**
  $\hat{\mathbf{x}}^n = \min_b \|\mathbf{\Phi}_{\mathbf{\Gamma}^n} \boldsymbol{b} - \boldsymbol{y}\|_2$ ▷ **Estimate Update**
  $\mathbf{\Gamma}^n =$ LargestIndices$(|\hat{\mathbf{x}}^n|^2, K)$
  $\mathbf{r}^n = \mathbf{y} - \mathbf{\Phi}\hat{\mathbf{x}}^n$ ▷ **Residual Update**
  $n = n + 1$
  end
  return Estimated sparse signal $\hat{\mathbf{x}}^n$
end procedure

2.2.3 Subspace Pursuit (SP)

SP [35] was proposed to reduce the complexity of CoSaMP by augmenting only K columns instead of $2K$ with slightly poorer theoretical reconstruction guarantees but better empirical performance than CoSaMP.

Parallel pursuit methods like CoSaMP [34] and SP [35] are highly dependent on the choice of sparsity level for good performance, which is not known in advance and can constantly vary in realistic scenarios. Furthermore, their computational complexity puts a significant burden on hardware resources even for moderate problem sizes.

2.2.3 Other greedy algorithms

Backtracking OMP (BaOMP) [36] alleviates the need for sparsity prior, but requires tuning of parameters for optimal performance. The performance of Sparsity Adaptive Matching Pursuit (SAMP) deteriorates on wrong choice of step size [37]. Modified Regularized Adaptive Matching Pursuit (MRAMP) [38] involves a combination of modified mean energy regularization strategy and the divide-and-conquer policy of SAMP, however, the step size parameter introduces a trade-off between recovery performance and fast convergence. Moreover, these sophisticated algorithms forbid efficient hardware realizations due to their irregular structure.

Threshold based strategies like Iterative Hard Thresholding (IHT) [39] and Approximate Message Passing (AMP) [40] are swifter and less complex than BP. However, computational complexity of IHT rises on adopting an adaptive step size for better performance while, AMP performs well for highly structured $\mathbf{\Phi}$. Bayesian approaches like Bayesian compressive sensing (BCS) [41] depend on an appropriate choice of signal prior and are computationally complex. Swift expander graph based strategies like [42, 43] use the adjacency matrix to reduce complexity of storing $\mathbf{\Phi}$ only when the signal of interest is sparse in the canonical basis [44].

The prominent algorithms discussed above will be experimentally evaluated in Chapter 3 for clearly understanding their merits and demerits under different settings. Table 2.1 shows the distinction between three broad classes of greedy algorithms: serial (like OMP [7], SGP [45] etc.) , multi-element like ROMP [33], BaOMP [36] etc. and parallel methods like CoSaMP [34], SP [35] etc. Serial pursuits select the column with the high-

est correlation to the residual in each iteration while multi-element and parallel methods employ a candidate list of K maximum correlation terms or more. Support augmentation involves matrix decomposition or inversion techniques whose complexity only incrementally increases with iterations in serial methods but has a fixed high complexity in parallel methods. The nature of the decomposition or inversion algorithm has a bearing on the complexity of the least squares based signal estimation. For instance, incremental QR updates can be employed to bypass signal estimation in each iteration to directly update the residual and perform signal estimation once after the algorithm terminates. These incremental schemes are not effective for multi-element or parallel strategies due to the varying nature of their support sets in every iteration.

Table 2.1: Sub-phases of greedy algorithms

Sub-phase	**Serial**	**Multi-element**	**Parallel**
Index selection	Correlation Maximum	Subset of K largest correlation terms	$2K$/$3K$ largest correlation terms
Support augmentation	Incremental QR or Cholesky decompositions, Conjugate gradient pursuits, Matrix inversion methods like Guass-Siedel or Jacobi	May or may not permit incremental decompositions or pursuits	Not suited for incremental pursuits
Signal estimation	Least squares	Least squares	Least squares
Residual update	Incremental or full	Incremental or full	Full

2.3 Existing Art

There has been substantial research in CS theory to propose swifter and less complex sparse recovery algorithms with theoretical convergence guarantees. Concurrently, efforts were underway to build physical systems capable of compressively sensing signals and efficiently reconstructing them. Here, the CS space is reviewed in terms of the existing acquisition and reconstruction systems with more focus attributed to the implementation of sparse recovery algorithms in hardware.

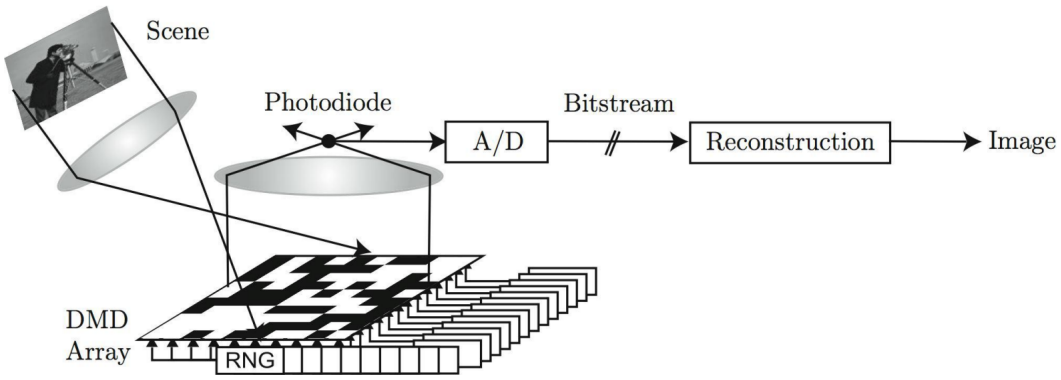


Figure 2.4: Single pixel camera architecture [46]

2.3.1 CS acquisition

From the CS preliminaries, it can be concluded that signal acquisition has to occur in a domain that is incoherent to the sparsifying basis of the signal under consideration. In certain applications, the sampling functionality ensures incoherence, for instance in MRI, where the 2-D Fourier space is sampled. However, other applications would need specialized acquisition devices that can handle the incoherence requirement of CS for facilitating successful reconstructions.

The single pixel camera [46] was the first imaging device built on CS principles that has vast applications in complex and expensive spectral imagers. It used a digital micro-mirror array as shown in Fig. 2.4 to compressively capture pixel information using a single sensor and acquires multiple measurements in this way to reliably reconstruct the original scene.

CS also finds tremendous application in the field of wideband signal acquisition where fewer number of samples will be sufficient for acquiring signals with sparse spectrum use. The random demodulator [47] carried out the acquisition by mixing the wideband signal with a pseudorandom ± 1 sequence and integrated to obtain sub-Nyquist samples, as shown in Fig. 2.5.

The random demodulator was implemented as an actual system in [14] where projecting onto PN sequences in the digital domain showed energy efficiency. An analog-to-

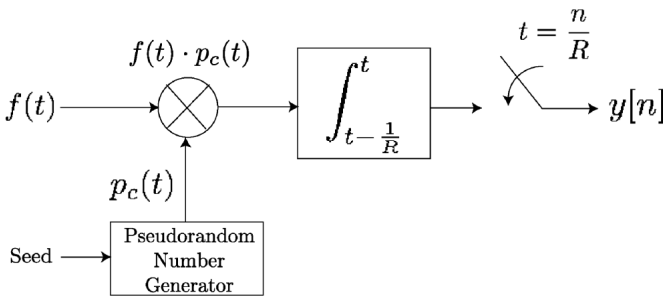


Figure 2.5: Block diagram of random demodulator [47]

information structure was presented in [48] that claimed better energy efficiency than the prior implementations.

There have been significant efforts to designing acquisition systems for radar [49] [50], EEG/ECG [20] and other interesting fields.

2.3.2 CS recovery implementation

The above discussed applications mandate real-time signal reconstruction which is often very complex despite the plethora of available algorithms. Further, the advent of many-core central processing units (CPUs) and graphics processing units (GPUs) can accelerate these reconstruction algorithms at significant power consumption and below real-time constraints [51].

In certain applications, cloud computing can be used to process the compressed data, albeit at longer latencies, higher computing power and data privacy risks [52]. Many medical and wireless communication applications necessitate energy efficient reconstruction of the compressed data to facilitate timely predictions or decisions. Custom FPGA or application-specific integrated circuit (ASIC) implementations can meet the time and power constraints of critical applications, compared to CPU or DSP implementations. A detailed review of the state-of-art algorithms in CS recovery are presented below to motivate the need for faster, inexpensive and energy-efficient implementations.

An early hardware implementation of the MP for acoustic channel estimation has been

presented in [53]. Since MP is prone to errors arising from repeated index selection, it has received little attention in the hardware literature since the introduction of OMP. One of the very first FPGA implementations [54] employed a two-stage variant of OMP on a Virtex-5 FPGA to estimate a 128-dimensional 5-sparse signal in 24μs, where the first stage utilizes a modified Gram Schmidt (MGS) process to determine the support set and update the residual followed by an alternating Cholesky least squares (LS) step in the second stage. Stanislaus et al. presented a 65nm ASIC and Virtex 5 FPGA implementation based on a two-stage OMP approach in [55] and [56] that reconstructed a 256-length signal having sparsity 8 in 14μs and 27μs respectively.

Bai et al. [57] utilized the MGS strategy of [54] to implement an incremental QR decomposition (QRD) instead of a full QRD for the LS update in each iteration, for the reconstruction of a 1024-length signal with a sparsity of 36 in 620μs on a Virtex 5 FPGA. The Cholesky strategy has also been exploited by works in [58] [59] to implement the LS block of OMP since it bypasses the need for costly inverse square root operations that arise in the QRD process. However, an increase in matrix dimensions increases the latency exponentially, contributing to lower speeds and throughputs.

Ren et al. [60] presented a single-precision OMP reconstruction engine on a Kintex-7 FPGA for 10$\times$ more sparse coefficients, requiring approximately 40μs per iteration of the OMP. Knoop et al. [61] investigated a *high-level synthesis* (HLS) based design of OMP which utilizes a rank-1 updating scheme for $\boldsymbol{Q}$ and $\boldsymbol{R}^{-1}$ matrices to reduce the cost of $\boldsymbol{R}$ matrix inversion in the final estimation step. Although HLS based design enables rapid prototyping on the FPGA, it incurs higher hardware costs for reconstructing a 256-length vector on a Virtex-7 device. Cheng et al. [62] proposed a matrix inversion free architecture based on QRD for a variable orthogonal multi matching pursuit (vOMMP) in 90 nm CMOS achieving an execution time of 232 μs for 128-dimensional signals, whose reconstruction performance is dependent on a good initial support set, which is not extendable to all signals.

Ren et al. [63] further proposed a sparse approximation engine based on OMP in 40 nm CMOS technology for reconstructing compressively acquired physiological signals at a maximum throughput of 237 kS/s. [64] utilized a *matrix inversion bypass* (MIB) technique to effectively decouple algorithmic steps and exploit the resulting parallelism to enforce faster reconstructions of around 250μs on a Kintex-7 FPGA. Kulkarni et al. [65] implemented application-specific architectures of the OMP for signal size ranging from 128 to 1024 on 65nm CMOS technology. While recovery of 8-sparse 256 dimensional vectors takes 10μs, timing results for 1024 dimensional vectors are absent. Fardad et al. [66] proposed low complexity hardware based on OMP using deterministic measurement matrices attaining a reconstruction speed of 333μs on a Virtex-6 device. Liu et al. [67] proposed an index selection scheme to curb the number of iterations by a factor of 2 and employed a complex Cholesky decomposition architecture to estimate a 1024-length 36-sparse signal in 170μs on a Virtex-6 FPGA. Roy et al. [68] employs a partial incremental QRD to bypass the normalization step in the computation of $\boldsymbol{Q}$ and achieves recovery in 327μs for identical problem sizes on a Virtex-6 device.

Ge et al. [69] incorporated square-root-free transformations to the incremental QRD algorithm to achieve recovery of the 36 non-zero coefficients of the 1024-length signal $\mathbf{x}$ in 238μs on a Kintex-7 FPGA. Li et al. [70] reported an HLS based design of OMP that achieves a recovery time of 423μs on the Zynq Ultrascale+ MPSoC at exorbitant hardware costs. Chen et al. [71] presented a CS reconstruction engine in 40 nm CMOS employing a two-stage process of blind sparsity estimation (SE) based on OMP followed by non-blind SP based fine reconstruction called SE-SP. Wang et. al. [72] proposed an architecture based on the alternating direction method of multipliers (ADMM) to reconstruct signals of length 128 in 44μs, whose arithmetic operations are quadratic in N^2. Liu et al. [73] also proposed an SP processor on a Virtex-7 FPGA for the recovery of 256-length signals having a sparsity of 8 with an expected heavy toll on hardware resources.

Table 2.2 tabulates the state-of-art hardware implementations targeting different greedy

Table 2.2: State-of-art hardware implementations on the FPGA platform

Reference & Year	Algorithm	Problem Size			Time	Frequency	Target
		M	N	K	(μs)	(MHz)	FPGA
[57] ('12)	OMP	256	1024	36	622	100	Virtex 6
[59] ('15)	OMP	256	1024	36	340	119	Virtex 6
[67] ('18)	Imp. OMP	256	1024	36	170	135.4	Virtex 6
[74] ('18)	OMP	256	1024	36	450/314	94/135	Virtex 6/7
[68] ('19)	OMP	256	1024	36	327	133.33	Virtex 6
[69] ('19)	OMP	256	1024	36	238	210	Kintex 7
[70] ('20)	Imp. OMP	256	1024	36	423	113	Zynq Ultrascale
[73] ('20)	SP	58	256	8	22.5	15.4	Virtex 7

algorithms on various FPGA platforms with their targeted problem sizes and corresponding reconstruction time.

2.4 Summary

From the study of the state-of-art in CS recovery algorithms and their implementations, it is surmised that CS recovery in general is very complex and needs improved algorithms that can achieve a good balance between hardware feasibility, reconstruction speed and scalability with higher dimensions. Being the most prominent algorithm in CS recovery, OMP suffers from poor reconstruction speeds in higher dimensional CS problems. SP introduced a parallel pursuit strategy to speed up reconstruction in such scenarios, but is dependent on knowledge of sparsity level for satisfactory performance, which would require a form of serial sparsity estimation in such blind or varying scenarios.

Despite a plethora of greedy reconstruction algorithms developed in recent years, the state-of-art hardware implementations have been mainly restricted to the OMP. This is because OMP possesses a regular structure compared to the other algorithms which use

complicated structures to reduce the iteration count and realize sparsity adaptiveness. In this context, it is summarized that algorithm development should also consider the hardware deployability aspect and involve simple as well as effective structures to realize higher reconstruction speed and sparsity independence.

Chapter 3

Sparsity Independent Regularized Pursuit algorithm

This chapter deals with the development of a novel sparse reconstruction algorithm which possesses the desired features of faster convergence, sparsity independence, hardware feasibility and scalability.

3.1 Introduction

OMP [7] sequentially estimates the support of the signal in k iterations, which consumes long operating cycles. Parallel pursuits like subspace pursuit (SP) [35] lower the number of iterations at the expense of elevated computational complexity and require sparsity information for good reconstructions. Though methods like sparsity adaptive matching pursuit (SAMP) [37] were devised to achieve sparsity independence, they are not amenable to hardware implementations owing to the complicated structures involved in attaining sparsity independence.

In this context, the development of a novel sparsity independent regularized pursuit is presented which scales well on hardware, significantly reduces the number of iterations and does not involve sophisticated mechanisms to estimate the sparsity. The mathematical guarantees for successful recovery and the associated computational complexity are discussed. A rigorous experimental evaluation of the algorithm's features are carried out to demonstrate its performance with respect to the state-of-art algorithms.

3.2 Proposed algorithm

For signal recovery of $\mathbf{x}$ from noisy measurements $\mathbf{y}$ by greedy pursuits, it is observed that the observation vector $\mathbf{u} = \mathbf{\Phi}^T\mathbf{y}$ serves as a fair approximation to the signal $\mathbf{x}$. However, construction of $\mathbf{u}$ is computation intensive particularly for large N. OMP [7] selects the index of $\mathbf{u}$ with the highest magnitude to be augmented to the estimated set $\mathbf{\Gamma}$, while ROMP [33], CoSaMP [34] and SP [35] employ sorting of the indices according to their descending correlation magnitudes and select k, $3k$ and $2k$ best candidates for further pruning. Greedy pursuits vary primarily in the selection of support set indices to be augmented to $\mathbf{\Gamma}$.

The SIRP algorithm proposes to speed up the identification stage which is constrained by the complex matrix-vector multiplication $\mathbf{\Phi}^T\mathbf{y}$ and sorting operation in certain cases, without degrading the reconstruction performance. SIRP adopts a two-stage approach of identification and regularization to ensure this. The identification stage fundamentally differs from that of existing methods in that the atoms of $\mathbf{\Phi}$ are clustered allowing for a parallel estimation of support set candidates. The atoms are clustered into n_c groups (where $n_c < N$) using any ordered or random clustering rule, which unravels the $m \times N$ measurement matrix $\mathbf{\Phi}$ into n_c sub-matrices $\mathbf{\Phi}_{d_1}, \mathbf{\Phi}_{d_2}, \cdots, \mathbf{\Phi}_{d_{n_c}}$. The residual is correlated with each group of atoms in parallel and the index with maximum correlation in each group is chosen as a candidate. A regularization on the correlation magnitudes of these n_c parallel indices is to be performed in order to prune out any false candidates before augmenting to the estimated $\mathbf{\Gamma}$.

3.3 Choice of regularization strategy

In order to optimize the choice of k candidate atoms of $\mathbf{\Phi}$ with maximum residual correlation, ROMP [33] employs a maximal energy criterion on subsets $\mathbf{J}_0$ with comparable

coordinates according to

$$\mathbf{u}(i) \leq 2\mathbf{u}(j) \; \forall \; i, j \in \mathbf{J}_0 \tag{3.1}$$

The optimum subset $\mathbf{J}_0$ is augmented to $\mathbf{\Gamma}$ and the procedure is iteratively repeated till the residual falls below a given threshold or the support set cardinality exceeds $2k$. This however requires a prior knowledge of k which becomes unreasonable in practical scenarios where k can constantly vary and cannot be determined prior. In such cases, the sparsity parameter for the ROMP algorithm is set to the upper bound which can be supported by the specified number of measurements M. However, this causes ROMP to erroneously select a group of atoms with comparable coordinates according to (3.1) having poor individual correlation but possessing maximum group correlation energy due to their sheer volume. This restricts the applicability of ROMP in sparsity blind scenarios.

This motivates the need for improving the regularization strategy for the SIRP algorithm to avoid such errors. Intuitively, the highest average correlation energy of the group becomes a good choice for the regularization criterion. This stringent condition would select true support set atoms instead of false ones. ROMP [33] shows that employing the maximal energy criterion would provide an optimal subset which captures a significant portion of the correlation energy of the original k-element candidate set according to

$$\|\mathbf{y}|A\|_2 \geq \frac{1}{2.5\sqrt{\log k}} \|\mathbf{y}\|_2 \tag{3.2}$$

where A is the subset of k largest correlation energies of $\mathbf{u}$ chosen according to (3.1) that possesses maximum energy. This is improved by the choice of highest average energy regularization as shown in Lemma 2.1 below.

Lemma 3.3.1. *Let* $\mathbf{y}$ *be any vector in* $\mathbb{R}^m, m > 1$. *Then there exists a subset* $A \subset 1, ..., m$ *with comparable coordinates:*

$$|\mathbf{y}(i)| \leq 2|\mathbf{y}(j)| \; for \; all \; i, j \in A \tag{3.3}$$

and with high energy

$$\|\mathbf{y}|_A\|_2 \geq \frac{1}{2}\sqrt{\frac{|A|}{\log m}}\|\mathbf{y}\|_2 \tag{3.4}$$

provided it holds the maximum average energy.

Proof. A is partitioned into at most $O(\log m)$ disjoint sets A_q with comparable coordinates as in (3.3) such that at least one of these sets will have large energy as in (3.4). A_q is defined as

$$A_q := \{i : 2^{-q}\|\mathbf{y}\|_2 < |\mathbf{y}_i| \leq 2^{-q+1}\|\mathbf{y}\|_2\},\ \ q = 1, 2, ...\infty$$

Let $q_0 = \lceil \log m \rceil + 1$, so that $|\mathbf{y}_i| \leq \frac{1}{m}\|y\|_2$ for all $i \in A_q, q > q_0$. Then if the set U is defined as $U = \bigcup_{q \leq q_0} A_q$, then the energy content of U^c is given by

$\|\mathbf{y}|_{U^c}\|_2^2 \leq (m(\frac{1}{m}\|\mathbf{y}\|_2)^2) = \frac{\|\mathbf{y}\|_2^2}{m} \implies \mathbf{y}_{avg}(U^c) = \frac{\|\mathbf{y}\|_2^2}{m^2}$

It is evident that $\sum_{q \leq q_0} \mathbf{y}_{avg}(A_q) \geq \mathbf{y}_{avg} - \mathbf{y}_{avg}(U^c)$

$\Rightarrow \sum_{q \leq q_0} \mathbf{y}_{avg}(A_q) \geq \frac{\|\mathbf{y}\|_2^2}{m} - \frac{\|\mathbf{y}\|_2^2}{m^2} = \frac{m-1}{m^2}\|\mathbf{y}\|_2^2 \geq \frac{\|\mathbf{y}\|_2^2}{4}$

If $|A_q|$ is the cardinality of set A_q, there exists some $q \leq q_0$

$\mathbf{y}_{avg}(A_q) \geq \frac{\|\mathbf{y}\|_2^2}{4q_0} = \frac{\|\mathbf{y}\|_2^2}{4\log m}$

$\Rightarrow \|\mathbf{y}|_{A_q}\|_2 \geq \frac{1}{2}\sqrt{\frac{|A_q|}{\log m}}\|\mathbf{y}\|_2 \quad \left(\because \mathbf{y}_{avg}(A_q) = \frac{\|\mathbf{y}|_{A_q}\|_2^2}{|A_q|}\right)$

which completes the proof. □

A simple comparison of the correlation energy retained in the optimal subset for the maximal energy and the highest average energy criteria in (3.1) and (3.4) respectively reveal that the latter achieves a better bound and hence its chosen optimal subset would most likely lie in the true support of the signal, unlike the former. This is empirically validated in simulations shown in Fig. 3.1 where the estimated support set includes the true support set while ROMP employing (3.1) selects more false indices than the true support set when sparsity prior is unknown.

3.4 RIP based Recovery Condition Analysis

It is undertaken to prove that the proposed SIRP algorithm will yield exact recovery of k-sparse signals $\mathbf{x}$ in $\mathbb{R}^N$ from measurements $\mathbf{y} = \mathbf{\Phi x}$ under the condition that the measurement matrix $\mathbf{\Phi}$ satisfies the Restricted Isometry Property, which produces an estimated support set such that $\text{supp}(\mathbf{x}) \subset \mathbf{\Gamma}$. Recovering the signal is simplified by computing $\hat{\mathbf{x}} = (\mathbf{\Phi_\Gamma})^{-1}\mathbf{y}$, where $\mathbf{\Phi_\Gamma}$ represents the measurement matrix $\mathbf{\Phi}$ restricted to atoms indexed by $\mathbf{\Gamma}$. We prove a more powerful version of this statement in Theorem 2.2, which states that at least half of all freshly selected atoms in each iteration belong to the true support of $\mathbf{x}$.

Theorem 3.4.1. *Assume $\mathbf{\Phi}$ satisfies the Restricted Isometric Property (RIP) with parameters $(2k, \delta)$ for $\delta = \frac{0.04}{\sqrt{\log k}}$. Let $\mathbf{x} \neq 0$ be a k-sparse vector with measurements $\mathbf{y} = \mathbf{\Phi x}$. Then at any iteration, after the regularization step of SIRP $\mathbf{J}_0 \neq \emptyset$, $\mathbf{J}_0 \cap \mathbf{\Gamma^{n-1}} = \emptyset$ and*

$$|\mathbf{J_0} \cap \text{supp}(\mathbf{x})| \geq \frac{1}{2}|\mathbf{J}_0| \tag{3.5}$$

This states that at least 50% of candidate indices in $\mathbf{J_0}$ are part of the true support of $\mathbf{x}$.

Proof. SIRP finds at least one new atom belonging to the support set of $\mathbf{x}$ in every iteration. Incorrect atoms may also be estimated, but (3.5) establishes that the number of such incorrect atoms will always be lower than the number of atoms belonging to the true support. This consequently implies that provided $\mathbf{\Phi}$ satisfies RIP, SIRP guarantees exact sparse recovery.

RIP necessarily states that every k columns of $\mathbf{\Phi}$ approximately incorporate an orthogonal system and hence every k coefficients of the N-dimensional observation vector $\mathbf{u}$ appear as correlations of the M-dimensional measurement vector $\mathbf{y}$ with this approximate orthonormal basis [33]. Therefore, every k coefficients of the original sparse vector $\mathbf{x}$ and corresponding k coefficients of $\mathbf{u}$ are close in the Euclidean norm, as detailed in Lemma

3.4.2.

Lemma 3.4.2. *Assume a measurement matrix* $\mathbf{\Phi}$ *satisfies the RIP with parameters* $(2k, \delta)$. *Then for every* k*-sparse* $\mathbf{x} \in \mathbb{R}^N$ *and every set* $\mathbf{\Gamma} \subset \{1, ..., N\}, |\mathbf{\Gamma}| \leq k$, *then the observation vector* $\mathbf{u} = \mathbf{\Phi^T \Phi x}$ *will satisfy*

$$\|\mathbf{u}|_{\mathbf{\Gamma}} - \mathbf{x}|_{\mathbf{\Gamma}}\|_2 \leq 2.03\|\mathbf{x}\|_2 \tag{3.6}$$

Theorem 3.4.1 is proved by induction on the iteration of SIRP by which it is claimed that in all previous iterations, $\mathbf{J}_0$ is non-empty, disjoint from the previously estimated support set $\mathbf{\Gamma}$ and inequality (3.5) holds. If $\mathbf{\Gamma}$ be the set that contains indices selected in the previous iteration, it is inferred that

$$|\text{supp}(\mathbf{x}) \cup \mathbf{\Gamma}| \leq 2k \tag{3.7}$$

Let $\mathbf{J}_0$ and $\mathbf{J}$ be the regularized and initial candidate sets respectively of the current iteration and by definition, $\mathbf{J}_0$ is not empty. Suppose $\mathbf{r} \neq 0$ is the residual during current iteration and the signal to be identified presently, its measurements and observation vector be considered as follows

$$\mathbf{x}_0 := \mathbf{x}|_{\text{supp}(\mathbf{x})\setminus\mathbf{\Gamma}}; \qquad \mathbf{y}_0 := \mathbf{\Phi x}_0; \qquad \mathbf{u}_0 := \mathbf{\Phi}^T\mathbf{y}_0 \tag{3.8}$$

where $\mathbf{x}_0$ denotes the coefficients of $\mathbf{x}$ that have not yet been estimated, $\mathbf{y}_0$ denotes the contribution of those coefficients in the original measurement $\mathbf{y}$ and $\mathbf{u}_0$ denotes the observation vector obtained by correlating $\mathbf{y}_0$ with $\mathbf{\Phi}$. Lemma 3.4.3 deals with approximation of the observations and shows that $\mathbf{u}$ and $\mathbf{u}_0$ are close in the Euclidean norm when restricted to a small enough subset [33].

Lemma 3.4.3. *Consider the observation vectors* $\mathbf{u}_0 = \mathbf{\Phi^T y}_0$ *and* $\mathbf{u} = \mathbf{\Phi^T r}$. *Then for any*

set $\mathbf{\Lambda} \subset \{1, ..., N\}$ *with* $|\mathbf{\Lambda}| \leq 2k$*,*

$$\|(\mathbf{u}_0 - \mathbf{u})|_{\mathbf{\Lambda}}\|_2 \leq 2.4\delta\|\mathbf{x}_0\|_2 \tag{3.9}$$

In the following lemmas, it is shown that energy of the observation vector when restricted to the candidate set $\mathbf{J}$ and further to the optimal set $\mathbf{J}_0$ is not trivial. It needs to be proved that SIRP correctly identifies a fixed percentage of the correlation energy of the yet unselected part of the signal and subsequently through regularization selects a part of the true support as well, which is the desired conclusion.

Lemma 3.4.4. $\|\mathbf{u}|_{\mathbf{J}}\|_2 \geq 0.8\|\mathbf{x}_0\|_2$ *(localizing the energy)*

Proof. Assuming $\mathbf{I} = \text{supp}(\mathbf{x}) \setminus \mathbf{\Gamma}$ and thus the cardinality of set $\mathbf{I}$ i.e. $|\mathbf{I}| \leq k$, the construction of set $\mathbf{J}$ using the group maximums implies

$$\|\mathbf{u}_0|_{\mathbf{J}}\|_2 \geq \|\mathbf{u}_0|_{\mathbf{I}}\|_2 \tag{3.10}$$

Since $\mathbf{x}_0|_{\mathbf{I}} = \mathbf{x}_0$ and from the proposition in [33], we have

$$\|\mathbf{u}_0|_{\mathbf{I}}\|_2 \geq (1 - 2.03\delta)\,\|\mathbf{x}_0\|_2 \tag{3.11}$$

This yields the following inequality on combining the inequalities in (3.10) and (3.11) and employing Lemma 3.4.3

$$\|\mathbf{u}|_{\mathbf{J}}\|_2 \geq (1-2.03\delta)\|\mathbf{x}_0\|_2 - 2.4\delta\|\mathbf{x}_0\|_2 \geq (1-4.43\delta)\|\mathbf{x}_0\|_2 \geq 0.8\|\mathbf{x}_0\|_2 \quad \left(\because \delta = \tfrac{0.04}{\sqrt{\log k}}\right) \tag{3.12}$$

which completes the proof □

The regularization strategy yields the optimal subset $\mathbf{J}_0$ from the candidate set $\mathbf{J}$ and

Lemma 3.3.1 provides a bound on the energy retained by $\mathbf{J}_0$ as follows

$$\|\mathbf{u}|_{\mathbf{J}_0}\|_2 \geq \tfrac{1}{2}\sqrt{\tfrac{|\mathbf{J}_0|}{\log k}}\|\mathbf{u}|_{\mathbf{J}}\|_2 \tag{3.13}$$

Putting inequalities (3.12) and (3.13) together yields Lemma 3.4.5

Lemma 3.4.5. $\|\mathbf{u}|_{\mathbf{J}_0}\|_2 \geq \dfrac{0.4\sqrt{|\mathbf{J}_0|}}{\sqrt{\log k}}\|\mathbf{x}_0\|_2$

In order to validate the first claim of Theorem 3.4.1 that $\mathbf{J}_0$ is not empty, it is assumed that $\mathbf{J}_0 = \emptyset$. However, it is noted that $\mathbf{x}_0 \neq 0$, otherwise by inequality (3.8), $\text{supp}(\mathbf{x}) \subset \mathbf{\Gamma}$. This indicates the residual $\mathbf{r}$=0 at the beginning of present iteration, which is a contradiction. The claim that $\mathbf{J}_0 \neq \emptyset$ is validated by observing Lemma 3.4.5. The second claim is established by the fact that the observation vector $\mathbf{u} = \mathbf{\Phi}^T\mathbf{r}$ satisfies $\mathbf{u}_{\mathbf{\Gamma}} = 0$. Since $\mathbf{J}$ is defined to contain non-zero coefficients of $\mathbf{u}$, it follows that $\mathbf{J} \cap \mathbf{\Gamma} = \emptyset$. As $\mathbf{J}_0 \subset \mathbf{J}$, it easily implies that $\mathbf{J}_0 \cap \mathbf{\Gamma} = \emptyset$.

To prove the final claim of the theorem in inequality (3.5), it is contrarily assumed that the inequality in (3.5) fails, i.e., $|\mathbf{J}_0 \cap \text{supp}(\mathbf{x})| < \dfrac{1}{2}|\mathbf{J}_0|$. Then it follows that

$$|\mathbf{J}_0 \setminus \text{supp}(\mathbf{x})| > \frac{1}{2}|\mathbf{J}_0| \tag{3.14}$$

Suppose $\mathbf{\Omega} = \mathbf{J}_0 \setminus \text{supp}(\mathbf{x})$; then due to the property of comparable coordinates in $\mathbf{J}_0$ and since $\mathbf{\Omega} > \dfrac{1}{2}|\mathbf{J}_0|$, there is a fraction of energy in $\mathbf{\Omega}$ obtained as

$$\|\mathbf{u}_{\mathbf{\Omega}}\|_2 > \frac{1}{\sqrt{5}}\|\mathbf{u}|_{\mathbf{J}_0}\|_2 \geq \frac{\sqrt{|\mathbf{J}_0|}}{5.6\sqrt{\log k}}\|\mathbf{x}_0\|_2 \geq \frac{1}{5.6\sqrt{\log k}}\|\mathbf{x}_0\|_2 \tag{3.15}$$

by employing Lemma 3.4.5. Whereas $\mathbf{u}$ can be approximated using $\mathbf{u}_0$ as

$$\|\mathbf{u}|_{\mathbf{\Omega}}\|_2 \leq \|\mathbf{u}|_{\mathbf{\Omega}} - \mathbf{u}_0|_{\mathbf{\Omega}}\|_2 + \|\mathbf{u}_0|_{\mathbf{\Omega}}\|_2 \tag{3.16}$$

Since $\mathbf{\Omega} \subset \mathbf{J}$ and from Lemma 3.4.3, it follows that

$$\|(\mathbf{u} - \mathbf{u}_0)|_{\mathbf{\Lambda}}\|_2 \leq 2.4\delta\|\mathbf{x}_0\|_2 \tag{3.17}$$

By the definition of $\mathbf{x}_0$ in inequality (3.8), $\mathbf{x}_0|_{\mathbf{\Omega}} = 0$. Therefore, by Lemma 3.4.2 it is seen that

$$\|\mathbf{u}_0|_{\mathbf{\Omega}}\|_2 \leq 2.03\delta\|\mathbf{x}_0\|_2 \tag{3.18}$$

Employing inequalities (3.17) and (3.18) in inequality (3.16), it is concluded that

$$\|\mathbf{u}|_{\mathbf{\Omega}}\|_2 \leq 4.43\delta\|\mathbf{x}_0\|_2 \tag{3.19}$$

This is a contradiction to inequality (3.15) as long as $\delta \leq \frac{0.04}{\sqrt{\log k}}$ and hence Theorem 3.4.1 is proved. Thus, SIRP guarantees exact sparse recovery of all k-sparse signals $\mathbf{x}$ in at most k iterations by yielding a set $\mathbf{\Gamma}$ such that supp$(\mathbf{x}) \subset \mathbf{\Gamma}$. However, SIRP is experimentally validated to require much fewer than k iterations to achieve exact recovery. □

3.5 Algorithm formulation

This section details the basic operation of the proposed SIRP algorithm which takes the measurements $\mathbf{y}$ and the measurement matrix sub-groups $\mathbf{\Phi}_{g_1,\cdots,g_{n_c}}$ as input. The estimated N-dimensional signal $\hat{\mathbf{x}}$ is initialized to 0 and the initial residual $\mathbf{r}^0$ is set to $\mathbf{y}$, while the estimated support set is initially empty. The algorithm involves three main stages: support identification, regularization and residual update. In every iteration, the index of the atom corresponding to the highest correlation with the residual $\mathbf{r}$ from each cluster is selected and added to an intermediate set $\mathbf{J}$, which contains the indices of n_c group maximums. This concludes the identification stage.

SIRP constructs disjoint subsets from $\mathbf{J}$ with comparable correlation energies according to (3.1), which ensures that the least energy in a given subset will be greater than the highest

energy within the same subset. SIRP then employs the highest group-average-energy based regularization criterion among the disjoint subsets of $\mathbf{J}$ to obtain the optimal subset of comparable correlation energies and their corresponding indices form the optimal index set $\mathbf{J}_0$. This is augmented to the estimated support set $\mathbf{\Gamma}^n$ in the current iteration.

The final stage involves estimating the signal using the currently estimated columns of $\mathbf{\Phi}_{\mathbf{\Gamma}^n}$ and given measurements $\mathbf{y}$ by employing least squares minimization. This can accomplished by adopting matrix decomposition methods like QR or Cholesky that efficiently compute the pseudo-inverse of the estimated sub-matrix i.e. $\mathbf{\Phi}^{\dagger}_{\mathbf{\Gamma}^n}$. Invoking these strategies simplifies the signal estimation process. The residual $\mathbf{r}^n$ is updated using this currently estimated signal to eliminate its contribution from the original measurement and the process repeats till the residual norm falls below a given threshold. The pseudo code of the algorithm is given in Algorithm 5. It is to be noted that it does not require knowledge of the sparsity level k to be passed as input unlike state-of-art techniques like ROMP [33], CoSaMP [34], SP [35] etc. Further, the number of clusters n_c and the corresponding grouping of $\mathbf{\Phi}$ are fixed for a specific choice of signal dimension N and number of measurements M, and remains independent of any change in the sparsity level of the underlying signal.

Algorithm 5: Sparsity Independent Regularized Pursuit (SIRP)

procedure SIRP ($\mathbf{y}$, $\mathbf{\Phi}$, $\mathbf{d}$)

Initialization $\hat{\mathbf{x}}^0$=$0_{N\times 1}$, $\mathbf{r}^0 = \mathbf{y}$, $n = 1$, $\mathbf{\Gamma}^0 = \phi$ **while** $\|\boldsymbol{r}\|_2 > T_r$ *or* $n < n_{\max}$ **do**

$J_i = \arg\max \ |\mathbf{\Phi}_{g_i}^T \mathbf{r}^{n-1}| \quad \forall \ i \in [1, n_c]$ ▷ **Identify**

Sort $\mathbf{J} = \{J_1, \cdots, J_{n_c}\}$ in descending order ▷ **Sorting**

Form disjoint subsets $\mathbf{J}_0^j$ of the sorted $\mathbf{J}$ such that for every subset $\mathbf{J}_0^j$

$|\mathbf{\Phi}_a^T \mathbf{r}^{n-1}| \geq \frac{1}{2}|\mathbf{\Phi}_b^T \mathbf{r}^{n-1}| \ \forall \ a, b \in \mathbf{J}_0^j$

Choose $\mathbf{J}_0^j$ with the highest average correlation energy

$\frac{1}{|\mathbf{J}_0^j|} \sum_{b \in \mathbf{J}_0^j} |\mathbf{\Phi}_b^T \mathbf{r}^{n-1}|^2$ ▷ **Regularize**

$\mathbf{\Gamma}^n = \mathbf{\Gamma}^{n-1} \cup \mathbf{J}_0$ ▷ **Support Set Augmentation**

$\hat{\mathbf{x}}^n_{\mathbf{\Gamma}^n} = [\mathbf{\Phi}^T_{\mathbf{\Gamma}^n} \mathbf{\Phi}_{\mathbf{\Gamma}^n}]^{-1} \mathbf{\Phi}^T_{\mathbf{\Gamma}^n} \mathbf{y}$ ▷ **Estimate Update**

$\mathbf{r}^n = \mathbf{y} - \mathbf{\Phi}_{\mathbf{\Gamma}^n} \hat{\mathbf{x}}^n_{\mathbf{\Gamma}^n}$ ▷ **Residual Update Step**

$n = n + 1$

end

return Estimated sparse signal $\hat{\mathbf{x}}^n$

end procedure

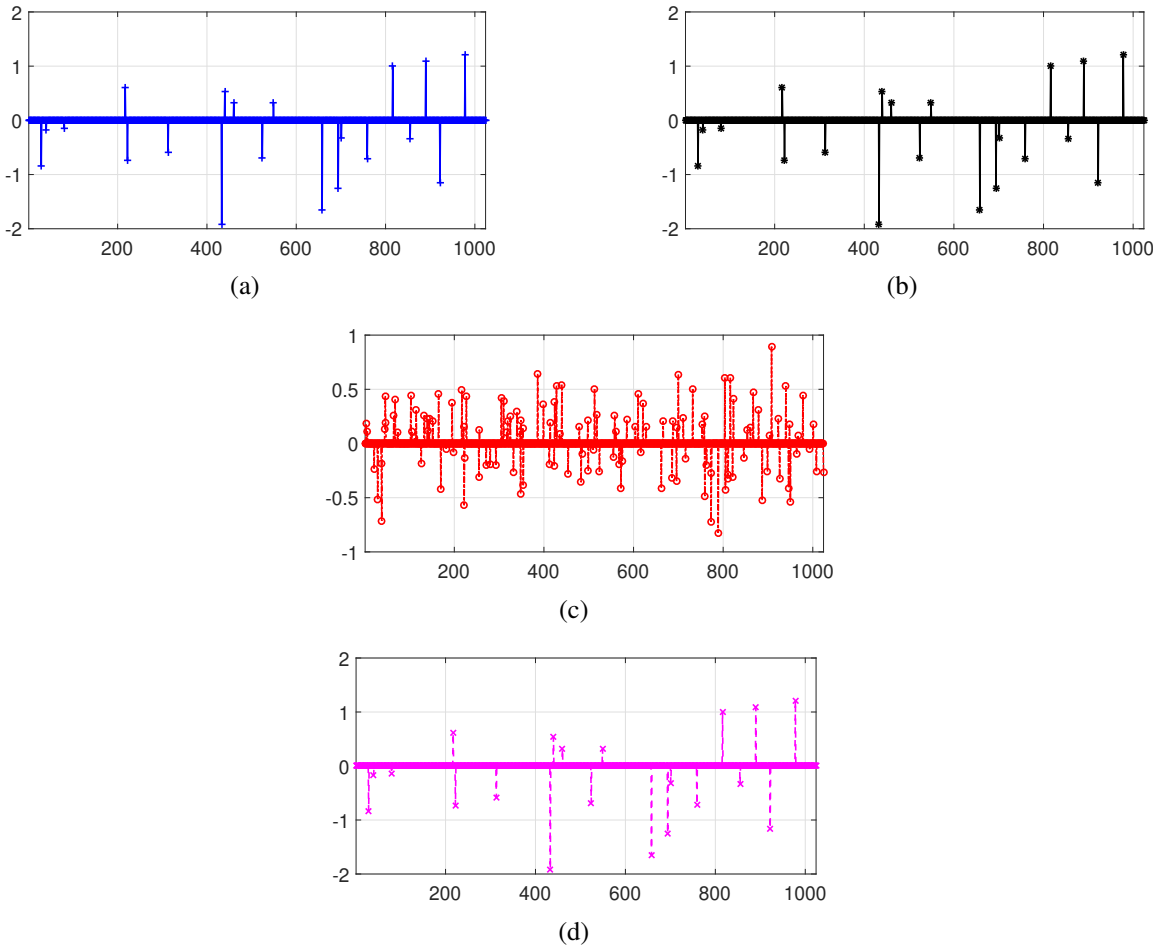


Figure 3.1: (a) Original signal (b) Reconstructed signal using ROMP given true k (c) Reconstructed signal using ROMP given $k_{\max}$ (d) Reconstructed signal using SIRP

Before proceeding further, the choice of the highest average-group-energy regularization criterion in SIRP over the maximum energy criterion employed in ROMP is vindicated using a simple experiment, as demonstrated in Fig. 3.1 where a 20-sparse 1024-length signal is recovered from 256 measurements using ROMP and SIRP. As in realistic scenarios wherein the sparsity level k cannot be known prior, the sparsity upper bound of 50 is passed as an input to ROMP. Fig. 3.1 (b) shows how ROMP wrongfully reconstructs a higher dimensional signal (which also happens to be one of the infinite solutions of the underdetermined system $\mathbf{y} = \mathbf{\Phi x}$), but SIRP successfully estimates the signal demonstrating its independence of sparsity prior knowledge.

3.5.1 Implementation and Complexity Analysis

This section deals with the implementation and running time complexity of the proposed SIRP algorithm. Unlike existing greedy methods, SIRP refines the identification stage by adopting a divide-and-conquer strategy to parallelize the complex matrix-vector multiplication and yields n_c parallel indices having the highest correlation energy within their respective groups. The inherent parallelism here make it attractive for targeting parallel hardware architectures or GPU implementations for swift completion of the correlation step. Sorting the correlation energies of the n_c parallel indices can be accomplished in $\mathcal{O}(n_c \log n_c)$, whereas the sorting step in ROMP, CoSaMP and SP costs $\mathcal{O}(N \log N)$.

The regularization process can be carried out swiftly by observing that it is sufficient to look for the optimal subset of correlation energies $\mathbf{J}_0 \subset \mathbf{J}$ among $\mathcal{O}(\log n_c)$ successive intervals whose endpoint magnitudes fall by a factor of 2 respectively. Thus, the regularization process can be completed in $\mathcal{O}(n_c)$.

In addition to these costs, all the least squares problems for the unstructured sub-matrix $\mathbf{\Phi}_{\mathbf{\Gamma}^n}$ can be completed in $\mathcal{O}(k^2 M)$ using Modified Gram Schmidt (MGS) to incrementally update Q and R matrices [75, 76] due to the regular structure of the SIRP algorithm. This is significant compared to strategies like CoSaMP and SP that have varying support sets for every iteration, which results in running time of $\mathcal{O}(k^2 M)$ in every iteration. Further, SIRP considerably brings down the number of iterations compared to OMP, which leads to tremendous savings in matrix-vector multiplications and least squares computations. The incremental QR update strategy using MGS has been adopted in the implementation of the algorithm.

When resource area is critical, the correlation and identification steps have to be performed serially for each cluster, using the same resources. In this scenario, the cluster centroid vector can be correlated with the residual to ignore the correlation and identification of certain clusters of $\mathbf{\Phi}$ whose centroid poorly correlate with the residual. An adaptive

threshold μ is adopted to select only those clusters whose centroid correlation magnitude is greater than μ times the maximum centroid correlation value. This variant of SIRP is called SIRP-II and can perform faster than the serial version of SIRP. Thus, SIRP's novel strategy of identifying support set locations based on clusters coupled with the enhanced regularization step ensure fast convergence, good scalability and robustness to noise, which is demonstrated in Section III.

3.5.2 Key insights

Convex optimization strategies are able to provide stable and uniform reconstruction guarantees, but do not possess the speed of greedy approaches. However, OMP fails to provide such uniform guarantees [77] as opposed to basis pursuit algorithms. Another weakness of the OMP despite its relative simplicity is the sequential pursuit process which forces significant amount of the reconstruction time to be consumed by the costly correlation step in high dimensional scenarios. Further, noisy conditions demand running the algorithm for considerably more iterations than the sparsity of the underlying signal [78]. There have been numerous attempts to accelerate the reconstruction process by selecting multiple indices in each iteration, but possess weaker theoretical guarantees than BP.

ROMP [33] emerged as the first multi-element pursuit that provided uniform reconstruction guarantees at complexities similar to that of OMP. However, its empirical performance was found to be weaker than OMP in certain cases [79] particularly in scenarios where the true sparsity is unknown. Parallel pursuit strategies like CoSaMP [34] and SP [35] also offer faster convergence by requiring fewer iterations, but are computationally very complex owing to least squares updates that are of the order of $3K$ and $2K$ terms in each iteration. Further, their heavy dependence on the knowledge of exact sparsity [45] for superior convergence presents a challenge in scenarios where the prior information is not available or the sparsity continuously changes.

The above discussed methods demonstrate tradeoffs between different system consid-

erations like uniform recovery, hardware feasibility, sparsity independence and reconstruction speed. SIRP promises to achieve a good balance of all these key considerations in its pursuit process by being sparsity independent, possessing uniform recovery guarantees like ROMP, hardware feasibility similar to OMP and fast convergence like the parallel strategies.

3.6 Experimental Evaluation

The performance of the proposed SIRP algorithm is compared with the state-of-art algorithms comprising OMP [7], BaOMP [36], ROMP [33], SP [35], SWOMP [32], MRAMP [38] and AMP [40] under varying experimental settings in order to comprehensively evaluate its competence. The experimental simulations are carried out in Matlab 2018a executing on Intel(R) Xeon(R) CPU at 3.7 GHz with 64-bit Windows 10 operating system.

The elements of the measurement matrix $\mathbf{\Phi} \in \mathbb{R}^{M \times N}$ are independently drawn from the standard normal distribution. The undersampling ratio given by M/N represents the level of compression in the acquisition. An N length k-sparse signal is generated by choosing the support set of size k uniformly at random from $[1, N]$ and drawing the coefficients from the Gaussian distribution. For experiments carried out in noisy environments, the measurements $\mathbf{y}$ are generated according to $\mathbf{y} = \mathbf{\Phi x} + \mathbf{w}$, where noise $\mathbf{w}$ belongs to the standard normal distribution with zero mean and standard deviation σ. This can also be realized by Matlab's inbuilt additive white gaussian noise (AWGN) function to generate noisy measurements corresponding to specified signal-to-noise ratio (SNR) levels.

The normalized root-mean-square-error (NRMSE) is an important metric used to evaluate reconstruction accuracy and is defined as

$$\text{NRMSE} = \frac{\sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \hat{x}_i)^2}}{x_{\max} - x_{\min}} \tag{3.20}$$

where $x_{\max}$ and $x_{\min}$ correspond to the largest and smallest elements in $\mathbf{x}$. The number of successful reconstructions having NRMSE below a certain threshold ϵ are averaged over the total trials to yield the percentage of successful recovery, while average recovery time is computed by averaging the running time of the respective algorithms for each case. The NRMSE threshold ϵ is set to 10^{-9} in noiseless settings, while it is set according to the perturbation level $\|\mathbf{w}\|_2$ in noisy scenarios, following the usual approach.

The parameters chosen for the algorithms are discussed here. BaOMP's parameters are set as μ_1=0.4 and μ_2=0.1 as these perform better in our experimental setup than the values in [36]. The parameters for ROMP and SWOMP can generally vary between 0 and 1, where '1' makes them effectively like OMP, while small values make them like thresholding algorithms. Hence, the parameter values for ROMP and SWOMP are set to 0.5 to achieve a trade-off between the reconstruction performance of the OMP and faster recovery time. MRAMP employs a divide-and-conquer approach to reconstruct the signal without knowledge of sparsity by using a step size parameter. Large step size can result in over-estimation or under-estimation of the signal, whereas small step size would involve higher running time. The step size is set to 20 in our simulation in order to achieve a trade-off between these two characteristics. The parameter for SIRP-II which plays a significant part in the task of bypassing groups of atoms in the correlation step that have weak centroid correlations with the residual, is set to 0.2 based on empirical analysis and is used throughout all experiments.

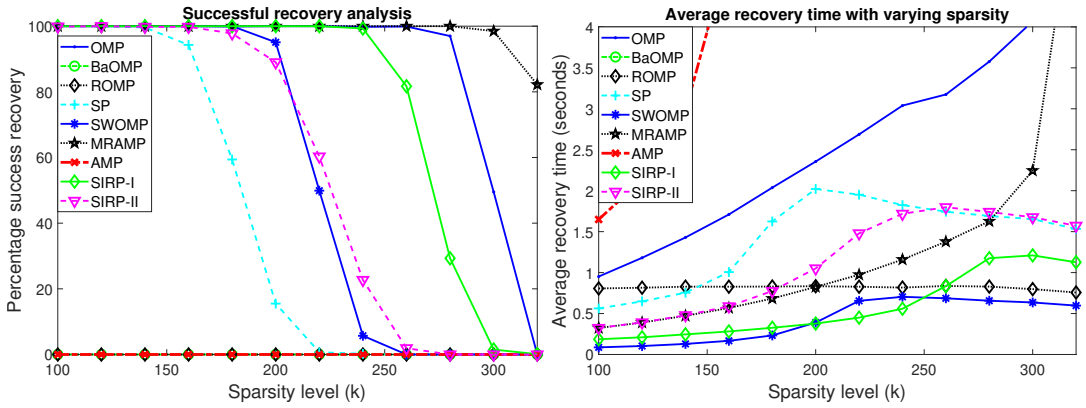
3.6.1 Recovery under noiseless scenario for varying k

In order to evaluate the performance of the algorithms for varying sparsity levels under noiseless scenarios for higher dimensions, a 1000×10000 dimensional system is realized using a random Gaussian measurement matrix $\mathbf{\Phi}$ where $N = 10^4$ and $M = 10^3$. For given sparsity level k and signal dimension N, the traditional Basis Pursuit (BP) requires just $\mathcal{O}(k \ln \frac{N}{k})$ measurements to recover the signal. While greedy algorithms theoretically

require much higher number of measurements for guaranteeing recovery, they are empirically found to perform well even with this bound. The sparsity upper bound $k_{\max}$ is the maximum sparsity level that can be supported for recovery with the given number of measurements by BP. Since $k_{\max}$ can be determined prior based on M and N, it can be used as sparsity information in cases where k cannot be known exactly or is constantly known to vary. OMP [7], ROMP [33] and SP [35] are provided with $k_{\max}$ as sparsity parameter for realistic comparison.

The true sparsity k of the signal is made to vary from 100 to 320 in steps of 20 and 1000 N-length Gaussian signals are generated as described earlier for each level of k. OMP, ROMP and SP are provided with $k_{\max} = 320$ as explained earlier. The other algorithms do not require any sparsity information. The cluster size for SIRP-I and SIRP-II is chosen to be 100 as this is found to perform well empirically (analysis included in subsection 3.7). The percentage of successful reconstructions and the reconstruction time are plotted versus the sparsity level in Figs. 3.2 (a) and (b) respectively. A reconstruction is considered successful if the NRMSE of the recovered signal lies below 10^{-9}.

It is observed that MRAMP achieves better recovery for moderately sparse signals (i.e. higher k) in this noiseless setting followed by OMP and SIRP-I as shown in Fig. 3.2 (a). MRAMP is able to achieve 100% recovery up to $k = 280$ and OMP achieves 100% sparse recovery up to $k = 260$. This can be attributed to the back-tracking nature of MRAMP which enables it to prune out wrongly selected atoms. However, MRAMP requires significant computational resources as described earlier to achieve this. Further, its weak performance in noisy settings as will be discussed in the following section discourage its applicability in realistic scenarios. The OMP implementation in this setting performs remarkably better than its existing implementations in the literature due to the fact that it is run for more iterations ($k_{\max}$) than the true sparsity k of the signal, which has also been demonstrated in [78]. SIRP-I attains 100% recovery up to $k = 220$ following which it begins to dip. SIRP-I's recovery curve is considerably better than those of SWOMP, SIRP-II,



(a) (b)

Figure 3.2: Recovery analysis of 10000-length signals under 'noiseless' conditions using 1000×10000 dimensional CS system with $k_{\max} = 320$ for OMP, ROMP and SP (a) Success rate vs true sparsity k (b) Average recovery time vs true sparsity k

SP, BaOMP, ROMP and AMP. SIRP-II performs slightly worser than SWOMP, but much better than SP. It is noticed that ROMP and AMP fail completely in this range of sparsity levels due to the unavailability of exact k and inadequate number of measurements respectively. The BaOMP strategy fails to reconstruct the signal accurately as the backtracking step prevents it from identifying few true indices, which leads to higher reconstruction errors.

Analysis from the average running time graph plotted in Fig. 3.2 (b) shows that SWOMP has fastest recovery speed followed closely by SIRP-I. MRAMP and SIRP-II take moderately higher time than SIRP-I in recovering signals with the difference becoming more pronounced as k becomes higher. SIRP-I performs significantly better than SP, OMP and AMP as observed in Fig. 3.2 (b). For instance, when $k = 160$, SIRP-I achieves a speed-up of $2\times$, $3.5\times$ and $6\times$ over MRAMP, SP and OMP respectively. The advantage of SWOMP in terms of recovery time fails in noisy settings, as will be shown later. Thus, a comprehensive evaluation of the algorithms validates the superiority of the proposed SIRP in significantly bringing down reconstruction time while achieving 100% sparse recovery up to signal sparsity of 220.

3.6.2 Analysis of the number of measurements

In order to study the impact of number of noise-free measurements M on the success rates of different algorithms for fixed sparsity levels, experiments were carried out for 10,000-length random gaussian signals for two specified values of sparsity level k. The number of measurements in the first case for k=200 is varied from 500 to 1250 and in the second case for k=400 from 1000 to 1750. The success rates of the algorithms at each of these levels averaged over 1000 trials is plotted in Figs. 3.3 (a) and (b) for $k = 200$ and $k = 400$ respectively. The algorithms which require sparsity information are provided the true value of k, since k does not vary. Fig. 3.3 (a) demonstrates that in the noiseless setting, when $k = 200$, MRAMP requires the fewest number of noise-free measurements (i.e. 850) to achieve 100% success rate, whereas SIRP-I and SP require 950. However, from the earlier analysis in noisy scenarios, it is known that MRAMP performs worse than SIRP-I in reconstructing the signals. Fig. 3.3 (a) also shows that SWOMP, SIRP-I and OMP require higher number of measurements for attaining full recovery. ROMP requires much higher measurements to guarantee exact sparse recovery. Similarly, when sparsity k=400, MRAMP requires 1350 measurements to guarantee 100% recovery whereas SIRP-I requires 1450 measurements, as shown in Fig. 3.3(b). The success rate curves of the other algorithms follow similar trends observed earlier with OMP not able to achieve 100% recovery with 1750 measurements.

3.6.3 Recovery under noisy scenario for varying k

In order to study the performance of the algorithms in varying sparsity conditions for similar data sizes under noisy settings, N and M are retained as 10,000 and 1,000 respectively. As in the earlier case, N-length Gaussian signals having sparsity k varying from 20 to 240 in steps of 20 are generated. The 1000×10000 random Gaussian measurement matrix is generated as earlier, which is multiplied with each signal to obtain the respective noise-free

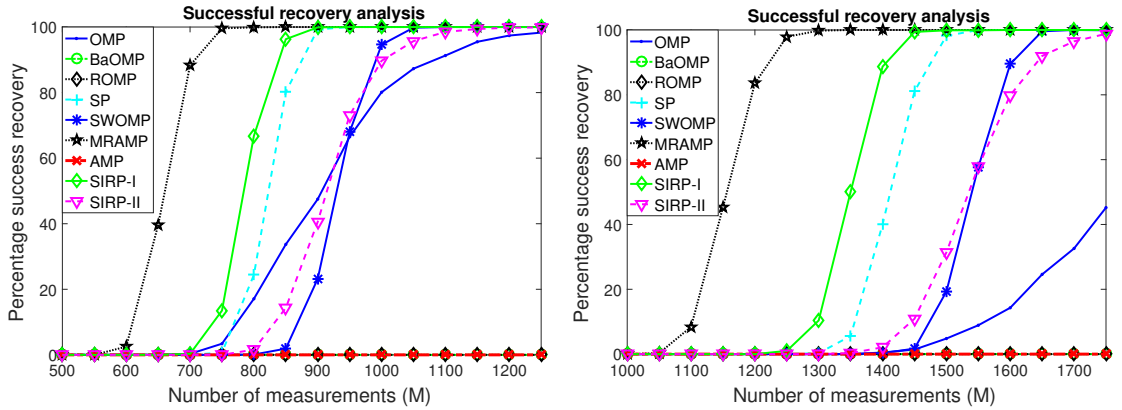
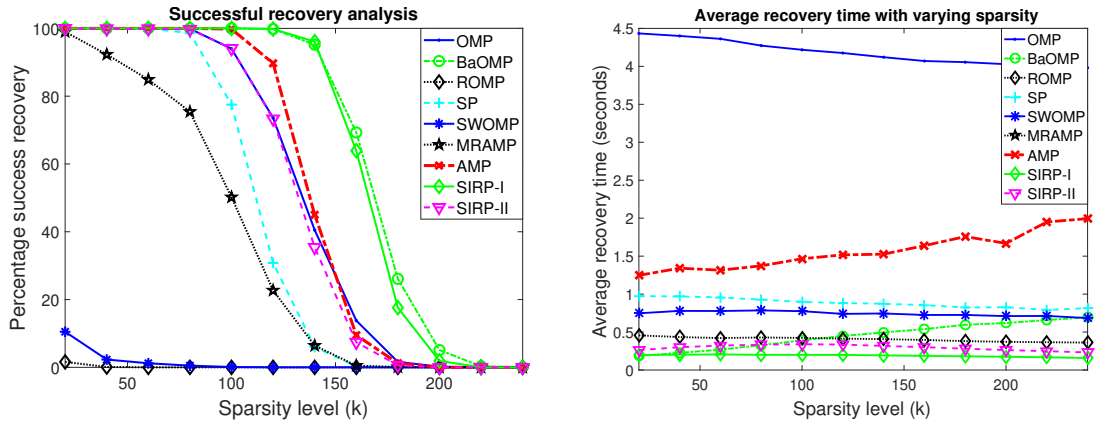


Figure 3.3: Recovery analysis of the number of measurements M to reconstruct 10000-length signals under noiseless conditions for fixed sparsity values (a) Success rate for k=200 (b) Success rate for k=400

measurements. These are corrupted by AWGN such that the SNR of the resulting noisy measurements is 10dB. The percentage of successful reconstructions and the average recovery time over 1000 trials is plotted versus the different sparsity levels in Fig. 3.4 (a) and (b) respectively. Due to high noise levels, the NRMSE threshold is relaxed to 10^{-2} to permit approximate sparse recovery.

It is observed from Fig. 3.4(a) that SIRP-I and BaOMP perform best in this noisy scenario than their competitors. It is seen that MRAMP's performance drastically degrades in the noisy framework. SIRP-II and OMP have almost similar success rate curves which are better than that of SP and worse than that of AMP. An analysis of the running time graph plotted against sparsity k reveals that SIRP-I retains its time efficiency and remains almost constant for the chosen range of sparsity. This can be contrasted with the running time of BaOMP which increases with higher k. SIRP-I improves the reconstruction time by factors of roughly $20\times$, $7\times$ and $5\times$ over OMP, AMP and SP respectively.

Fig. 3.5 shows the corresponding graphs for similar experiments carried out for relatively less noisy environments with measurements having SNR of 30dB. Fig. 3.5(a) depicts the efficiency of the proposed algorithm in this scenario as well, with SIRP-I and SIRP-II outperforming the other algorithms. BaOMP now fails within the chosen range of spar-



(a) (b)

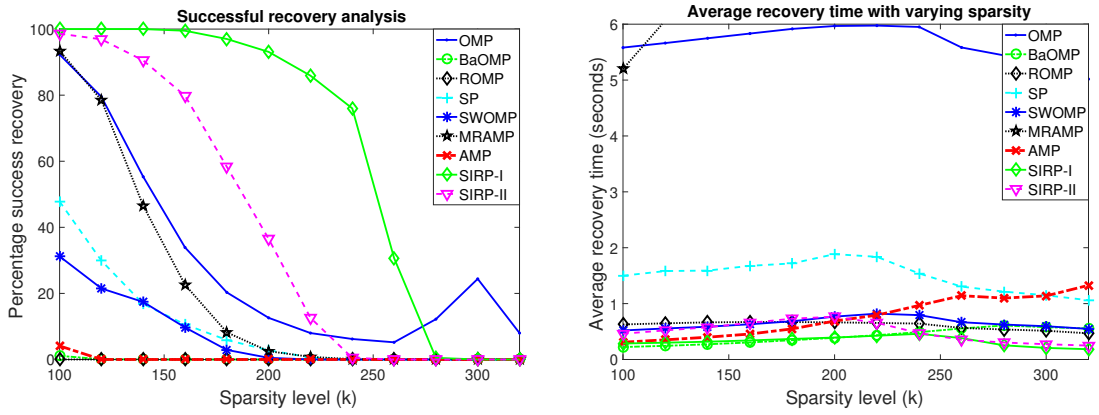
Figure 3.4: Recovery of 10,000-length signals from 1,000 noisy measurements with SNR of 10dB (a) Successful recovery curve (b) Average running time

sity similar to its performance in the noisy setting. The running time graph offers similar conclusion to the time efficiency of the proposed algorithm.

3.6.4 Recovery under varying noisy scenarios

To highlight the performance of the algorithms under various noisy environments, noise-free measurements generated by multiplying the 1000×10000 random Gaussian measurement matrix with 10000-length signals of fixed sparsity, are corrupted by additive white gaussian noise (AWGN) to obtain noisy measurements with a specified signal-to-noise ratio (SNR). The experiment is set up to measure the percentage of successful recovery versus SNR, averaged over 1000 trials, for three distinct sparsity levels of 50, 100 and 200. A trial is regarded as successful if its NRMSE falls below a threshold of 10^{-2}.

Fig. 3.6(a) depicts the performance of the algorithms when $k = 50$ for SNR levels varying from 0dB to 8dB. It is observed that SIRP-I, SP and OMP perform remarkably well achieving 100% sparse recovery from noisy measurements having SNR upwards of 3.5dB, demonstrating their robustness in low SNR regimes. It is seen from Figs. 3.6(b) and 3.6(c) that as the sparsity level increases, slightly higher levels of SNR are necessary to ensure 100% recovery if the number of measurements remains fixed. For instance in Fig.



(a) (b)

Figure 3.5: Recovery of 10,000-length signals from 1,000 mild noisy measurements with SNR of 30dB (a) Successful recovery curve (b) Average running time

3.6(b), when $k = 100$, SIRP-I achieves 100% from SNR of 7dB closely followed by OMP and SP. When $k = 200$, OMP and BaOMP attain 100% recovery at SNR of 14dB whereas SIRP-I achieves it at 16dB. Increasing the number of measurements can drive down the required SNR levels for higher k values.

3.6.5 Phase transition

Phase transition (PT) diagrams were first articulated in the field of CS by Donoho and Tanner [80] to inspect the success of CS applications in certain settings. Each point in a PT diagram represents a concrete setting of the parameters, namely, signal dimensions N, number of measurements M and sparsity level k. Every point reflects the probability of success of the reconstruction process for the specific N, M and k by measuring the number of faithful reconstructions of k-sparse vectors $\mathbf{x} \in \mathbb{R}^N$ when it is provided M linear measurements corresponding to $\mathbf{y} = \mathbf{\Phi x}$, where $\mathbf{\Phi} \in \mathbb{R}^{M \times N}$. The y-axis can represent number of measurements M, whereas the x-axis represents the sparsity k. The PT diagram comprises three main regions: low success denoted by blue, high success denoted by red and the thin transition region between them.

Fig. 3.7(a) depicts the phase transition of the proposed SIRP-I algorithm for chosen

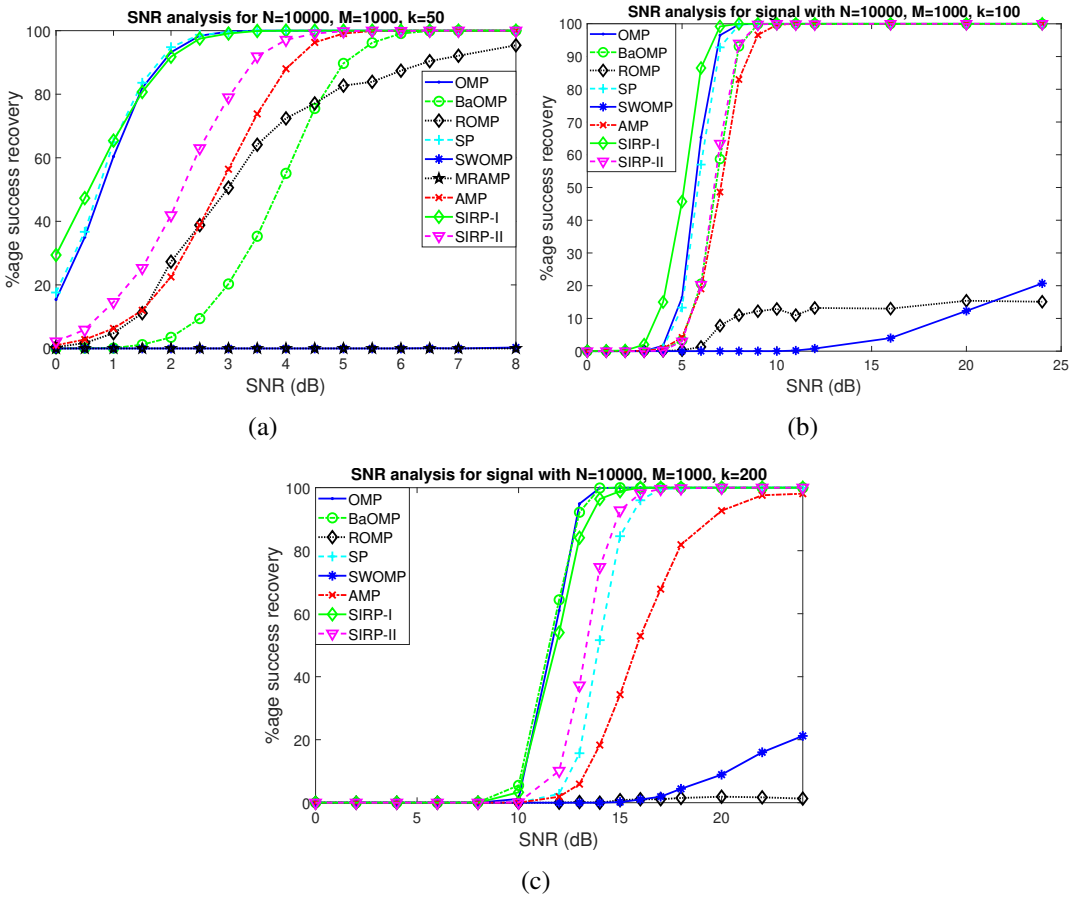
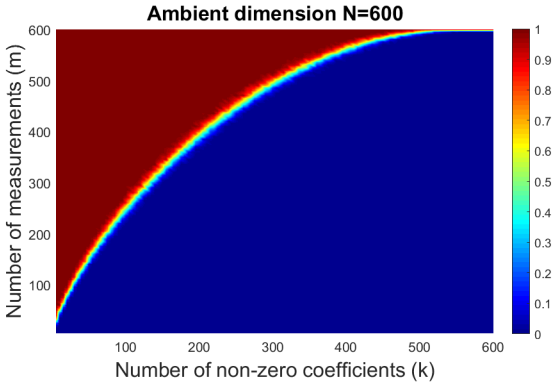
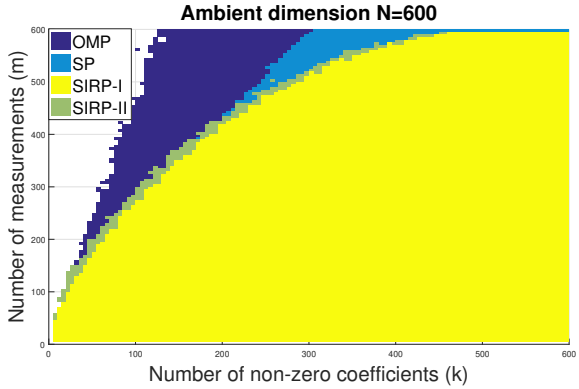


Figure 3.6: SNR analysis



(a) Phase transition width of the SIRP-I algorithm



(b) Phase transition curves

Figure 3.7: 3.7(b) Phase transition curve of the SIRP-I algorithm 3.7(a) Phase transition curves for OMP, SP, SIRP-I and SIRP-II algorithms

signal dimensionality $N = 600$ and M as well as k are made to vary from 1 to 600 in steps of 5. To measure the probability of successful reconstructions at each (k,M), 100 independent trials are carried out. Fig. 3.7(a) shows a sharp transition between regions of high and low success probabilities as expected [80] and indicates concrete settings (i.e. M and k) for guaranteed success of the algorithm in CS applications.

Fig. 3.7(b) shows the phase transition curves for OMP, SP, SIRP-I and SIRP-II for similar signal dimensionality N=600, averaged over 100 trials, to contrast the state-of-art greedy algorithms with the proposed SIRP variants. For each algorithm, area enclosed by the corresponding curve represents regions of its certain failure, whereas the other region corresponds to its certain success. It is observed that the region of success is highest for SIRP-I followed by SIRP-II, SP and OMP.

3.6.6 Recovery under noisy high dimensional settings

In order to validate the performance of the algorithm in high dimensional settings, this section deal with experiments conducted for signals with dimension $N = 1,000,000$ and 10,000 noisy measurements generated using random gaussian measurement matrix of appropriate dimensions, such that the SNR of measurement vector $\mathbf{y}$ is 10dB. The sparsity k is made to vary from 100 to 600 and the sparsity upper bound is set as $k_{\max} = 1000$ for algorithms requiring information of k, to realize scenarios where k cannot be known exactly. The number of clusters parameter n_c for the proposed SIRP-I is set to 500 in order to deal with the huge N. The performance of SIRP-I in terms of percentage of successful reconstructions and average recovery time is compared with that of OMP and SP in Table 3.1 to motivate the benefit of SIRP-I in high dimensional settings.

It is observed from Table 3.1 that SIRP-I achieves 100% success rate upto a sparsity of 400, while OMP and SP attain only 20% and 50% recovery at the same level respectively. Further, SIRP-I achieves remarkable improvement in reconstruction speed over OMP and SP by factors of approximately 60$\times$ and 2.5$\times$ respectively. The experimental simulation justifies the large dimensional applicability of the proposed algorithm owing to its superior reconstruction performance and running time efficiency.

3.6.7 Analysis on the number of clusters

In order to analyze the impact of the number of clusters n_c on the performance of the proposed SIRP-I algorithm, this section performs experiments on reconstruction of 10,000-dimensional signals from 1,000 compressed measurements with sparsity k varying from 200 to 320. It should be noted that SIRP-I achieves full recovery for lower sparsity values as well, but the range 200 to 320 is chosen to clearly contrast the performance of the algorithm for different cluster number choices. Fig. 3.8(a) shows the success rate of the proposed algorithm for four different choices of the number of clusters parameter n_c (viz.

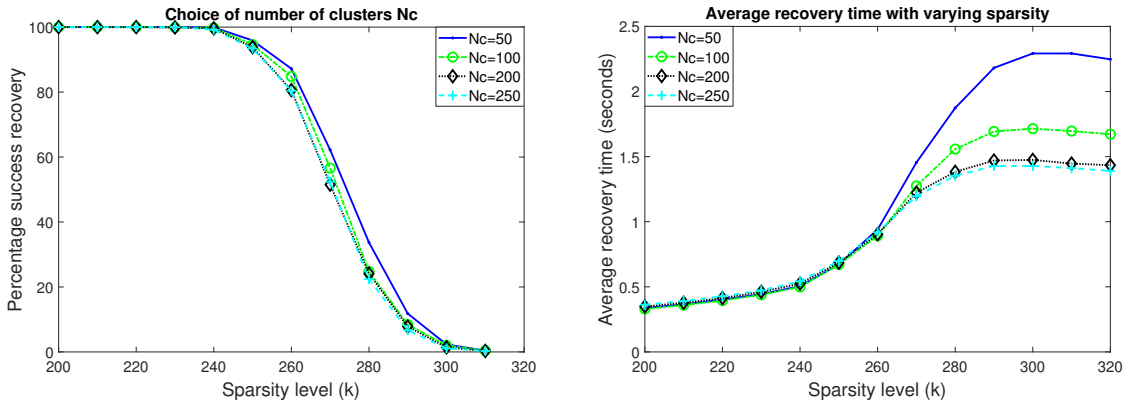
True Sparsity k	% Success			Average recovery time (sec)		
	OMP [7]	SP [35]	SIRP-I	OMP [7]	SP [35]	SIRP-I
100	100	100	100	1856.5	84.6	29.0
150	100	100	100	1857.4	84.0	31.4
200	90	100	100	1854.2	88.7	32.0
250	80	90	100	1859.4	86.6	33.4
300	60	80	100	1905.1	82.3	32.8
350	60	70	100	1863.2	83.0	36.3
400	20	50	100	1903.1	91.6	35.6
450	10	0	50	1865.2	87.4	32.1
500	10	0	30	1904.0	82.0	36.5
600	0	0	0	1904.0	67.0	34.4

Table 3.1: Comparison of OMP, SP and SIRP-I in recovery of 1000000-length signals using 10000$\times$1000000 dimensional CS system with $k_{\max}$=1000 for OMP and SP in terms of success percentage and average recovery time over 100 trials

50, 100, 200 and 250). It is observed that the curves almost coincide with $n_c = 50$ slightly performing better than $n_c = 100$, $n_c = 200$ and $n_c = 250$ respectively. For instance, when sparsity is 260, $n_c = 50$ attains 87.2% recovery followed closely by $n_c = 100$ with 85%, $n_c = 200$ with 80.6% and $n_c = 250$ with 80.4% successful recovery. Fig. 3.8(b) plots the average recovery time versus the sparsity k. It is observed that upto $k = 240$ where all curves attain 99% recovery, the running times are almost similar with $n_c = 100$ achieving the least average recovery time in all the cases. As sparsity increases and the success rate falls down, it is observed that $n_c = 250$ requires lower time to terminate than $n_c = 200$, $n_c = 100$ and $n_c = 50$ respectively.

3.6.8 Applications in compressive imaging

In order to analyze the reconstruction efficiency of the proposed algorithm in a real setting, experiments are carried out on binary and natural images with significant dimensionality.



(a) (b)

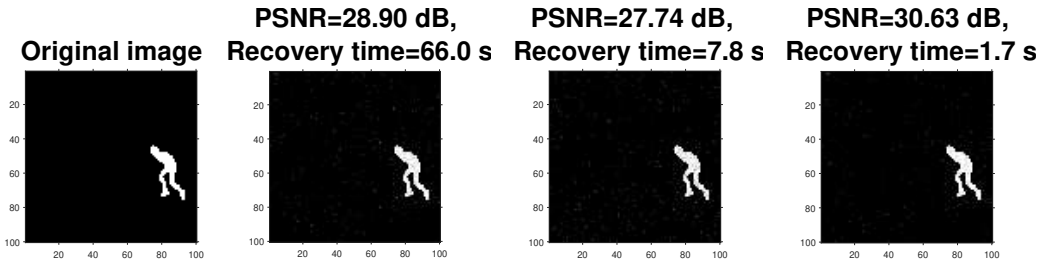
Figure 3.8: Recovery of 10,000-length signals from 1,000 noisy measurements for varying k and different choice of the number of clusters parameter n_c (a) Success rate curve (b) Average running time

3.6.8 Binary image recovery

Fig. 3.9(a) depicts a 100×100 black and white image of a person and recovery of the vectorized $10,000 \times 1$ signal from noisy projections onto a random Gaussian matrix $\mathbf{\Phi} \in \mathbb{R}^{2000\times 10000}$ is attempted using OMP[7], SP[35] and the proposed SIRP-I (noise standard deviation $\sigma = 0.05$). The sparsity level $k_{\max}$ for OMP and SP is set to 900, as discussed in Section 3.1, such that $M = \mathcal{O}(k \ln N/k)$. The cluster number parameter n_c for SIRP-I is set to 200, as this has similar performance to the case when $n_c = 100$ (from Fig. 3.8). Their respective reconstructed images as well as the peak signal-to-noise ratio (PSNR) metric and running times are shown in Figs. 3.9(b), 3.9(c) and 3.9(d). It is observed that SIRP-I achieves approximately 2dB improvement in PSNR over OMP in roughly $1/40^{\text{th}}$ of the time required by OMP. SIRP-I also achieves approximately 3 dB gain in PSNR over SP and is $5\times$ faster than SP.

3.6.8 Natural image recovery

CS theory facilitates recovery of compressible vectors as well, which implies that, $\mathbf{x}$ need not be exactly sparse, but can be compressed using a suitable orthogonal transform $\mathbf{\Psi}$



(a) Person (b) (c) (d)

Figure 3.9: Reconstructed 100×100 images of person using (b) OMP (c) SP (d) SIRP-I from 2000 measurements with $k_{\max} = 900$ for OMP and SP

such that in $\mathbf{x} = \mathbf{\Psi z}$, $\mathbf{z}$ consists of only few significant coefficients. As natural images are known to be sparsified by orthogonal transformations like Discrete Cosine Transform (DCT), Wavelets etc., these can be employed to test the efficiency of the algorithm in a realistic setting. Compressed measurements are generated by stacking the columns of the $q \times q$ sized image and projecting the resultant $q^2 \times 1$ vector onto a random Gaussian measurement matrix of size $p \times q^2$ where $p < q^2$. From the perspective of the algorithm which receives these measurements i.e. $\mathbf{y} = \mathbf{\Phi\Psi z}$, it has to decode the compressible vector $\mathbf{z}$ using $\mathbf{y}$ and the overall sensing matrix $\mathbf{\Omega} = \mathbf{\Phi\Psi}$. Finally, the image can be reconstructed by reshaping $\hat{\mathbf{x}} = \mathbf{\Psi}\hat{\mathbf{z}}$ back to the image dimensions.

The experiment in Fig. 3.10 follows this approach to compressively sample the 128×128 *Lena* image 3.10(a) by 50% and employs the DCT basis to realize $\mathbf{\Psi}$. The sparsity level $k_{\max}$ for OMP and SP is set to 1500, as described earlier and the cluster number n_c is taken to be 128. Reconstructions using OMP, SP and SIRP-I are shown in Figs. 3.10(b), 3.10(c) and 3.10(d) respectively. The structural similarity index measure (SSIM) metric is also used to measure the reconstructed image quality with reference to the original image, with 1 indicating the highest similarity. The given recovery times indicate the time taken by the respective algorithm to reconstruct $\hat{\mathbf{z}}$. It is observed that SIRP achieves a tremendous improvement in recovery speed over OMP and SP by factors of approximately 60 and 3 respectively with marginal improvement in PSNR as well.

In order to study the impact of the $k_{\max}$ values on the performance of the algorithms,

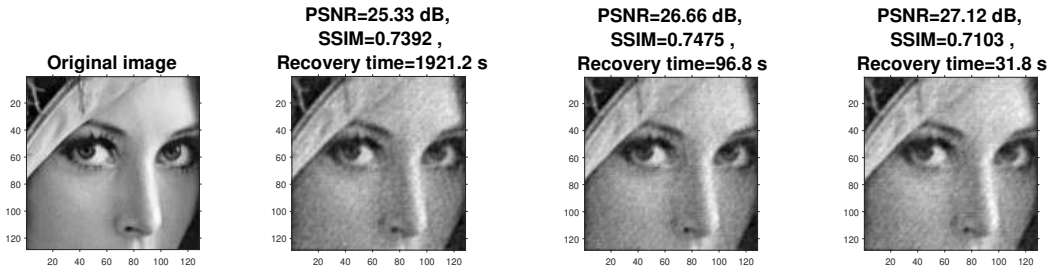


Figure 3.10: Reconstructed 128×128 'Lena' images using (b) OMP (c) SP (d) SIRP-I from 50% compressed measurements, setting $k_{\max}$=1500

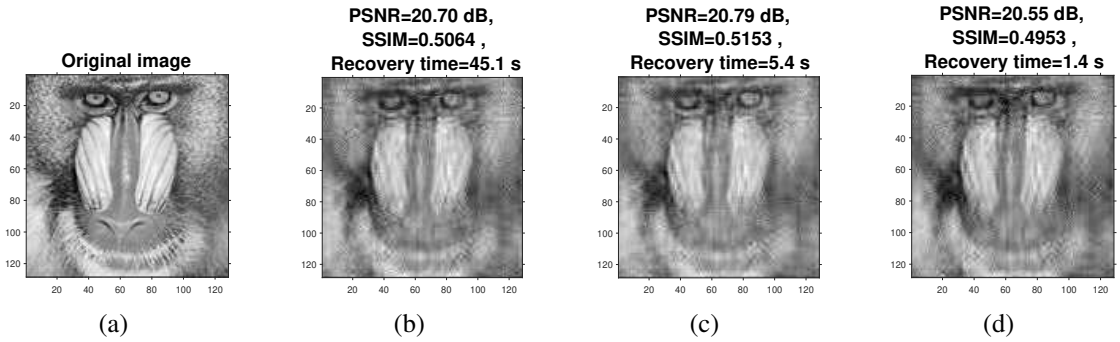
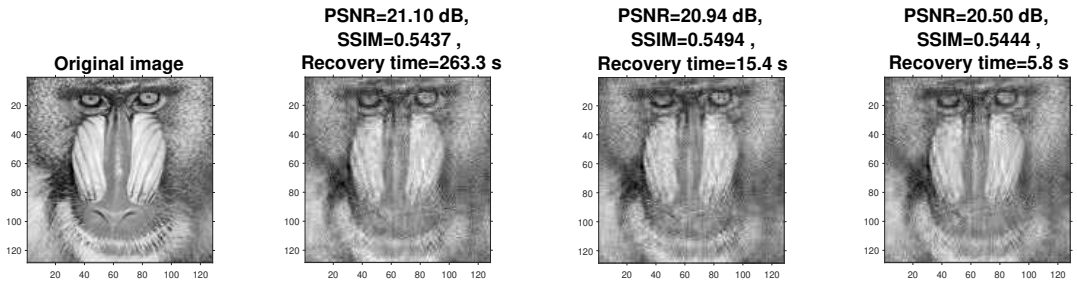


Figure 3.11: Reconstructed 128×128 'Mandrill' images using (b) OMP (c) SP (d) SIRP-I from 50% compressed measurements, setting $k_{\max}$=500

the experiments in Figs. 3.11, 3.12 employ 50% undersampling and the Discrete Wavelet Transform (DWT) based on Daubechies db10 wavelets to realize $\mathbf{\Psi}$. The reconstructions of the 128×128 standard *Mandril* image 3.12(a) using OMP, SP and SIRP-I for $k_{\max} = 500$ are shown in Figs. 3.11(b), 3.11(c) and 3.11(d) respectively. SIRP-I again achieves remarkable improvement in recovery speed over OMP and SP with comparable PSNR and SSIM. The reconstructions of OMP, SP and SIRP-I when $k_{\max}$ is set to 1000 are depicted in Figs. 3.12(b),3.12(c) and 3.12(d), which show slightly reduced visual distortions than oberved in Fig. 3.11. This is evidenced by the fact that the SSIMs of the respective reconstructions by OMP, SP and SIRP-I improve from the earlier case. SIRP-I maintains its running time advantage over OMP and SP in this case as well.



(a) (b) (c) (d)

Figure 3.12: Reconstructed 128×128 'Mandrill' images using (b) OMP (c) SP (d) SIRP-I from 50% compressed measurements, setting $k_{\max}$=1000

3.7 Summary

In this work, the SIRP algorithm is proposed which employs a novel identification and enhanced regularization step to considerably speed up pursuit while ensuring recovery performance at par with the state-of-art algorithms. The rigorous experimental evaluation establishes its independence of the sparsity prior, robustness in noisy frameworks, significant speed-up and feasibility for real world problems. These merits pave the way for realizing efficient hardware implementations in the future that can offer swifter off-line recovery for real-world applications.

Chapter 4

High throughput architecture for sparsity independent regularized pursuit

The motivation behind work presented in this chapter is to evaluate the VLSI design aspects of the sparse reconstruction engine for the sparsity independent regularized pursuit proposed in Chapter 3. An algorithm-architecture co-design strategy is adopted in order to induce hardware specific optimizations that allows for lower resource burden and reuse wherever possible. The reformulated algorithm is evaluated in terms of its computational complexity, experimental reconstruction performance and profiling results. Further to this, a detailed architecture design is presented encompassing the crucial steps of the algorithm and finally the implementation results on FPGA and ASIC are discussed.

4.1 Reformulated algorithm

SIRP employs parallel index selection and an effective regularization procedure to speed up the reconstruction process. To reduce the hardware burden, the regularization strategy and the LS update steps in SIRP are simplified to give the improved SIRP algorithm. As shown in Algorithm 1, SIRP clusters the columns of $\mathbf{\Psi}$ into c disjoint clusters by any random or ordered clustering (denoted by $\mathbf{\Psi}^1, ..., \mathbf{\Psi}^c$) and finds the index J_i within each cluster that possesses the highest correlation with the residual (step 3). Disjoint sets of the cluster maximums are constructed based on the overall maximum as in step 4, which can be done by simple shift and comparison operations. Cluster maximums with values less than $1/16$ times the overall correlation maximum will be allotted to the final $\mathbf{\Upsilon}^5$, as they are highly

unlikely to be part of the true support. A maximal energy per index-based criterion is then employed on these subsets $\boldsymbol{\Upsilon}^p$ to choose the optimal set $\boldsymbol{\Pi}$ to be added to the estimated support set $\boldsymbol{\Gamma}^n$.

Algorithm 6: Improved SIRP with incremental QRD

Input: $\mathbf{y}$, $\boldsymbol{\Psi}$
Output: $\hat{\mathbf{z}}$
1 **Initial**: $\mathbf{r}^0 = \mathbf{y}$, $n = 1$, $\boldsymbol{\Gamma}^0 = \emptyset$, $\boldsymbol{Q}^0 = \emptyset$, $\boldsymbol{R}^0 = \emptyset$, $t = 0$
2 **while** $\|\boldsymbol{r}\|_2^2 > T_r$ *or* $t \leq k_{\max}$ **do**
3 $\quad J_i = \arg\max_j | < \boldsymbol{\Psi}_j^i, \mathbf{r}^{n-1} > | \quad \forall\ i \in \{1, ..., c\}$
4 $\quad$ Form disjoint subsets $\boldsymbol{\Upsilon}^p$ of $\mathbf{J} = \{J_1, \cdots, J_c\}$ such that for every subset $\boldsymbol{\Upsilon}^p$ where $p = 1, ..., 5$ $|\boldsymbol{\Psi}_a^T \mathbf{r}^{n-1}| \geq \frac{1}{2^p} \max |\boldsymbol{\Psi}^T \mathbf{r}^{n-1}| \quad \forall\ a \in \boldsymbol{\Upsilon}^p$
5 $\quad$ Choose $\boldsymbol{\Upsilon}^p$ with the highest average correlation energy as optimal set $\boldsymbol{\Pi}$
6 $\quad \boldsymbol{\Gamma}^n = \boldsymbol{\Gamma}^{n-1} \cup \boldsymbol{\Pi}$
7 $\quad$ **for** *l=1 to* $|\boldsymbol{\Pi}|$ **do**
8 $\quad\quad [\boldsymbol{Q}^{t+l}, \boldsymbol{R}^{t+l}] = MGS(\boldsymbol{\Psi}_{\Pi_l}, \boldsymbol{Q}^{t+l-1}, \boldsymbol{R}^{t+l-1})$
9 $\quad\quad \mathbf{r}^{t+l} = \mathbf{r}^{t+l-1} - (\boldsymbol{Q}^{t+l})(\boldsymbol{Q}^{t+l})^T \mathbf{r}^{t+l-1}$
10 $\quad$ **end**
11 $\quad t = t + |\boldsymbol{\Pi}|$
12 $\quad n = n + 1$
13 **end**
14 $\hat{\mathbf{z}} = \boldsymbol{R}^{-1} \boldsymbol{Q}^T \mathbf{y}$

The newly selected columns of $\boldsymbol{\Psi}$ in each iteration $\boldsymbol{\Psi}_{\boldsymbol{\Pi}}$ are successively passed through an MGS procedure and a residual update step shown in steps 8 and 9 of Algorithm 1. Since the updated columns in $\boldsymbol{Q}$ can be used to determine the new residual, the partial estimation of $\hat{\mathbf{z}}$ is avoided until the algorithm terminates. These incremental updates however require storage of the $\boldsymbol{Q}$ and $\boldsymbol{R}$ matrices for subsequent computations. Estimation of the signal can be performed after iterations have terminated by solving the triangular system of equations given by $\boldsymbol{R}\hat{\mathbf{x}} = \boldsymbol{Q}^T\mathbf{y}$. The algorithm is terminated either when the squared norm of the residual falls below a threshold T_r or when the estimated support set size reaches a maximum $k_{\max}$.

OMP [7], SIRP [81], improved SIRP, SE-SP [71] and ADMM [72] are decomposed into different computing steps and the arithmetic operations of each step in the i^{th} are tabulated

Table 4.1: Computational Complexity comparison of the i^{th} iteration of OMP, SIRP, Improved SIRP, SE-SP and ADMM (M is the number of measurements, N is the signal size, k is the sparsity level and $|\Gamma|^n$ is the support set size in the i^{th} iteration of SIRP)

Algorithm	**Steps**		**Multiplications**	**Additions**
OMP	Index searching		MN	$(M-1)N$
	Least squares	$C = {\mathbf{\Psi}_{\Gamma^n}}^T \mathbf{\Psi}_{\Gamma^n}$	Mi	$(M-1)i$
		C^{-1}	$i(2i-1)-1$	$3(3i+1)(i-2)$
		$\hat{\mathbf{z}}$	$(i+1)(i-1)$	$\frac{1}{2}i(i-1)$
	Residual update		Mi	Mi
SIRP	Index searching		MN	$(M-1)N$
	Least squares	$C = {\mathbf{\Psi}_{\Gamma^i}}^T \mathbf{\Psi}_{\Gamma^i}$	$M\|\Gamma^i\|$	$(M-1)\|\Gamma^i\|$
		C^{-1}	$\|\Gamma^i\|(2\|\Gamma^i\|-1)-1$	$3(3\|\Gamma^i\|+1)(\|\Gamma^i\|-2)$
		$\hat{\mathbf{z}}$	$\|\Gamma^i\|^2-1$	$\frac{1}{2}\|\Gamma^i\|(\|\Gamma^i\|-1)$
	Residual update		$M\|\Gamma^i\|$	$M\|\Gamma^i\|$
Improved SIRP	Index searching		MN	$(M-1)N$
	MGS		$4M+2M\|\Gamma^i\|$	$2(M-1)+(2M-1)\|\Gamma^i\|$
	Residual update		$2M\|\Gamma^i\|$	$(2M-1)\|\Gamma^i\|$
SE-SP [71]	Phase 1	Index search	MN	$(M-1)N$
		l_2 min	$2Mi+i$	$(M+1)i$
		Residual update	Mi	M
	Phase 2	Index search	MN	$(M-1)N$
		l_2 min	$2Mk+k$	$(M+1)k$
		Residual update	Mk	M
ADMM [72]	Sparse coefficient update		N^2	$2N$
	Primal variable update		-	N
	Dual variable update		-	$2N$

in Table 4.1. The index searching operation in OMP, SIRP, Improved SIRP and SE-SP costs MN multiplications every iteration. Since OMP is serial, it would require at least k iterations to complete the reconstruction process. SE-SP performs sparsity estimation similar to OMP in the first phase which requires k iterations while the second phase of subspace pursuit based fine reconstruction can be accomplished in much fewer iterations. SIRP require fewer overall iterations for reconstructions as can be seen later from Fig. 4.3.

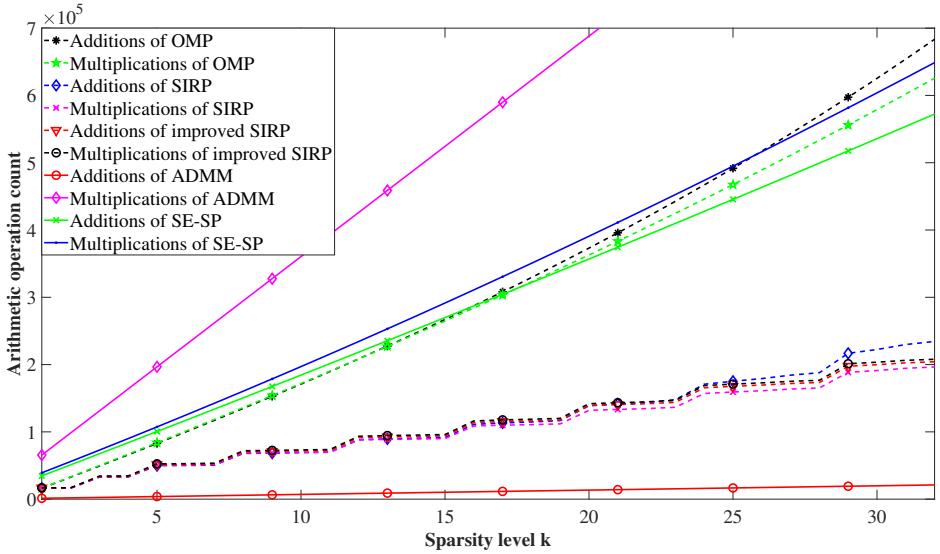


Figure 4.1: Growth trend of the number of multiplications and additions versus the sparsity level

This contributes to corresponding slower growth rates of the number of arithmetic operations for higher sparsity levels compared to OMP. The LS update in each iteration of OMP and SIRP involve computing the inverse of C as seen in Table 1 which is quadratic in i and consequently has a complexity of $\mathcal{O}(i^3)$ over all iterations. In the improved SIRP, the LS update is substituted with a QRD which is linearly proportional to the number of selected columns denoted by $|\Gamma^i|$ and thus has a worst-case complexity of $\mathcal{O}(i^2)$ over all iterations.

4.2 Performance evaluation of the improved SIRP algorithm

Numerical simulation was carried out using MATLAB 2018a running on an Intel Xeon(R) CPU at 3.7 GHz to observe the performance of the improved SIRP algorithm with respect to OMP and SP. The experiment was set up for a sparse signal of dimensionality N=1024 and compression ratio of 0.25 such that M=256. The support set of the non-zero coefficients of each signal is randomly chosen and random Gaussian entries are used to build the measurement matrix $\mathbf{\Phi}$ while the non-zero coefficients belong to the normal distribution. The number of clusters c is analysed empirically in terms of reconstruction performance and running time, and found to be optimal at 32 for the given problem setting. The percentage of successful reconstructions over all trials is measured as the success rate of the algorithm.

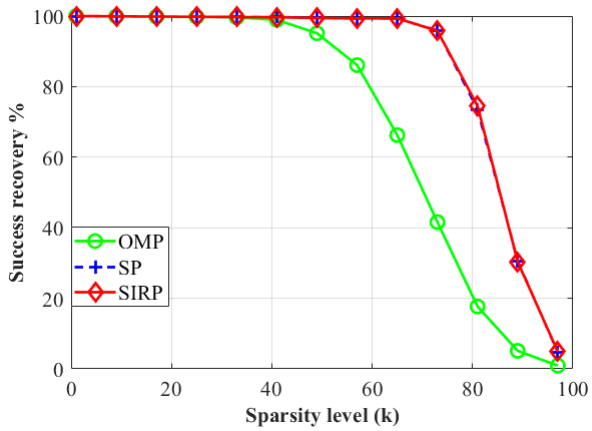


Figure 4.2: Success rate comparison of OMP, SP and improved SIRP

It can be seen from Fig. 4.2 that the improved SIRP performs better than OMP and at par with SP with increasing sparsity levels. It should be noted that the par performance of the improved SIRP with SP is achieved at significantly lower computational costs. In order to compare the efficiency of the iterations of each algorithm, the average support cardinality error (ASCE) metric is plotted versus the iteration number for 10 dB, 20 dB and

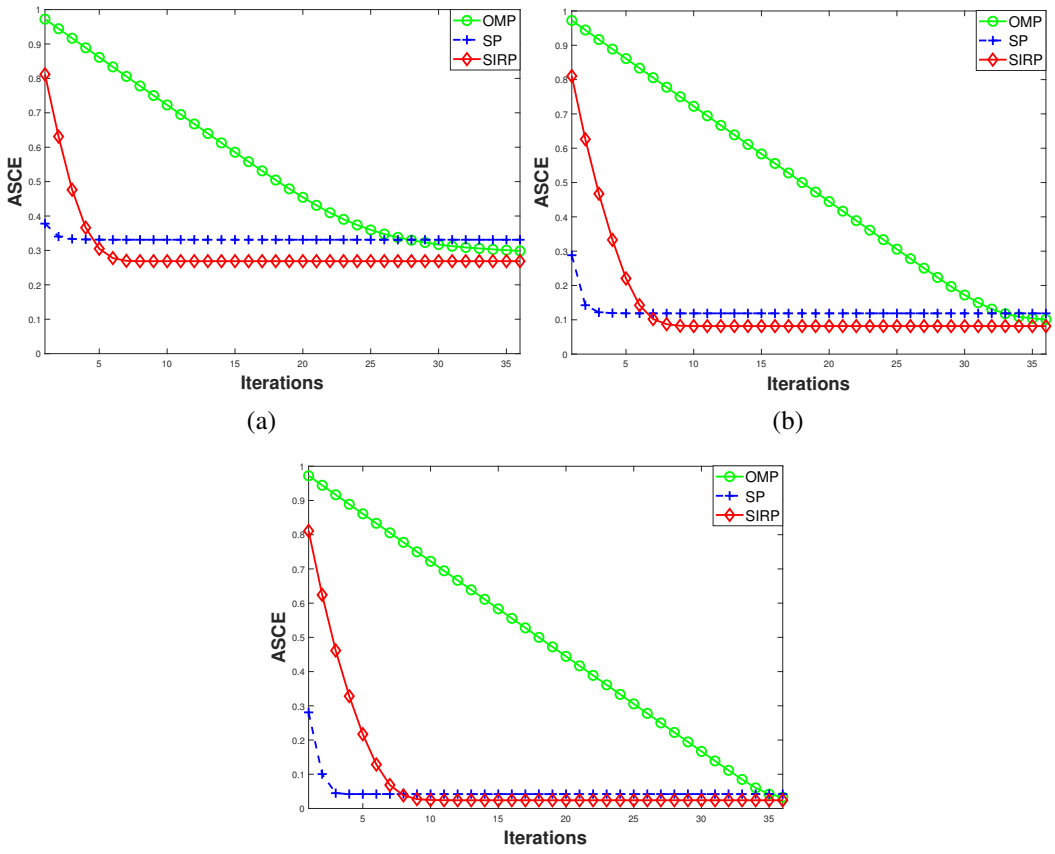


Figure 4.3: ASCE versus iterations for (a) 10 dB (b) 20dB and (c) 30 dB noisy measurements

30 dB noisy measurements in Fig. 4.3 with the residual threshold for all the algorithms set to 10^{-2}. It is seen that compared to OMP, SP and the improved SIRP detect bulk of the true support in far fewer iterations. Though SIRP requires relatively more iterations than SP, it is computationally much less complex than SP. This validates the improved SIRP as an ideal choice for hardware implementations of CS based signal recovery.

In order to evaluate the performance of the algorithm in a real world setting, the problem of recovering electrocardiogram (ECG) signals from compressed measurements is considered. The MIT-BIH Arrhythmia database [82] containing 30 minute ambulatory ECG recordings sampled at 360 Hz with 11-bit precision over a 10mV range is used for this experiment. Fig. 4.4 (a) depicts a 14 second ECG recording of patient record 100 in the

database which is divided into 20 segments of length 512. In contrast to the previous experiment, ECG signals are not sparse in time, but can be sparsely represented in the wavelet domain. The sensing matrix therefore corresponds to the product of the random Gaussian measurement matrix and the Haar wavelet basis.

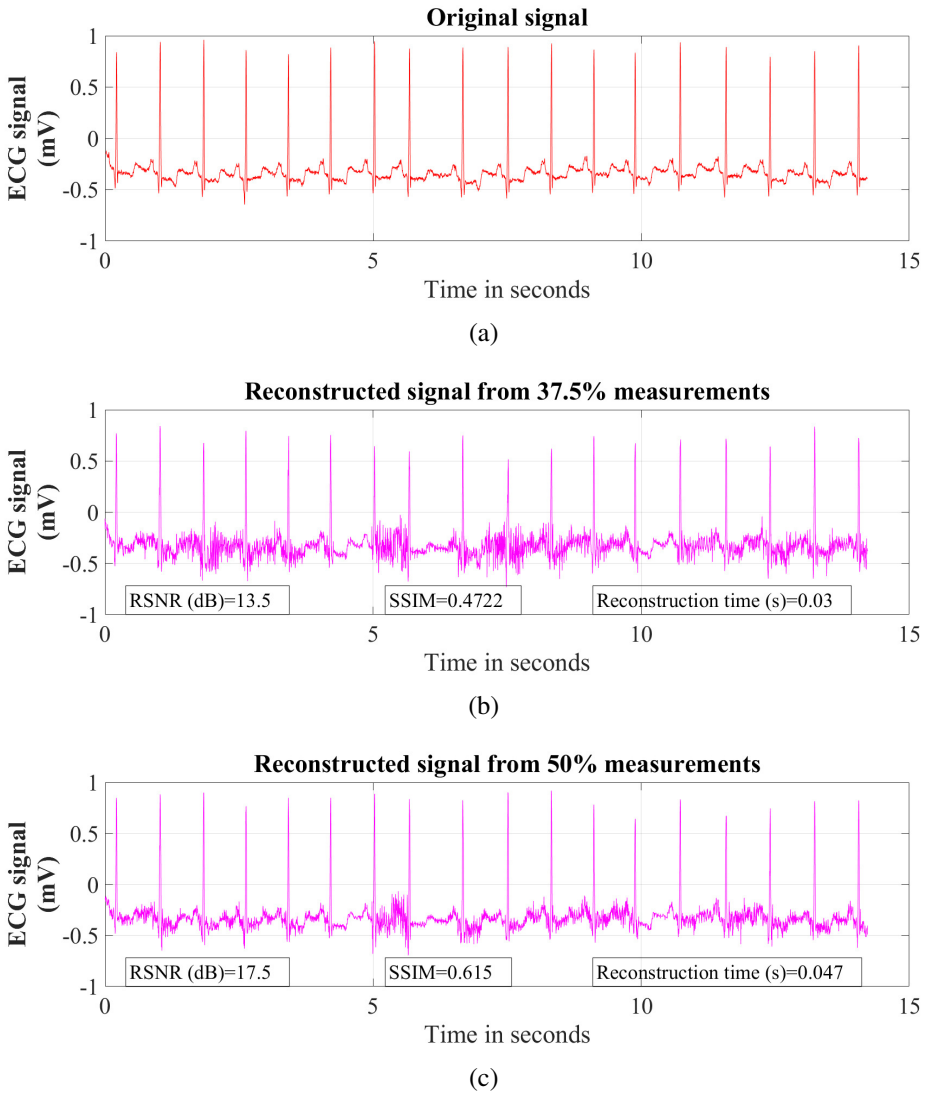


Figure 4.4: MIT-BIH database ECG signal reconstruction

The SIRP accepts the sensing matrix and the compressed measurements as input and performs the reconstruction. Figs. 4.4 (b) and (c) portray the recovery of the ECG signal from 37.5% and 50% samples in terms of the reconstructed signal-to-noise ratio (RSNR), structural similarity index (SSIM) and reconstruction time respectively. As observed, the

SIRP recovered signal shows an RSNR of 13.5 dB and SSIM of 0.4722 from 37.5% measurements within 30 ms. As the measurements are increased to 50%, SIRP attains an RSNR of 17.5 dB and SSIM of 0.615 within 47 ms.

The proposed algorithm is profiled using MATLAB profiler to track the execution time of the critical steps involved and the results are depicted in Fig. 4.5. It can be seen that the processor spends considerable amount of time in the incremental QRD based LS update step (about 54%). The index searching and back-substitution steps each occupy roughly 20% of the execution time. The total reconstruction time measured by the profiler is 0.013s, which corresponds to a throughput of 77 vectors/second. From these results, it becomes evident that accelerating the proposed algorithm on FPGA platforms is quintessential to realizing swifter reconstruction speeds. Furthermore, it can be understood that the incremental QRD algorithm also needs to be accelerated through intelligent use of parallelism to overcome the speed limitations.

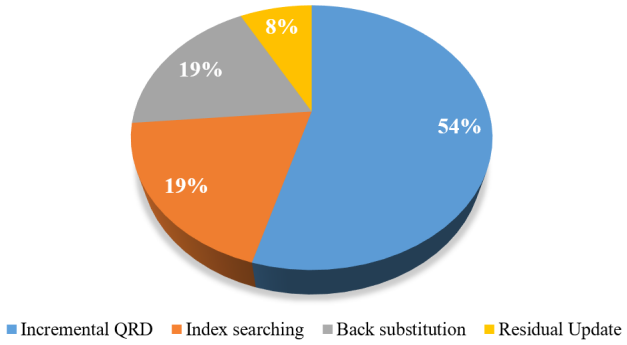


Figure 4.5: Profiling results of the proposed algorithm

The complexity analysis and performance evaluation subsections raise the prospects of designing efficient hardware for the proposed algorithm, capable of delivering reliable reconstructions. Unlike SP, the proposed algorithm does not permit deletion of selected indices from the estimated support set at later stages. This feature imparts a simpler structure from the hardware perspective and the experimental analyses reinforce the reconstruction capability of the proposed algorithm. These aspects motivate the requirement of designing efficient hardware to leverage its merits.

4.3 Architecture Design

This section details the architecture design of the improved SIRP algorithm targeting a structure that supports the reconstruction of signals with size N=1024 and a maximum support set size of 64, from M=256 compressed measurements. Fig. 4.6 presents the high level block diagram of the proposed hardware which is composed of three major blocks: 1) Index searching block; 2) Support set augmentation block and 3) Refining block. The measurement matrix $\mathbf{\Phi}$ is stored in M static random access memories (SRAMs) each of size 2kb such that a single address will output M elements of the corresponding columns of $\mathbf{\Phi}$ at a combined rate of 512 bytes per clock cycle. The dedicated control unit ensures a seamless interaction between all blocks by generating read/write addresses for the different memories as well as initiation enable signals for the various processing blocks according to a well planned schedule.

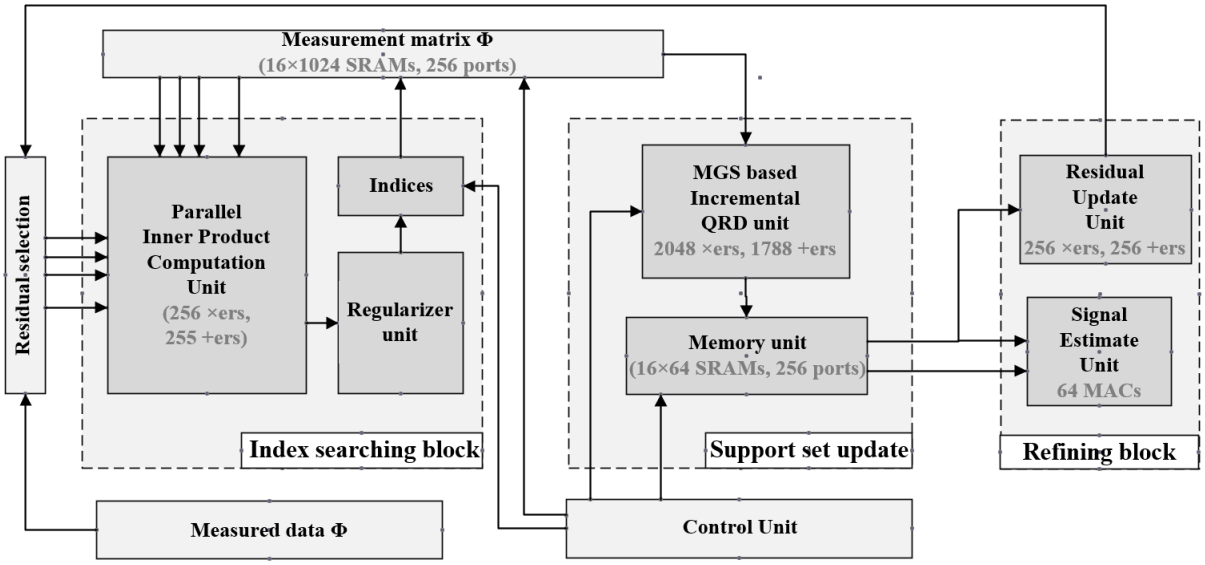


Figure 4.6: Block diagram of the improved SIRP algorithm

4.3.1 Index searching block

The function of the index searching block is to determine the optimal set of indices to be added to the estimated support of the measured data based on $\mathbf{\Psi}^T\mathbf{r}^n$. The naive implementation of the inner product computation would require M parallel multipliers to maximize parallelism and an M-input adder tree to obtain the dot product. Thus, the critical path would consist of 1 multiplier delay and $\log_2 M$ adder delays which can significantly decrease the operating clock frequency. Moreover, the entire process would require N clock cycles to complete the inner product computation of all N columns of the measurement matrix.

The FPGA fabric has dedicated carry logic that can significantly improve the performance of carry propagate adders. Carry save adder structures using 4:2 compressors can potentially deliver greater speed-up for higher word lengths [83], but the gains are not noticeable for word lengths up to 16 bits. Due to this, the work focuses on leveraging the fast carry chains of FPGAs to implement carry propagate adders for the adder tree unit.

In order to reduce the critical path delay, a six-stage pipelined inner product computation unit (IPCU) is proposed which consists of M pipelined multipliers and a pipelined adder tree unit (ATU) to successively compute the N inner products of $\mathbf{\Psi}^T\mathbf{r}^n$ in $N + 9$ clock cycles. The architecture of the IPCU block is illustrated in Fig. 4.7 below.

In this way, the index searching block is able to exploit parallelism and pipelining features to improve overall system performance. Further, the IPCU block is reused in the incremental QRD architecture that will be presented later to improve resource sharing.

The regularizer unit (RU) works in tandem with the IPCU to determine the subset of indices that will be used to incrementally update the matrix decompositions. It does this primarily by employing a grouper module that keeps track of the maximum correlation value in every 32 clock cycles and its index, effectively storing 32 cluster maximums and their indices into the respective RAMs shown in Fig. 4.8.

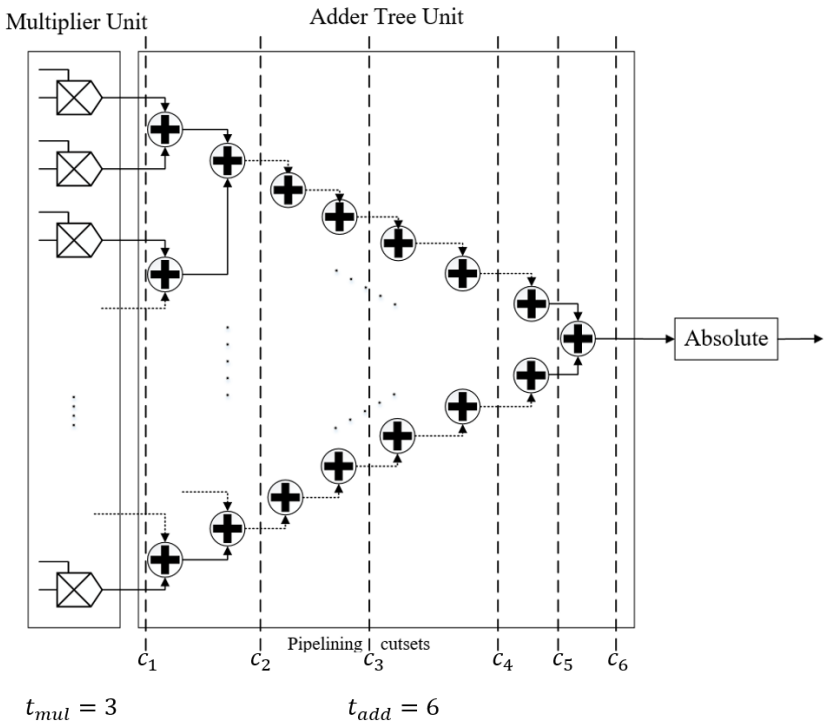


Figure 4.7: Architecture of the 256-input IPCU block

The grouper module is also tasked with determining the overall maximum correlation and passing this value to the range comparator at the end of $N + 9$ cycles. The cluster maximums are now read sequentially into the range comparator which determines the accumulator in which its energy will be accumulated. The architecture of the range comparator is shown in Fig. 4.9 based on the optimized regularization step of the improved SIRP algorithm. In order to perform the aggregation into corresponding subsets, the comparator array compares shifted versions of the global maximum to identify the range in which the cluster maximum would lie by employing a priority encoder. This enables the cluster maximum to be accumulated in the appropriate accumulator in the regularizer unit.

Once all 32 cluster maximums are processed in this manner, the index search block initiates the average energy computation by employing parallel LUT based dividers to obtain the optimal cluster indices. Since the maximum number of elements in a subset which would be used to divide the accumulated energy is restricted by the cluster size 32, the fixed point values of the corresponding fractions are stored in look up tables, so that the average

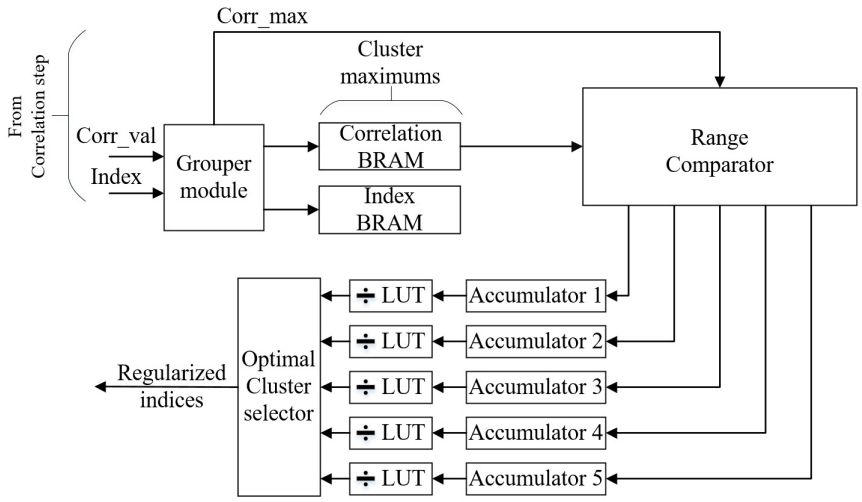


Figure 4.8: Architecture of the regularizer unit

energy can be computed by a simple multiplier. The architecture of the LUT divider is shown below in Fig. 4.10

Once the average energy of each subset is computed in parallel, the control unit initiates a simple maximum computation loop that determines the subset with the maximum average energy in 5 clock cycles and the RU stores the indices of the optimal set to be used for later processing.

Case study of the index search block

In order to demonstrate the computation of the equivalent support set of the sparse signal, a case study is presented for the reconstruction of a 16-sparse 1024-dimensional signal shown in Fig. 4.11 from 256 measurements, whose support indices are {137, 145, 212, 222, 376, 395, 406, 561, 563, 606, 620, 756, 775, 954, 989, 1002}. The random Gaussian sensing matrix of size 256×1024 is clustered into 32 ordered groups in such a way that first 32 columns form the first cluster and so on, as represented in Fig. 4.12.

In the first iteration, the residual vector of size 256×1 is sequentially correlated with the columns in $\mathbf{\Psi}$, resulting in an overall latency of 1030 clock cycles. The grouper module stores the value and index of the column within each cluster exhibiting the highest correlation energy to form the initial candidate set of size 32, which in this case study is

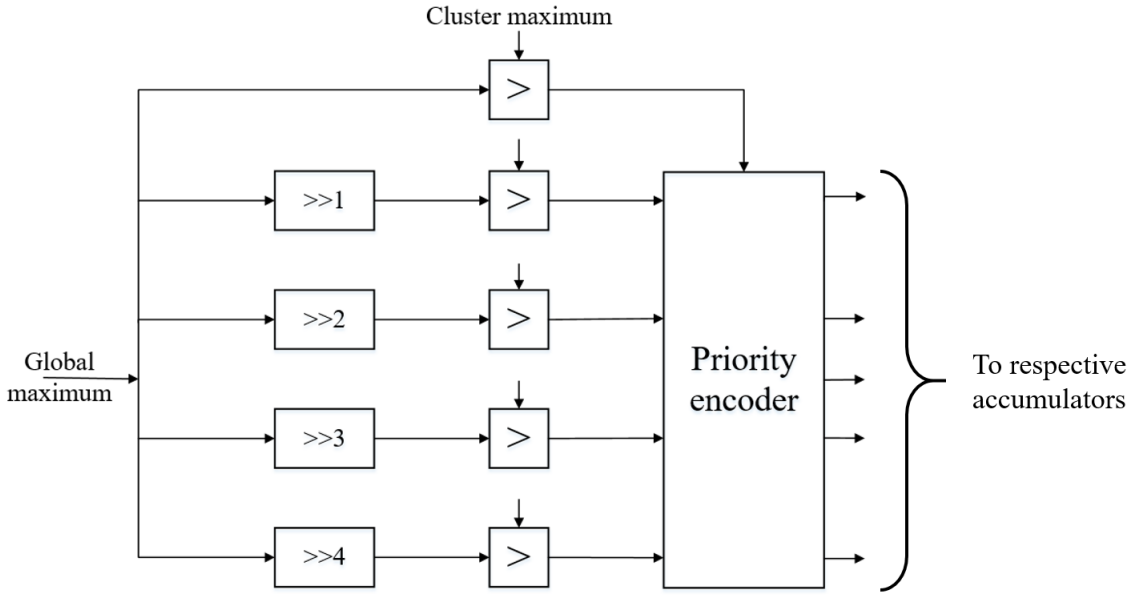


Figure 4.9: Architecture of the range comparator unit

determined to be $\boldsymbol{J}$= {30, 52, 95, 97, 137, 176, 222, 251, 276, 317, 322, 376, 395, 419, 459, 497, 532, 561, 598, 622, 650, 674, 734, 756, 775, 823, 852, 873, 914, 952, 985, 1002}. The correlation values stored in the associated BRAM are read out sequentially and it is determined that the column at index 561 possesses the highest correlation $u_{\max}$=1.7838 over all clusters. This is used to construct the subsets according to the following intervals $[u_{\max}, \frac{u_{\max}}{2}]$,$(\frac{u_{\max}}{2}, \frac{u_{\max}}{4}]$,$(\frac{u_{\max}}{4}, \frac{u_{\max}}{8}]$,$(\frac{u_{\max}}{8}, \frac{u_{\max}}{16}]$ and $(\frac{u_{\max}}{16}, 0]$. The subset with highest average correlation energy is found to be $\mathbf{\Pi}_1$={561, 137, 395, 1002, 775, 222} corresponding to the interval $[u_{\max}, \frac{u_{\max}}{2}]$ and forms the final candidate set of the first iteration. It is observed that all the chosen indices belong to true support set of the sparse signal. These are written to a support memory for later processing and concludes the operation of the RU in 42 clock cycles. The subsequent QR and residual update steps are carried out based on the updated set. If the residual norm remains above the threshold within each cluster, the algorithm proceeds to add more indices to the support set. The subsequent iterations proceed in a similar manner to incrementally augment the equivalent set of the sparse signal, with $\mathbf{\Pi}_2$={563, 145, 756}, $\mathbf{\Pi}_3$={954, 376, 212, 989, 606}, $\mathbf{\Pi}_4$={406} and $\mathbf{\Pi}_5$={620}. The improved SIRP algorithm rightly detects all the support locations of the original signal and

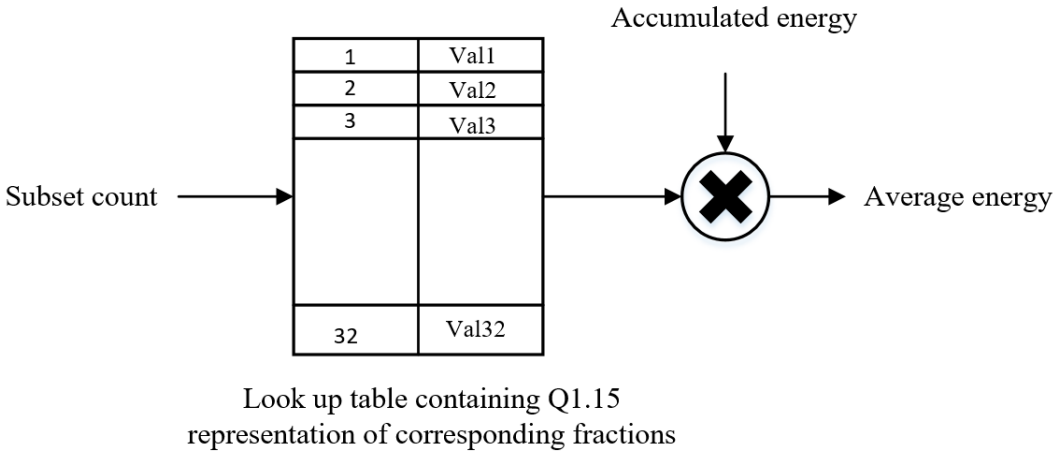


Figure 4.10: Architecture of the LUT divider

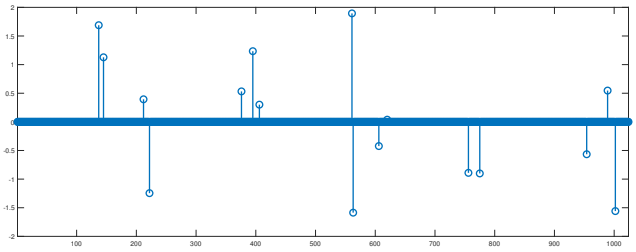


Figure 4.11: Original 1024-length signal

the algorithm terminates as the residual norm falls below the threshold.

4.3.2 Support set update block

This block is tasked with computing the QR decomposition of the selected columns of $\boldsymbol{\Psi}$ represented by $\boldsymbol{\Psi}_{\boldsymbol{\Pi}}$ whose number can vary in each iteration. The MGS strategy permits incremental updates to the Q and R matrices as newer columns are added to the signal support and can be partitioned into two principal steps: triangularization process (TP) and a diagonalization process (DP) as shown in equations (4.1) and (4.2) respectively

$$\boldsymbol{\lambda}^n = \boldsymbol{\lambda}^n - (\boldsymbol{Q}_j^{n-1})^T \boldsymbol{\lambda}^n \boldsymbol{Q}_j^{n-1}, \forall j \in [1, n-1] \tag{4.1}$$

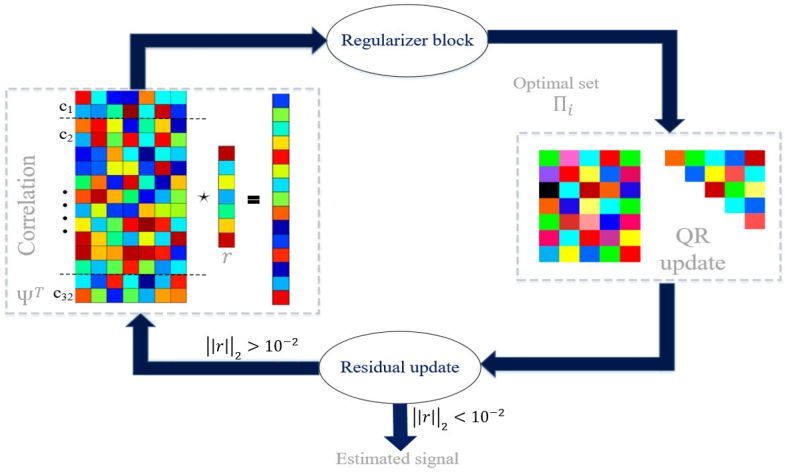


Figure 4.12: Flow of the proposed algorithm in estimating $\hat{\mathbf{z}}$

$$r_{nn} = ||\boldsymbol{\lambda}^n||_2 \tag{4.2}$$

The simplified hardware architecture of the DP and TP steps are shown in Figs. 4.13 and 4.14 respectively.

These complex processes can be decomposed into a few elementary operations like vector inner products, multiplication, subtraction, division and square root. As the designed structure is required to handle a column size of 256, this would entail 256 parallel dividers in the DP block which would drastically increase the hardware resources and the latency. In order to mitigate the drawbacks of the diagonalization step, a fast inverse square root (FISR) block based on Quake's algorithm [84] is employed in our architecture.

Though the FISR was originally intended for floating-point numbers, there have been fixed-point implementations as well [55]. Quake's algorithm employs a right shift on the floating point number followed by a subtraction from 0x5f3759df to yield a good estimate for subsequent Newton's iterations. The fixed point variation consists of performing the fixed point equivalents of these operations and is laid out in [55]. Based on this fixed point Quake's algorithm, the FISR hardware unit is designed to support two Newton's iterations and is fully pipelined to yield the inverse square root value in 6 clock cycles.

As the shared IPCU yields $||\boldsymbol{\lambda}^n||_2^2$, the FISR block can easily compute $\frac{1}{||\boldsymbol{\lambda}^n||_2}$ as shown in Fig. 4.13, which can then be simply multiplied to λ_n yielding the column $\boldsymbol{q}_n$ that can be stored. Therefore, it can be seen that this results in significant hardware and latency savings compared to the naive implementation of the traditional diagonal process.

Since, this strategy ends up computing the reciprocal of $\mathbf{R}$'s diagonal coefficients $(1/r_{ii})$, this yields an added benefit in the final signal estimation step that will be described later. The DP and TP cannot be executed in parallel due to the interdependencies involved and a simple structure implementing a sequence of DP and TP steps can exacerbate the associated latency of the incremental QRD algorithm as the number of TP steps rise linearly with the support set size.

4.3.2 Diagonal Process block

The DP block shown in Fig. 4.13 is responsible for carrying out the normalization of the selected columns and outputs the normalized column $\boldsymbol{q}_j$as well as the corresponding $1/r_{ii}$. Initially, the input column $\boldsymbol{\lambda}_i$ is multiplied with itself through M parallel multipliers having a 3-stage pipeline behaviour and an M-input adder tree unit with a 6-stage pipeline to yield the squared norm of $\boldsymbol{\lambda}_i$ represented by $\|\boldsymbol{\lambda}_i\|^2$. After computing this quantity, it is required

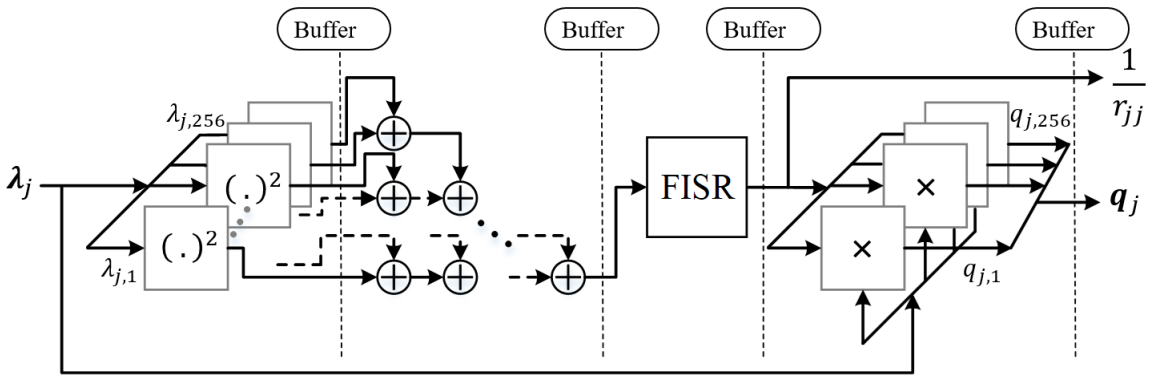


Figure 4.13: Internal hardware architecture of the DP block

to compute its square root and perform division with respect to $\boldsymbol{\lambda}_i$. As described earlier, this is circumvented by deploying the FISR to compute $1/\|\boldsymbol{\lambda}_i\|$, which can then be simply

multiplied to $\boldsymbol{\lambda}_i$ to properly normalize the column. The FISR computes the inverse square root by making the initial guess based on Quake's algorithm and employing two Newton iterations to improve the guess. The total latency of the DP step is determined to be 18 clock cycles.

4.3.2 Triangular Process block

The TP block shown in Fig. 4.14 is tasked with computing the triangular coefficients of $\mathbf{R}$ and orthogonalizing incoming column $\boldsymbol{\lambda}_j$ with respect to current columns in $\mathbf{Q}$. The M parallel multipliers and M-input ATU with pipeline stages of 3 and 6 respectively ensure computation of $\boldsymbol{q}_j^T \boldsymbol{\lambda}_i$ which corresponds to r_{ji} in $\mathbf{R}$. To orthogonalize $\boldsymbol{\lambda}_i$ with $\boldsymbol{q}_j$, it is required to subtract r_{ji} times $\boldsymbol{q}_j$ from $\boldsymbol{\lambda}_i$. This is accomplished by the M parallel multipliers and subtracters shown in Fig. 4.14. The total latency of the TP step is 13 clock cycles.

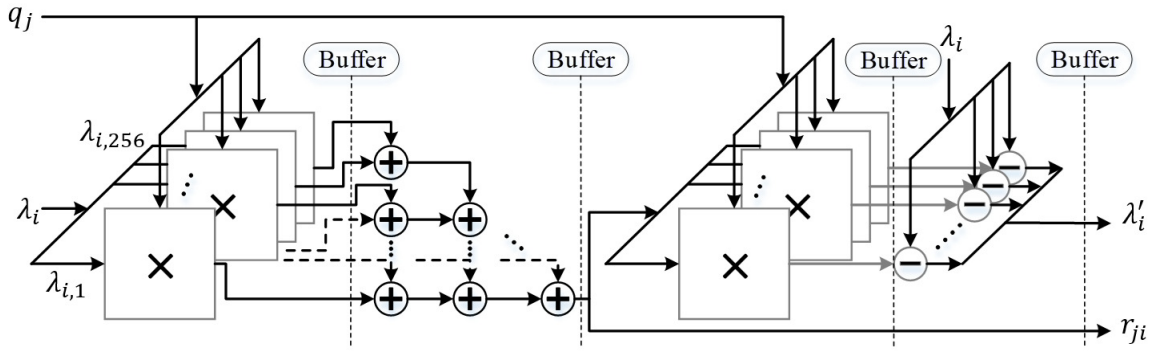


Figure 4.14: Internal hardware architecture of the TP block

4.3.2 Iterative incremental QRD architecture

Since the SIRP algorithm permits multiple index selection in each iteration, the support set update block should be capable of handling the DP and TP tasks leveraging parallelism wherever possible without drastically increasing hardware resources. It can be seen that when newly selecting columns need to be decomposed, they must be orthogonalized to columns stored in Q. When multiple columns need to be orthogonalized in an identical

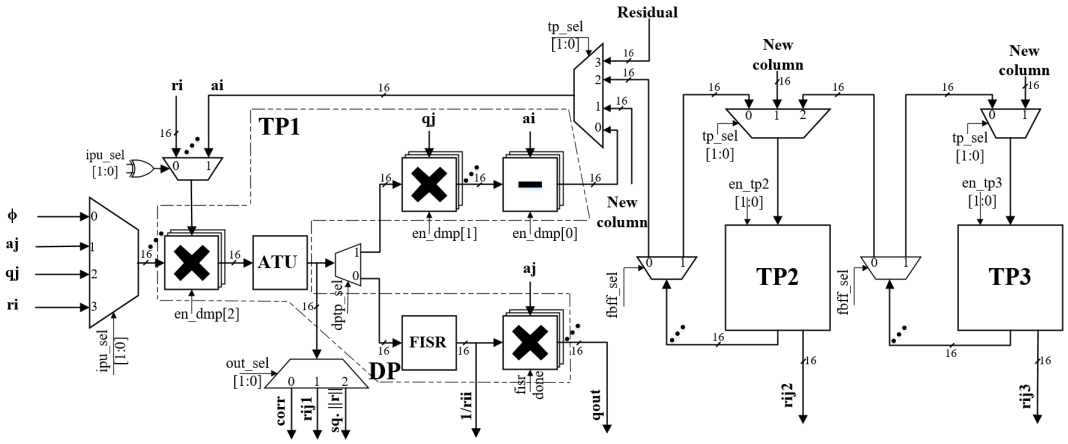


Figure 4.15: Iterative architecture of the Incremental QRD

manner, significant time savings can be induced by concurrent updates for each memory access.

To this intent, a novel iterative incremental QRD architecture using an elaborate feedback circuit is shown in Fig. 4.15 that is amenable to the incremental nature of the support set update of the SIRP algorithm and exploits the inherent parallelism in the TP step, consuming higher hardware resources. The proposed architecture comprises one DP block and three TP blocks labelled TP1, TP2 and TP3 among which the DP and TP1 blocks can be combined to save resources as depicted in Fig. 4.15. The architecture has four modes of operation which can be selected by the ipu_sel signal as shown in Table 4.2.

This mechanism is used to share the hardware resources among the independent processing steps of the proposed algorithm to induce resource savings. When ipu_sel is 0, the the first set of M parallel multipliers and the ATU in the combined DP-TP1 block are used to perform correlation of the columns of $\mathbf{\Phi}$ with the residual $\mathbf{r}_i$. All other compute units are switched off by generating corresponding low enable signals. The out_sel signal can be used to send the output of the ATU to the regularizer unit, whose operation has been discussed earlier.

When ipu_sel is 1, it implies the architecture is configured to perform the operations in the DP step to normalize the column $\boldsymbol{\lambda}_i$. In this case, the unused hardware in the TP blocks

Table 4.2: Operation modes of the proposed architecture

ipu_sel	Operation modes
0	Correlation of Φ with $\mathbf{r}$
1	Diagonal process
2	Triangular process
3	Residual update

are turned off by appropriate control signals. The new column $\boldsymbol{a}_j$ is multiplied with itself through proper ipu_sel and tp_sel signals. The ATU produces $\|\boldsymbol{a}_j\|_2^2$ which is then passed to the FISR block by setting dptp_sel high. After computing the inverse square root, the FISR block passes the ready signal and the output to the parallel multipliers to yield the orthonormal column $\boldsymbol{q}_j$ which is then stored in memory.

When ipu_sel is 2, the architecture is configured to perform the TP operations for 3 columns in parallel using the TP1, TP2 and TP3 blocks, based on the number of incoming columns. If it is less than 3, then the corresponding TPs are disabled by the control unit. The incoming columns have to be orthogonalized to the columns read from Q storage. For instance, if there are 4 stored columns in Q and there are three new columns selected in the current iteration, $\boldsymbol{a}_5$ to $\boldsymbol{a}_7$ are primarily orthogonalized to $\boldsymbol{q}_1$ concurrently by passing $\boldsymbol{q}_1$ to the corresponding TP blocks. The modified columns are then fed back to the respective TPs parallelly through demultiplexers to orthogonalize them with respect to $\boldsymbol{q}_2$ and similarly for $\boldsymbol{q}_3$ and $\boldsymbol{q}_4$. Once this is completed, the control unit initiates the diagonal process for the modified column $\boldsymbol{a}_5^{''''}$ to finally obtain $\boldsymbol{q}_5$. As this is stored to the Q memory, it is also passed to TP1 and TP2 in order to orthogonalize $\boldsymbol{a}_6^{''''}$ and $\boldsymbol{a}_7^{''''}$ with respect to $\boldsymbol{q}_5$. The demultiplexers ensure that these are fed forward to the corresponding TPs through proper control signals. The process goes on till the final column $\boldsymbol{q}_7$ is computed and stored.

When ipu_sel is 3, the architecture is configured to estimate the new residual based on newly augmented columns in $\mathbf{Q}$. The update equation is shown below

$$\boldsymbol{r}_i = \boldsymbol{r}_{i-1} - \mathbf{Q}_n \mathbf{Q}_n^T \boldsymbol{r}_{i-1} \tag{4.3}$$

The new residual vector corresponding to the current column is then used to update the residual after considering the subsequent columns iteratively.

Each TP block operates in three modes based on the data flow. In the first mode, a freshly selected column from the previous index searching block is accepted into the TP. The second mode is a feedback mode where the output of the TP is fed back to itself to enforce orthogonality to the next column in Q. This permits computation of three respective triangular coefficients and modified columns $\lambda_i^{'}$'s in parallel. In the third mode, the output of the TP is fed forward to the adjacent TP or DP to compute the subsequent column of Q. These modes are controlled by the multiplexer and demultiplexer select lines as can be seen from Fig. 4.15. It should be noted that the estimated support set size can vary in every iteration and the control unit is required to properly sequence these operations by producing the corresponding enable signals.

Since vector inner product is ubiquitous in all stages of the algorithm, a hardware sharing scheme is proposed to bring down resource consumption and increase hardware utilization efficiency using a set of multiplexers regulated by the control unit. This scheme facilitates usage of the same IPCU and ATU blocks for the index searching, diagonalization, triangularization, residual update and residual norm computation steps of the algorithm.

4.3.3 Refining block

The refining block is tasked with updating the residual in each iteration and computing the signal estimate after the residual norm falls below the threshold or sparsity maximum is reached.

4.3.3 Residual Update unit

As shown in equation step 9 of Algorithm 1, the newly updated columns of Q are sufficient to update the residual for the following iteration without requiring an intermediate estimation of the signal. Since the hardware structure needed to implement the residual update

in this manner is similar to the structure of the TP block, it is proposed to control the flow of data to the TP1 block in the incremental QRD architecture to realize this task. But this requires to iteratively approach the final residual of each iteration by breaking down the problem into a series of updates for each newly added column of Q represented by $\boldsymbol{q}_s$.

The residual update in equation (4.3) can broken down into three simple operations shown below that can be mapped on to TP1 block so that resources can be shared.

$$\boldsymbol{v} = \mathbf{Q}_n^T \boldsymbol{r}_{i-1} \tag{4.4}$$

$$\boldsymbol{v}^{'} = \mathbf{Q}_n \boldsymbol{v} \tag{4.5}$$

$$\boldsymbol{r}_i = \boldsymbol{r}_{i-1} - \boldsymbol{v}^{'} \tag{4.6}$$

The operation in equation (4.4) can be mapped to the first set of parallel multipliers in the dual DP/TP1 block as one input is the column $\boldsymbol{q}_j$ and the other input is the residual $\boldsymbol{r}_{i-1}$. The scalar product of these quantities is then multiplied to $\boldsymbol{q}_j$ through the second set of multipliers in the TP1 block and the resulting vector is subtracted from the residual through parallel subtracters also available in TP1.

The residual update is followed by its norm computation to determine if the architecture should continue to search for possible true support indices or if the current estimated index set closely matches the true support set of the signal. It would suffice to compute the squared norm of the residual using the IPCU block in TP1 and a comparison operation with the modified threshold is carried out to inform the control unit to generate the signals for carrying out the next iteration or proceed for signal estimation.

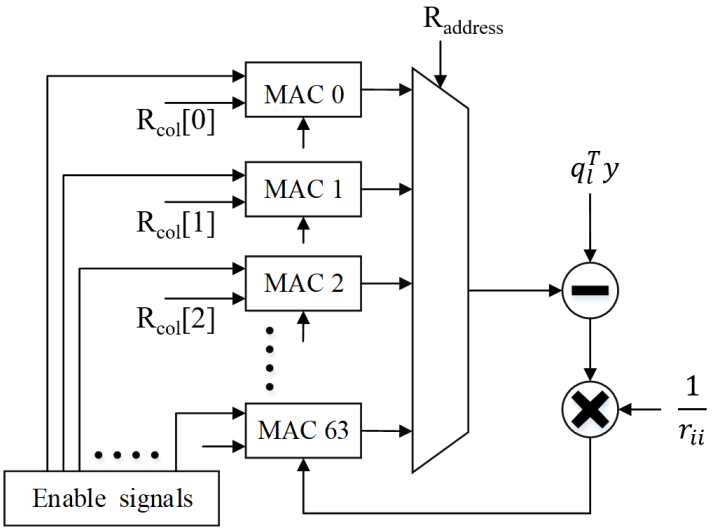


Figure 4.16: Signal estimation block

4.3.3 Signal Estimation unit

The signal estimation unit is tasked with computing the solution of the linear system $\boldsymbol{R}\hat{\mathbf{x}} = \boldsymbol{Q}^T\mathbf{y}$ by using a back substitution approach. The architecture implementing this functionality consists of 64 multiply and accumulate (MAC) modules that permit parallel substitution of estimated signal coefficients into the upper layers based on the corresponding enable signals, shown in Fig. 4.16. The $\boldsymbol{q}_l^T y$ quantity is sequentially computed using the shared IPCU of the TP1 block from the lower to upper layers. A 64×1 multiplexer chooses which MAC output is to be subtracted from the corresponding $\boldsymbol{q}_l^T\mathbf{y}$ quantity which is then multiplied by the $1/r_{ii}$ coefficient stored in the R matrix. The conventional division approach would have consumed longer accumulated latencies which has been circumvented by computing the reciprocal of the diagonal elements of R and incurring only multiplier latency costs.

4.3.4 Control Unit

The complex hardware blocks discussed in the previous subsections require a sophisticated control structure to execute the SIRP algorithm seamlessly. The control unit is responsible

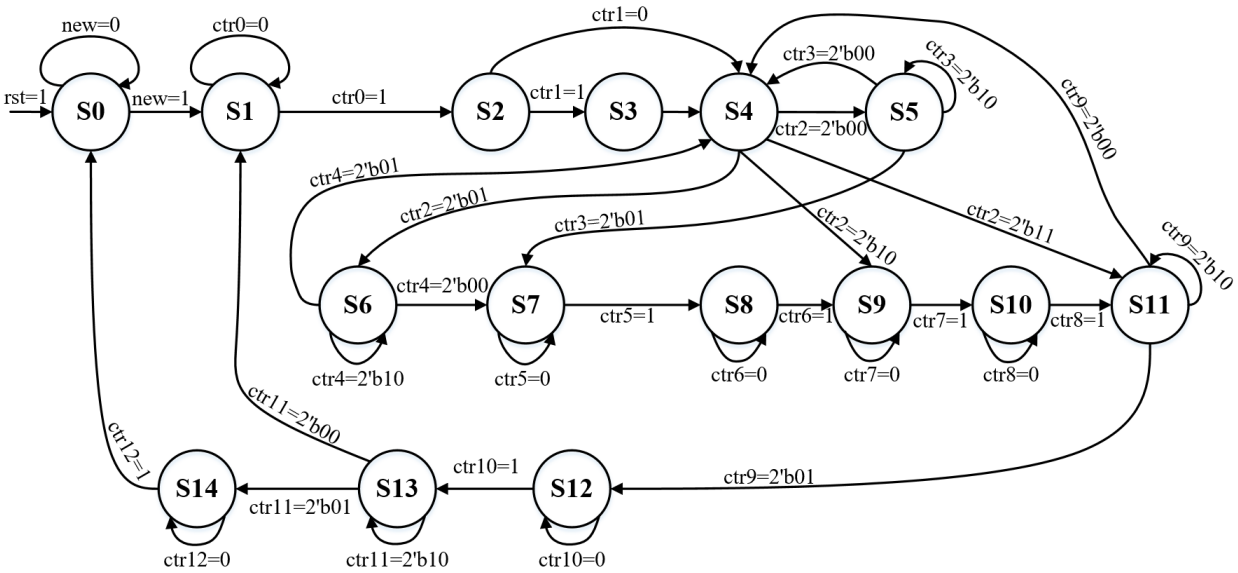


Figure 4.17: Finite state machine diagram

for carrying out the complex hardware sharing mechanism among the different processing steps of the algorithm. This is realized through the design of a complex finite state machine (FSM) shown in Figure 4.17, which will be discussed in detail. A Moore FSM design is pursued where the output signals are only dependent on the current state. The control signals for the data paths of the overall architecture are manipulated by each state for synchronizing the various operations of the SIRP algorithm. The ctr$n$ signals that decide state transitions are controlled by the timing behaviour and/or meeting the iteration constraints that will be detailed below. The FSM has been highly optimized for the number of states.

State S0 is the initial wait state upon reset, which waits for the 'new' signal indicating a fresh set of measurements have been loaded into the measured data memory. State S1 deals with the correlation and regularization steps of the algorithm and the number of clock cycles taken to complete this state are fixed in every iteration. The FSM issues the necessary control signals to sequentially input columns of $\mathbf{\Psi}$ into the 256-parallel IPCU unit along with the residual. The correlation of $\mathbf{\Psi}$ with the residual $\mathbf{r}^{n-1}$ takes $N + 6$ cycles. The regularizer unit works in parallel during this time slot to find the cluster maximums followed by an additional 42 cycles to compute the optimal support indices. Since columns of $\mathbf{\Phi}$ are

normalized to 1, the first selected column can be written to $\boldsymbol{Q}$ bypassing the orthonormalization step. This is accomplished by a special first iteration (FI) mode characterized by state S3. The adjudicator module within the control unit is tasked with efficiently carrying out the transition to subsequent states. Fig. 4.18 corresponds to the timing diagram when six new columns are being augmented to the support set and the corresponding residual update.

S2 is a single latency state that transitions to an exclusive first iteration (FI) state S3 or directly to S4 in other iterations. Since all columns of $\boldsymbol{\Psi}$ in the traditional CS recovery problem possess unit norm, the column corresponding to the first index in the estimated support set can be written directly to the $\boldsymbol{Q}$ memory without undergoing additional wasteful operations. This task is accomplished in state S3 which is completed in 5 clock cycles.

State S4 is one of the key states in the FSM as it is required to transition to one of four states based on the number of selected columns remaining to be orthogonalized. State S5 is triggered when new columns have to be input from the measurement matrix RAMs into the incremental QRD unit. State S5 is tasked with reading three new columns $\mathbf{a}_2 - \mathbf{a}_4$ sequentially and orthogonalizing them with respect to the first column of $\boldsymbol{Q}$ (from state S3), while simultaneously computing and storing r_{12}, r_{13} and r_{14}. At the end of S5, the orthogonalized version of $\mathbf{a}_2$ viz. $\mathbf{a}_2^{'}$ is transferred from the output of TP1 to the shared DP unit which computes $\frac{1}{r_{22}}$ and subsequently $\mathbf{q}_2$. These operations are carried out in state S7, consuming 18 clock cycles. As S8 starts execution, the orthogonalized $\mathbf{a}_3^{'}$ and $\mathbf{a}_4^{'}$ are transferred from TP2 and TP3 to TP1 and TP3 respectively, and $\mathbf{q}_2$ is also passed as input to both modules in order to compute $r_{23}, r_{24}, \mathbf{a}_3^{''}$ and $\mathbf{a}_4^{''}$. The output of TP1, viz. $\mathbf{a}_3^{''}$, is fed to the DP in state S9 in order to compute the corresponding inverse diagonal coefficient of $\boldsymbol{R}$ and column of $\boldsymbol{Q}$. The process to compute r_{34} and $\mathbf{a}_4^{'''}$ follows in S10, with $\mathbf{a}_4^{'''}$ being fed back to DP in S11 to compute the related coefficients, which concludes the first inner loop of the 1^{st} iteration.

The new column orthogonalization (NCO) mode is used to fetch new columns from the

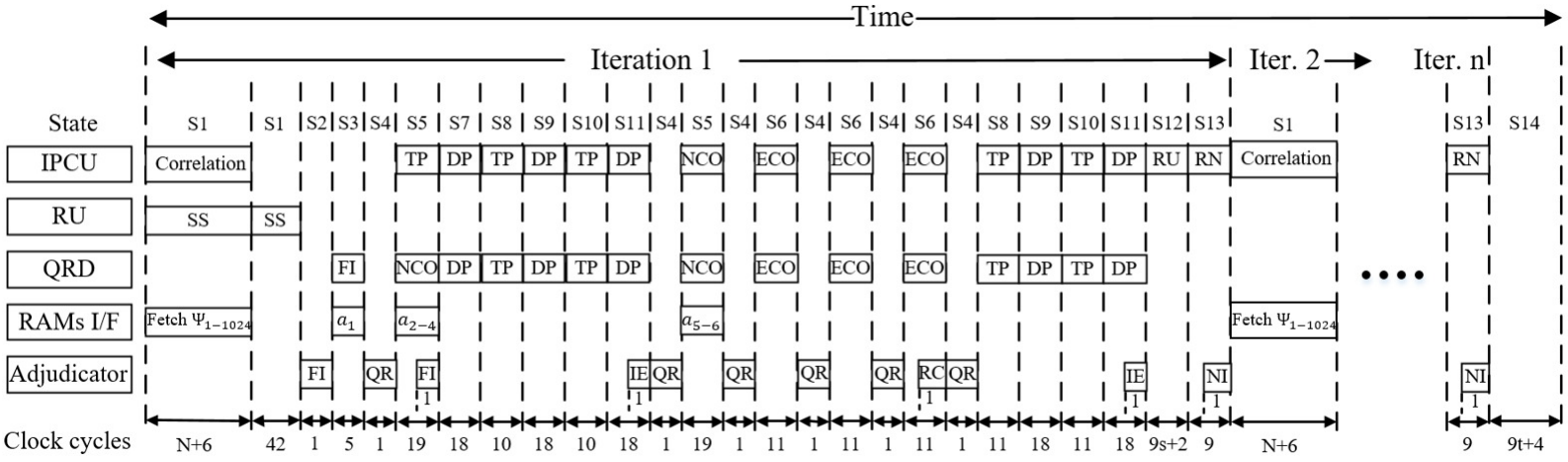


Figure 4.18: Timing overhead of the sub-blocks in the proposed architecture. 's' represents the number of indices selected in each iteration and 't' represents the total number of selected indices

$\boldsymbol{\Psi}$ RAMs and orthogonalize them with respect to $\mathbf{q}_1$ while also computing the respective triangular coefficients of $\boldsymbol{R}$. The orthogonalized version $\mathbf{a}_2^{'}$ is given as input to the DP sub-module in order to obtain $\mathbf{q}_2$ during state S7. The subsequent states will feed forward the orthogonalized vectors to the adjacent TP modules for successive orthonormalizations. At state S11, an iteration end (IE) adjudicator looks at the number of remaining support columns that need to be augmented and transitions the system back to state S4, and the process repeats till all new columns have been augmented to $\boldsymbol{Q}$ and $\boldsymbol{R}$. The existing column orthogonalization (ECO) mode is undertaken to use TP1, TP2 and TP3 sub-modules to orthogonalize the new set of columns to the successive columns in $\boldsymbol{Q}$ in a feedback mode. Towards the end of S11, the IE adjudicator finds that columns stored in $\boldsymbol{Q}$ match the number of newly selected support indices. This initiates the residual update (RU) step which utilizes the TP1 sub-module recursively to update the residual respective to the columns newly stored in $\boldsymbol{Q}$. This is followed by the residual norm computation step that determines if a new iteration is to be pursued or not. Once the stopping criterion is met, signal estimation is carried out in state S14 in $9t + 4$ clock cycles.

At this stage, the FSM performs a check to ascertain if number of columns stored in $\boldsymbol{Q}$ match the estimated support set size of the present iteration. If it finds more indices are present, control is transferred back to S4 to resume the process of orthogonalization. State S5 is then executed in order to fetch the new columns and simultaneously orthogonalize them with respect to $\mathbf{q}_1$. These have to be subsequently orthogonalized with respect to $\mathbf{q}_2$ to $\mathbf{q}_4$, which is accomplished by the S4-S6 loop as seen in Figure 4.17. After this process is completed, the successive DP and TP states are undertaken to obtain the new columns of $\boldsymbol{Q}$ and corresponding coefficients of $\boldsymbol{R}$.

Once the number of columns in $\boldsymbol{Q}$ and the estimated support set size match, the FSM transitions to state S12 which deals with the residual update corresponding to the newly added columns of $\boldsymbol{Q}$ according to equation (7), iteratively. If s is the number of newly added columns in the current iteration, S12 consumes $9s+2$ clock cycles to obtain the residual for

the subsequent iteration of SIRP. This is followed by a residual norm computation stage that shares the IPCU hardware in S13 to compute the norm of the new residual and determine if more iterations are needed. If the residual norm falls below the threshold, support set estimation can be stopped and the FSP proceeds to state S14 to estimate the signal using the stored $\boldsymbol{Q}$ and $\boldsymbol{R}$ matrices. The estimation process occurs in $9t + 4$ clock cycles, where t is the total size of the estimated support set.

It should be noted that the designed FSM is capable of handling variable size of selected indices in each iteration, using shared hardware to effectively bring down the number of iterations needed to estimate the signal.

4.4 Experimental Results

In this section, the proposed implementation's efficiency of signal reconstruction is analyzed for different data precision parameters. The resource utilization profile and the recovery performance of the proposed hardware is compared with other state-of-art works.

4.4.1 Fixed point analysis

Fixed-point designs are more efficient than their floating-point counterparts with regard to area, speed and power considerations. In order to estimate an appropriate data precision format for the proposed design, a series of experiments are performed for different choices of data widths and number of fractional bits using MATLAB 2018a fixed point toolbox. The randomly generated Gaussian measurement matrix of size $\mathbf{\Phi} \in \mathbb{R}^{256\times1024}$ is used to compressively sample a randomly generated k-sparse signal to obtain the measurements vector $\mathbf{y}$. In each experiment, $\mathbf{\Phi}$ and $\mathbf{y}$ are quantized according to the respective data and fractional widths and given as input to the fixed point version of the improved SIRP algorithm.

The signal recovery efficiency is measured in terms of the reconstruction signal-to-noise

Table 4.3: Reconstruction efficiency under varying data precision for N=1024, M=256 and k=36

Data width	**16-bit**		**18-bit**		**20-bit**	
Fractional	RSNR (dB)	ASCE	RSNR (dB)	ASCE	RSNR (dB)	ASCE
9	25.88	0.0278	24.79	0.0556	24.17	0.1111
10	30.57	0.0278	35.07	0	32.26	0.0556
11	41.58	0	42.11	0	38.28	0.0278
12	45.47	0	46.72	0	48.35	0
13	55.96	0	53	0	54.20	0
14	25.12	0	62	0	57.76	0
15	8.57	0	67.23	0	64.71	0

ratio (RSNR) and the average support cardinality error (ASCE). The RSNR can be defined as

$$\text{RSNR} = 20\log_{10}\frac{\|\mathbf{x}\|_2}{\|\mathbf{x}-\hat{\mathbf{x}}\|_2} \tag{4.7}$$

ASCE refers to ratio of number of undetected support set elements to the total number of true support elements and can be mathematically written as

$$\text{ASCE} = 1 - \frac{\mathbb{E}\{\mathbf{\Gamma} \cap \widetilde{\mathbf{\Gamma}}\}}{k} \tag{4.8}$$

where $\mathbf{\Gamma}$ and $\widetilde{\mathbf{\Gamma}}$ are the true and estimated support sets respectively.

As seen in Table 4.3, for a fixed data width, an increase in the fractional width improves the RSNR and ASCE parameters as expected. However, the RSNR is found to drop with further escalation in fractional width, as the corresponding integer width would be inadequate for appropriate representation of data.

4.4.2 Implementation results

In order to compare with existing designs that target sparse signal recovery, the proposed architecture is implemented for a problem size of N=1024 and M=256 on a Xilinx Virtex Ultrascale+ xcvu9p FPGA device. The fixed precision adder, subtracter and multiplier

Table 4.4: Comparison with state-of-art designs

Reference & Year	**Algorithm**	**Problem Size**			**Time**	**Frequency**	**Format**	**Target**	**Resource Utilization**			**Process**	**RSNR / PSNR**	
		M	N	K	(μs)	(MHz)	Qm.n	**FPGA**	BRAM	DSP	Slices	**cycles**	(dB)	
[57] ('12)	OMP	256	1024	36	622	100	25-bits	Virtex 6	258	268	32001	-	23.5	-
[59] ('15)	OMP	256	1024	36	340	119	Q9.9.	Virtex 6	576	589	6208	40788	-	38.9
[67] ('18)	Imp. OMP	256	1024	36	170	135.4	Q12.12	Virtex 6	342	1544	7860	23009	31.04	-
[74] ('18)	OMP	256	1024	36	450/314	94/135	18-bits	Virtex 6/7	-	-	-	-	-	-
[68] ('19)	OMP	256	1024	36	327	133.33	Q6.12	Virtex 6	430	386	10902	43604	18.34	-
[69] ('19)	OMP	256	1024	36	238	210	Q6.12	Kintex 7	386	523	28443	50030	-	25.57
[70] ('20)	Imp. OMP	256	1024	36	423	113	floating	Zynq U	521	1700	-	-	-	-
[73] ('20)	SP	58	256	8	22.5	15.4	Q15.13	Virtex 7	0	3324	46268	-	28.2	-
Proposed	SIRP	256	1024	36	77	190	Q3.13	Virtex U+	290	1878	10396	14525	20.1	41.1
Proposed	SIRP	256	1024	36	122	119	Q3.13	Virtex 7	290	1878	16817	14525	20.1	41.1

units occupying either look up tables (LUTs) or DSP48E2 slices are generated by Vivado's IP generator , whereas XILINX BRAMs (RAM18E2) are configured as dual port RAMs or ROMs. The distributed memory within the configurable logic blocks (CLBs) across the device are used to store intermediate results. The architecture is implemented based on System Verilog description, using Vivado synthesis and implementation strategies for the chosen target. In order to validate the functional performance of the design in the presence of logic and routing delays, post-implementation timing simulation was carried using the Synopsys VCS simulator.

Since existing approaches present their results for a signal sparsity level of 36, corresponding random test signals with amplitude varying between -1 and $+1$ are used to test the proposed design. Table 4.4 compares reconstruction time, maximum operating frequency, detailed resource utilization and recovery performance of our implementation with state-of-art works. It can be seen that the proposed design operates at 190 MHz and is able to reconstruct 36-sparse signals within 77μs, which considerably excels the prior works. This can be attributed to the fewer number of iterations required by SIRP to recover the vector, compared to the 36 iterations needed by OMP. The OMP variant proposed in [67] which brings down the iterations by a factor of 2, is found to have a reconstruction time of 170μs. The proposed implementation trades off the hardware resource usage in order to achieve significant speed-up in the running time, compared to the state-of-art OMP implementations. It should be noted that the SP processor [73] targeting a relatively smaller problem size (N=256, M=58, k=8) consumes significantly greater resources than the proposed implementation, as seen in Table 4.4. Moreover, while the SP processor requires approximately 4 iterations on average to reconstruct the 8-sparse signal in 22.5μs, our design can achieve faster recovery of 8-sparse signals by roughly 15%, as shown in Table 4.7. Based on the achieved reconstruction speed for 36-sparse vectors, the proposed hardware is able to attain a processing throughput of roughly 13100 Vectors/second or equivalently 13.4 million samples per second (MSPS). The results of the proposed architecture for the Xilinx

Virtex-7 FPGA device are also tabulated. The proposed implementation is able to achieve a PSNR (RSNR) of 41.1 dB (20.1 dB) which is better than [59], [68] and [69]. It can be seen that [57] and [73] report better RSNR at considerably higher resource utilization. It is seen that the proposed implementation incurs an increase in the resource utilization when using the Virtex-7 FPGA board compared to the Virtex Ultrascale+ board. This can be attributed to the architecture of the configurable logic block (CLB) in both families. Since Virtex Ultrascale+ CLBs house 6-input LUTs compared to 4-input LUTs for the Virtex-7, the former is able to pack logic into fewer slices.

Table 4.5: Virtex Ultrascale+ Resource utilization for varying data widths

Data width	DSP48E2	RAM18E2	Slice LUTs	Slice Reg.
16	1878 (27%)	290 (13%)	56032 (5%)	22015 (1%)
18	1878(27%)	290 (13%)	66602 (5.5%)	24404 (1%)
20	3736(54%)	420 (19%)	72209 (6%)	27530 (1%)

Table 4.5 shows the increase in resource consumption for increasing data widths. The DSP48 slices found on Virtex Ultrascale+ and Virtex-7 devices support atmost 27×18 and 25×18 multiplier widths respectively. Therefore, the number of DSP slices occupied double as the data width is increased beyond 18. In order to maintain moderate resource consumption, the chosen word size in the proposed implementation is 16 bits with 13 bits allocated for fractions. Since application specific adjustments to the fractional precision can be made without impacting resource consumption and timing, this choice is found to be sufficient for the proposed implementation.

4.4.3 Timing analysis

Timing complexity provides a general idea about the overall time needed to reconstruct the signal. In the proposed SIRP architecture, the index searching stage comprising correlation and regularization operations require $N + 48$ clock cycles. The time complexity estimation of the incremental QRD process is complicated due to the variable number of columns

augmented in each iteration. In order to analytically derive the cycle consumption of this process, it is assumed that s=6 columns are added to the signal support in every iteration. The time complexity of the incremental QRD, residual update and signal estimation steps based on this assumption are shown in Table 4.6. Accordingly, reconstruction of 36-sparse signals is assumed to require 7 iterations to accommodate possible false index choices as well. It can be seen that substituting N=1024, k=36 and t=7 estimates overall time complexity as 12879 clock cycles, which can be though of as the lower bound. Similarly, assuming s=3 columns are augmented in every iteration requires $t = 12$ iterations and yields 17152 clock cycles as the upper bound. As the observed average iteration count is around 8, the overall time complexity will lie between these calculated lower and upper bounds.

Table 4.6: Clock Cycle consumption for N signal size, k sparsity level and t number of iterations

Assumption	Index search	Incremental QRD	Residual Update	Estimate $\hat{\mathbf{x}}$
s=6	$t(N+48)$	$72t^2+152t$	$65t$	$9k+4$
s=3	$t(N+48)$	$18t^2+76t$	$38t$	$9k+4$

For the results shown in Table 4.4, the proposed Virtex Ultrascale+ implementation consumes 14525 clock cycles to complete signal estimation. Since time complexity is dependent on the signal sparsity level, tests are performed for different values of k as shown in Table 4.7. It is observed that for lower sparsity levels, reconstruction speed is greatly improved without significant change in RSNR. It should also be noted that unlike OMP and SP based designs, the proposed implementation does not require prior knowledge of k for termination or signal estimation such that the reconstruction process can remain unmonitored for signals of varying k.

Table 4.7: Performance under varying sparsity levels with fixed N=1024 and M=256

Parameters	k**=8**	k**=16**	k**=24**
RSNR (dB)	24.99	25.6	23.33
Recovery time (μs)	19	29.7	45.2

4.4.4 Power Consumption

Power consumption plays an important role in evaluation of the design performance. The proposed design employs control enabling of the DSP arithmetic and memory elements to ensure power savings. The dynamic power consumption of the Virtex Ultrascale+ based design is estimated by Vivado Power Analyzer for various operating frequencies. Since the throughputs of the proposed design is calculated as the reciprocal of the respective reconstruction times, the dynamic power efficiency [73] can be determined as the ratio of dynamic power consumption to the throughput and is measured in μW/Vector. It shows the power cost to the processor to reconstruct one 1024-length signal. The results tabulated in Table 4.8 show that as the operating frequency is increased, dynamic power also rises. It can be observed that there is a marked difference in the dynamic power consumption when the operating frequency is raised from 125 MHz to 138 MHz, whereas for the following increases in frequency the change is not as drastic. Thus, the dynamic power efficiency rises sharply to 561 at 138 MHz and falls for subsequent values as the throughput increases. The dynamic power estimated by the Vivado Power Analyzer increases gradually for higher clock frequencies (after possible power optimizations done automatically by the tool), while the corresponding throughput gains are more pronounced. Since dynamic power efficiency is measured as the ratio of dynamic power to the throughput, we see a fall in the dynamic power efficiency and not the actual dynamic power.

Table 4.8: Throughput and dynamic power efficiency for various operating frequencies

Frequency (MHz)	125	138	166	190
Dynamic Power (mW)	3898	5517	5683	5835
Throughput (Vectors/sec)	8550	9835	11450	13100
Dynamic power efficiency (μW/Vector)	455	561	481	445

4.4.5 ECG signal recovery

CS based wireless health monitoring systems have gained significant traction in recent years [71, 72] pushing for faster and cheaper sparse reconstruction engines. In this experiment, ECG signal reconstruction from the MIT-BIH database is considered. The signal shown in Fig. 4.19 consists of 20 segments of length 512 and each segment is multiplied by $\mathbf{\Psi} = \mathbf{\Phi\Omega}$ yielding 256 measurements, where $\mathbf{\Phi}$ is a random Gaussian matrix and $\mathbf{\Omega}$ is a 512×512 wavelet basis. The reconstructed $\hat{\mathbf{z}}$ vectors corresponding to each segment undergo a post-processing step ($\mathbf{\Omega}\hat{\mathbf{z}}$) in order to obtain the ECG signal. Fig. 4.19 shows that

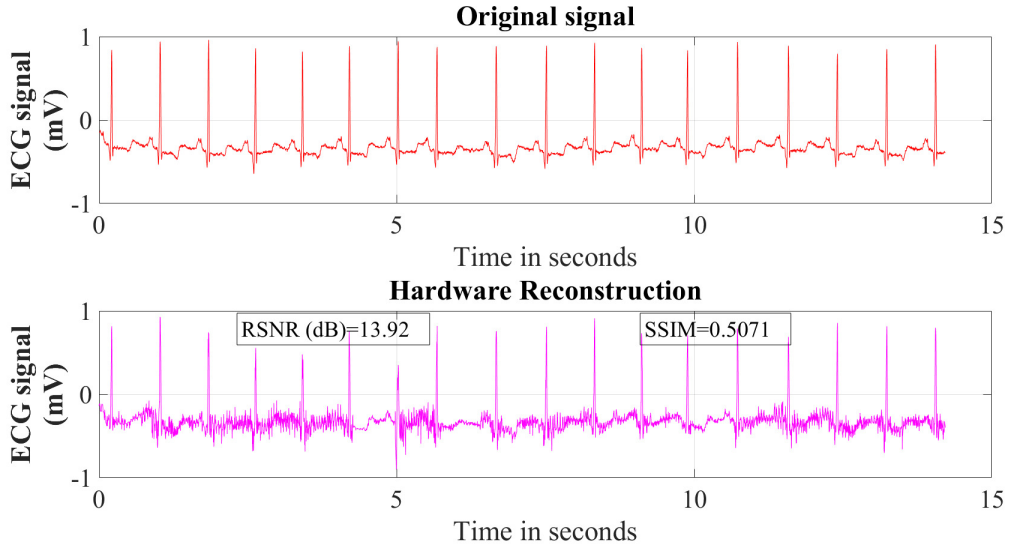


Figure 4.19: ECG hardware reconstruction from 50% measurements

the reconstructed signal achieves an RSNR of 13.92 dB and an SSIM of 0.5071 retaining many of the morphological features present in the original signal, taking a total reconstruc-

tion time of 1.48 ms. This approximately translates to 30× speed up over the software implementation of the SIRP algorithm.

4.4.6 SIRP offloading system

In order to realize a working system implementation of the proposed hardware, a peripheral component interconnect express (PCIe) bus architecture can potentially be used to connect the Virtex VC707 FPGA to the host workstation. The interface is implemented by the Reusable Integration Framework for FPGA Accelerators (RIFFA) [85] which is documented to be a quick working solution to integrate PCIe communication capability to the FPGA accelerators.

The high-level design overview of the proposed system is shown in Fig. 4.20, where the principal components are the the host workstation where pre-processing of the measurements vectors and post-processing of the reconstructed signals from the FPGA would take place. The communication occurs through a PCIe Gen2-8 lane link with the interrupt based PCIe driver and the PCIe controller IP on the FPGA.

The system would be designed to support the pre-processing and post-processing functionalities independently. The reconstruction engine wrapper is attached to the channel interfaces provided by the RIFFA interface, after which RIFFA manages the traffic communicated to and from the FPGA through the PCIe link. The SIRP offloading functionality is responsible for invoking the RIFFA transmit and receive functions and takes care of the pre/post processing operations. The offloading block transmits the measurements and then waits for the corresponding ready signal from the FPGA before sending the next measurement vector.

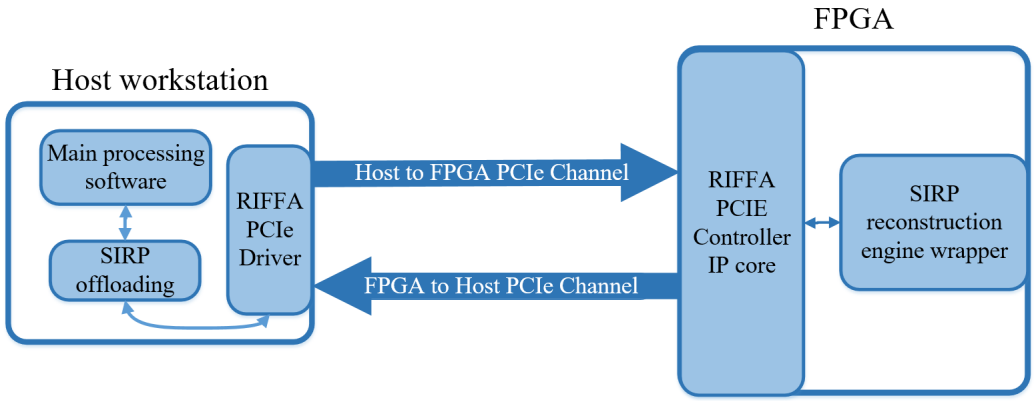


Figure 4.20: High level design overview of the PCIe based CS reconstruction system

4.5 VLSI design

In this section, the proposed high throughput reconstruction engine implemented for a smaller problem size of N=256, M=64 and sparsity level of 16. The architecture has been synthesized and the post-synthesis simulation results are tabulated.

4.5.1 Design Methodology

4.5.1 Front-end design

The SIRP reconstruction engine presented in this section is coded with Verilog hardware description language (HDL) and the functional verification is carried out with measurement test vectors using Synopsys verilog compiler simulator (VCS). The functionally validated HDL code is synthesized from the standard cell libraries of UMC 65 nm CMOS technology by using the Synopsys design compiler (DC) tool. The generated gate level netlist is analysed under worst and best corner cases to avoid setup and hold violations. The netlist after STA verification is subjected to post-synthesis simulations to validate the sparse recovery.

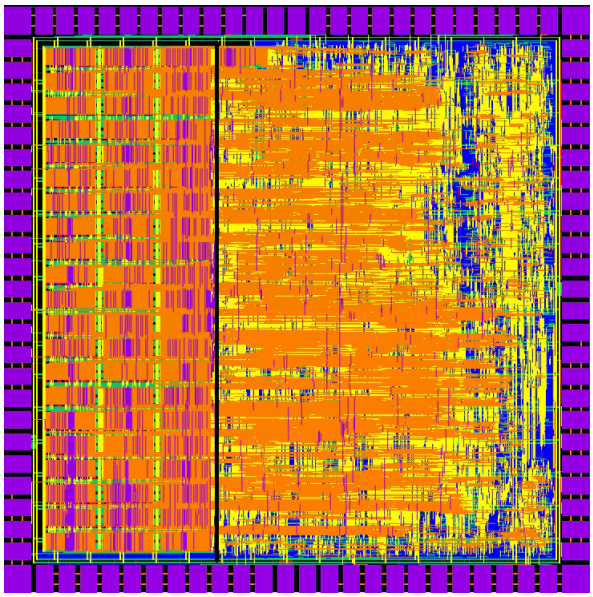


Figure 4.21: Chip layout of SIRP sparse reconstruction engine in 65 nm CMOS technology node

4.5.1 Back-end design

The back-end design is carried out using Cadence Innovus where the synthesized netlist and the synopsys design constraints (sdc) files are input to the back-end tool. UMC 65 nm technology offers six metal layers of which the lower metal layers are used for Vdd and Vss power routing, middle layers for standard cell implementation and the top layers for global routing. The input and output pads as well as the corner pads are placed in the respective places around the standard cell core area. Further to this, clock tree synthesis (CTS) is performed to derive optimal clock tree structures with minimum skew. Once special routing is carried out to connect the standard cells, static timing analysis is performed to ensure absence of setup or hold violations. Finally, core and I/O fillers are placed in the design to avoid gaps between cells and I/O pads. A final STA check is performed to validate timing closure after geometry, connectivity, antenna effects and metal density checks.

Table 4.9: Design metrics after post-synthesis simulation of sparse reconstruction engine in 65 nm CMOS technology node

Design metrics	Values
Hierarchical cell count	426003 standard cells
Combinational area	1.13 mm^2
Non-combinational area	0.66 mm^2
Design core area	2.44 mm^2
Critical path delay	9.9 ns
Maximum clock frequency	101 MHz
Leakage power @ 101 MHz clock frequency	3.56 mW
Dynamic power @ 101 MHz clock frequency	37.42 mW
Total power consumption	40.99 mW

4.6 Summary

This work presents a sparse reconstruction engine implementation on FPGA for fast CS recovery. A hardware friendly version of the SIRP algorithm is proposed by simplifying the regularization and LS update. The architectural design incorporates a linear iterative QRD block to exploit parallelism in the MGS and fasten the augmentation of newly selected columns. The implementation results on a Xilinx Virtex Ultrascale FPGA show that design can operate a clock frequency of 190 MHz permitting reconstructions of 36-sparse signals within 77μs consuming just 14525 processing cycles, which is a significant improvement over the existing works. The implementation is also scalable for different sparsity levels and does not require prior knowledge of signal sparsity. The implementation is capable of reconstructing 13K vectors of size 1024 per second achieving a dynamic power efficiency of 445μW/vector.

Chapter 5

Low complexity architecture for sparsity independent regularized pursuit

The major focus of this chapter will be on refining the sparsity independent regularized pursuit algorithm to induce hardware savings for area or power critical applications without severely impacting the sparse reconstruction capability.

5.1 Introduction

Formulating the iterative signal estimation task as a least squares (LS) problem necessitates efficient matrix decomposition strategies like QR, Cholesky etc. to achieve simplified hardware. The architectures based on Cholesky inversion are known to consume huge resources and reconstruction time for increasing matrix sizes. QR decomposition based on Givens rotations significantly trade-off reconstruction time for low hardware consumption [62], while MGS based QR strategies is capable of swifter recovery at much higher hardware costs.

From a wireless communication perspective, the LS step is equivalent to the zero-forcing receiver in multiple-input multiple-output (MIMO) linear detection, whose performance can be significantly degraded in noisy scenarios due to amplification. The baseband model for this environment with n_t transmit and n_r receive antennas is shown below

$$\mathbf{y} = \mathbf{H}\mathbf{x} + \mathbf{n} \tag{5.1}$$

where $\mathbf{y} \in \mathbb{C}^{n_r \times 1}$ is received from the Rayleigh flat fading channel denoted by $\mathbf{H} \in \mathbb{C}^{n_r \times n_t}$ from transmission of the symbol vector $\mathbf{x} \in \mathbb{C}^{n_t \times 1}$. The channel is assumed to be perturbed by additive white Gaussian noise $\mathbf{n} \in \mathbb{N}(0, \sigma_n^2 \mathbf{I})$.

The symbol vector detection using the zero-forcing receiver can be written as

$$\hat{\mathbf{x}}_{\text{ZF}} = (\mathbf{H}^H \mathbf{H})^{-1} \mathbf{H}^H \mathbf{y} = \mathbf{x} + (\mathbf{H}^H \mathbf{H})^{-1} \mathbf{n} \tag{5.2}$$

Since the ZF receiver is prone to noise amplification when $\mathbf{H}$ is ill-conditioned shown by the last term of equation (5.2).

Contrarily, the minimum mean squared error (MMSE) receiver is capable of suppressing the noise amplification effects when the channel matrix is ill-conditioned . The MMSE receiver is designed by adding an additional term corresponding to the noise variance in the filter, as shown below

$$\hat{\mathbf{x}}_{\text{MMSE}} = (\mathbf{H}^H \mathbf{H} + \sigma_n \mathbf{I})^{-1} \mathbf{H}^H \mathbf{y} \tag{5.3}$$

It can be seen that noise is suppressed when $\sigma_n \gg 1$ and the MMSE receiver approaches the ZF receiver when the noise is minimal. Motivated by this improvement, the LS estimation in the original OMP algorithm was replaced with the MMSE criterion in [45] to address the noise vulnerability of OMP.

This motivates the adoption of the linear least-mean-squares (LMS) strategy in the context of sparse signal estimation to approach the MMSE solution as well. An l_0 regularized LMS formulation called l_0-LMS [86] has been demonstrated to achieve similar performance to the state-of-art Basis pursuit de-noising (BPDN) in noisy frameworks but has significant hardware complexity. The stochastic gradient pursuit (SGP) [45] and sparsity-estimation subspace pursuit (SE-SP) [71] algorithms have adopted the LMS to produce the signal estimate in every iteration of the respective pursuit process. However, SGP pursues the support set serially as in OMP and thereby has a higher reconstruction time, while

SE-SP needs an accurate sparsity estimation phase prior to subspace pursuit.

Since the proposed SIRP algorithm is inherently faster owing to the multi-element pursuit in each iteration, this chapter will focus on replacing the expensive LS update in the SIRP algorithm with a simpler LMS process in order to explore a better trade-off between hardware resources and reconstruction speed. The algorithm refinement, key architectural design aspects and the experimental details will be detailed in the following sections.

5.2 Proposed algorithm

Since noise robustness has remained a challenge in OMP, [45] proposed the stochastic gradient pursuit (SGP) which retains the pursuing process of OMP and substitutes the LS step with the minimum mean squares estimation (MMSE). This blind reconstruction algorithm is still susceptible to measurement noise due to the iterative index updating step similar to OMP. [71] employs a preliminary blind estimation scheme which provides an approximate estimate of the sparsity for SP to operate on, with a 10% iteration overhead. SIRP [81] can perform signal recovery in fewer iterations without requiring sparsity prior. In this section, SIRP is optimized for hardware realization by substituting the complex LS step shown below

$$\hat{\mathbf{x}}^{n}_{\mathbf{\Gamma}^n} = [\mathbf{\Phi}^{T}_{\mathbf{\Gamma}^n}\mathbf{\Phi}_{\mathbf{\Gamma}^n}]^{-1}\mathbf{\Phi}^{T}_{\mathbf{\Gamma}^n}\mathbf{y} \tag{5.4}$$

with a gradient descent approach as discussed in subsection Subsection 5.2.1. The previous chapter aggressively pursued an incremental QRD approach to implement the LS step in order to significantly speed up the reconstruction speed by permitting higher resource usage. The following subsections will focus on adapting the SIRP algorithm to resource constrained environments by trading off the reconstruction speed.

5.2.1 Improved SIRP algorithm

The SIRP algorithm [81] decomposes the pursuit process by forming c clusters of the sensing matrix $\mathbf{\Phi}$ and determining the indices within each cluster that possess the highest correlation with the residual as shown below

$$J_i = \arg\max_j | < \mathbf{\Phi}_j^i, \mathbf{r}^{n-1} > | \quad \forall\ i \in \{1, ..., c\} \tag{5.5}$$

where $\mathbf{\Phi}_j^i$ represents the j^{th} column of the i^{th} cluster. CoSaMP [34] and SP [35] augment the support set by $2K$ and K highly correlated indices respectively and trim it down to K terms after a costly least squares step.

Algorithm 7: Refined SIRP with block LMS update

Input: $\mathbf{b}$, $\mathbf{\Phi}$, l
Initial: $\mathbf{r} = \mathbf{y}$, $\hat{\mathbf{b}} = \mathbf{0}$, $t = 0$, $n = 0$
while $\|\boldsymbol{r}\|_2^2 > T_r$ *or* $t \leq K_{\max}$ **do**
  $J_i = \arg\max_j | < \mathbf{\Phi}_j^i, \mathbf{r}^{n-1} > | \quad \forall\ i \in \{1, ..., c\}$
  Form disjoint subsets $\mathbf{\Upsilon}^p$ of $\mathbf{J} = \{J_1, \cdots, J_c\}$ such that for every subset $\mathbf{\Upsilon}^p$
    where $p = 1, ..., 5\ |\mathbf{\Phi}_a^T \mathbf{r}^{n-1}| \geq \frac{1}{2^p} \max |\mathbf{\Phi}^T \mathbf{r}^{n-1}| \quad \forall\ a \in \mathbf{\Upsilon}^p$
  $\mathbf{\Gamma} = \mathbf{\Gamma} \cup \mathbf{\Upsilon}^1$
  $\tilde{\mathbf{b}}_1 = \hat{\mathbf{b}}_{|\mathbf{\Gamma}}$
  for $\lambda = 1$ to M in steps of l **do**
    $\tilde{\mathbf{\Phi}}_\lambda = \mathbf{\Phi}_{\mathbf{\Gamma}}(\lambda, :)$
    $d_\lambda = y_\lambda$
    $\epsilon_\lambda = d_\lambda - \tilde{\mathbf{\Phi}}_\lambda \tilde{\mathbf{b}}_\lambda$
    $\tilde{\mathbf{b}}_\lambda = \tilde{\mathbf{b}} + \mu \epsilon_\lambda \tilde{\mathbf{\Phi}}_\lambda^T$
  end
  $\hat{\mathbf{b}}_{|\mathbf{\Gamma}} = \tilde{\mathbf{b}}_{M+1}$
  $\mathbf{r} = \mathbf{y} - \mathbf{\Phi}_{|\mathbf{\Gamma}} \tilde{\mathbf{b}}_{M+1}$
  $n = n + 1$
end
Output: $\hat{\mathbf{b}}$

Unlike these parallel pursuit strategies, SIRP regularizes the chosen cluster maximums into subsets $\mathbf{\Upsilon}^p$ based on their correlation energies and applies a criterion to select the

subset with the highest average energy $\boldsymbol{\Upsilon}^1$, as shown in steps 4 and 5 of Algorithm 7. The chosen measurement sub-matrix for the gradient task can be formed as

$$\boldsymbol{\Phi}_{\boldsymbol{\Gamma}} = \{\overrightarrow{\boldsymbol{\phi}_j} | j \in \boldsymbol{\Gamma}\} \tag{5.6}$$

where $\overrightarrow{\boldsymbol{\phi}_j}$ is the j^{th} column of $\boldsymbol{\Phi}$. The contribution of the selected columns has to deducted from the measurements to compute the residual vector. In order to arrive at the residual at minimum hardware costs, it is proposed to minimize the mean squared error instead of the squared error. To this end, the stochastic gradient descent can be used to approach the MMSE solution by reformulating the CS reconstruction problem as a system identification task [45], where $\mathbf{y}_i$ can be understood as desired data, $\boldsymbol{\Phi}_{\boldsymbol{\Gamma}_{\text{opt}}}$ can be regarded as the matrix of input signals and $\tilde{\mathbf{b}}_{\text{opt}}$ represents the coefficients that the adaptive filter strives to optimize, after correctly identifying the optimal support set $\boldsymbol{\Gamma}_{\text{opt}}$.

Denoting the size of support set $\boldsymbol{\Gamma}$ by L, the chosen submatrix of size $M \times L$ forms the input signal matrix for the LMS process. The error ϵ_λ between the desired signal y_λ and the filter output $\epsilon_\lambda = d_\lambda - \tilde{\boldsymbol{\Phi}}_\lambda \tilde{\mathbf{b}}_\lambda$ is used to iteratively approach the MMSE solution as

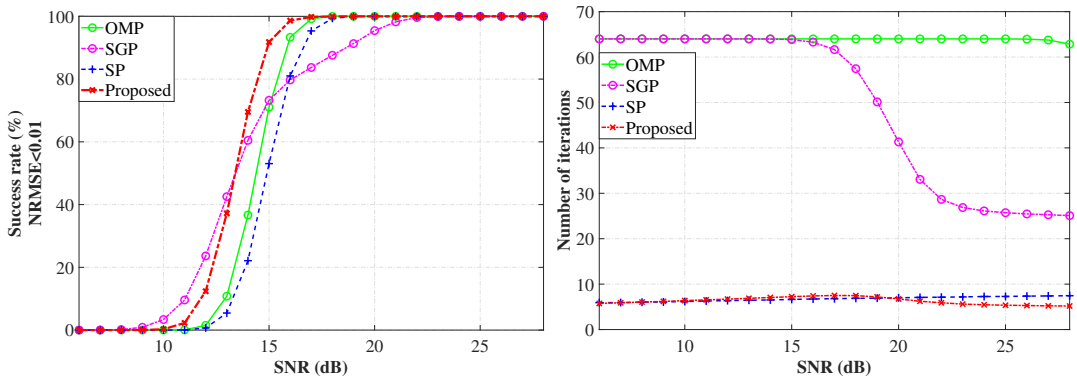
$$\tilde{\mathbf{b}}_\lambda = \tilde{\mathbf{b}} + \mu \epsilon_\lambda \tilde{\boldsymbol{\Phi}}_\lambda^T \tag{5.7}$$

The LMS output is used to compute the residual signal $\mathbf{r}$ as $\mathbf{r} = \mathbf{y} - \boldsymbol{\Phi}_{|\boldsymbol{\Gamma}} \tilde{\mathbf{b}}_{M+1}$ whose squared l_2 norm is used to determine algorithm termination, in addition to $K_{\max}$.

5.2.2 Software Experimental Results

In order to compare the reconstruction performance of the proposed algorithm with state-of-art methods, the parameters of signal length N, measurements M and sparsity level K are fixed to (1024, 256, 32) respectively and the measurement matrix is designed to be random Gaussian with unit normalized columns. The test signal coefficients are chosen from the normal distribution $\mathcal{N}(0, 1)$ having a randomly chosen support set. The SNR of

the measurement process is varied from 6dB to 28dB in steps of 0.5dB with 10,000 trials for each case. The halting criterion of OMP, SP and the proposed algorithm is set to 10^{-2} while it is set according to the criterion in [45] for SGP and the sparsity maximum $K_{\max} = 64$ is passed to the algorithms.



(a) (b)

Figure 5.1: Reconstruction performance comparision (a) Success rate vs SNR (b) Average iterations vs SNR

It can be observed from Fig. 5.1 (a) that the proposed method outperforms OMP, SP and SGP in terms of the success rate curve and requires much fewer iterations than OMP and SGP (Fig. 5.1 (b)).

5.2.3 Computational complexity analysis

In this subsection, the computational complexity of the proposed algorithm is compared with the state-of-art methods. OMP requires $\mathcal{O}(L^2)$ multiplications where L is the size of support set detected by OMP, compared to just $\mathcal{O}(L)$ required by SGP [45], SE-SP [71] and the proposed algorithm. Table 5.1 highlights the multiplication complexities of different steps in the algorithms with signal length N, measurements size M, number of iterations T required by OMP, SGP and SE-SP phase 1, number of iterations t required by SE-SP phase 2 and number of iterations n required by the proposed algorithm. The exact multiplications required by the different algorithms are tabulated in Table 5.2 for various noise levels,

Table 5.1: Computational complexity comparison of multiplications for different reconstruction algorithms

	OMP[7]	**SGP** [45]	**SE-SP**[71]	**Proposed**
Correlation	$\sum_{L=1}^{T} NM$	$\sum_{L=1}^{T} NM$	$\sum_{L=1}^{T} NM + \sum_{k=1}^{t} NM$	$\sum_{k=1}^{n} NM$
LS/LMS process	$\sum_{L=1}^{T} 3L^2 + (M-1)(L-1) - 3$	$\sum_{L=1}^{T} M(2L+1)$	$\sum_{L=1}^{T} M(2L+1) + \sum_{k=1}^{t} M(4K+1)$	$\sum_{k=1}^{n} M(2\lvert\mathbf{\Gamma}_k\rvert + 1)$
Residual update	$\sum_{L=1}^{T} ML$	$\sum_{L=1}^{T} ML$	$\sum_{L=1}^{T} ML + \sum_{k=1}^{t} MK$	$\sum_{k=1}^{n} M\lvert\mathbf{\Gamma}_k\rvert$
Termination*	$\sum_{L=1}^{T} M$	$\sum_{L=1}^{T} M$	$\sum_{L=1}^{T} M + \sum_{k=1}^{t} M$	$\sum_{k=1}^{n} M$

* Halting condition set to 10^{-2} for OMP, SE-SP and proposed method

along with the iterations required and the percentage reduction with respect to OMP. It is seen that SGP [45] reduces the multiplication count of OMP by upto 45%. [71] reported a 10% iteration overhead compared to OMP, which is validated here for the noiseless case and the complexity is exacerbated under noisy scenarios. The proposed algorithm is able to achieve approximately a 5-fold reduction in the required multiplications compared to OMP in the noisy regime and a 3-fold reduction in the noiseless case.

Table 5.2: Required multiplications of different algorithms: $N = 1024$, $M = 256$, $K = 32$, $K_{\max} = 64$

Algorithm	**SNR**		
	20dB	**24dB**	**Noiseless**
OMP[7]	17,977,216 (T=64)	17,801,154 (T=63)	9,826,056 (T=36)
SGP[45]	11,725,056 (T=42) 34.7%	9,676,800 (T=35) 45.6%	9,967,104 (T=36) -1.4%
SE-SP[71]	20,858,880 (T=64, t=8) -16%	20,547,328 (T=63, t=8) -15%	10,883,328 (T=36, t=3) -10%
Proposed	3,468,288 (T=12) **>80%**	3,751,936 (T=12) **>78%**	2,829,568 (T=10) **>71%**

5.2.4 Choice of step size parameter

The choice of step size has a great impact on the adaptation speed and accuracy of the algorithm. [45] arrived at the following bound for μ which guarantees bounded MSE

$$\mu \leq \frac{2}{3} \times \frac{M}{L} \tag{5.8}$$

where the number of filter taps vary in every iteration. It was proposed in [45] to use substitute L with $K_{\max}$ to attain adequate convergence. [71] proposed a hardware efficient way to realize variable step size by rounding L to the nearest power of 2. In our simulations, it was revealed that adopting a fixed step yielded relatively better reconstructions.

5.3 Architectural Design and Implementation

This section presents the architectural design of the modified SIRP algorithm targeting a problem structure of $N = 1024$, $M = 256$ and a maximum sparsity of 64 in order to compare with state-of-art FPGA implementations. The number of clusters c is chosen to be 8 as it offers a good balance between reconstruction speed and hardware resources. Higher values of c have been experimentally determined to not significantly improve by the reconstruction speed while linearly increasing the hardware resources needed.

Fig. 5.2 shows the overall hardware architecture of the proposed reconstruction engine composed of the following major blocks: 1) Measurement matrix memory bank size managed by a dedicated controller; 2) Cluster maximum compute block that outputs the highest correlation values and corresponding indices of 8 clusters, in which CMC0 shares hardware for the subsequent LMS and residual update steps as well; 3) Regularizer block that outputs the optimal subset of the selected indices and 4) LMS row access memory bank; and 5) Residual register bank which stores the measurements on initialization and the updated residuals of subsequent iterations. The control unit regulates the flow of data,

beginning from the input of measurements into the system till the reconstructed signal is generated.

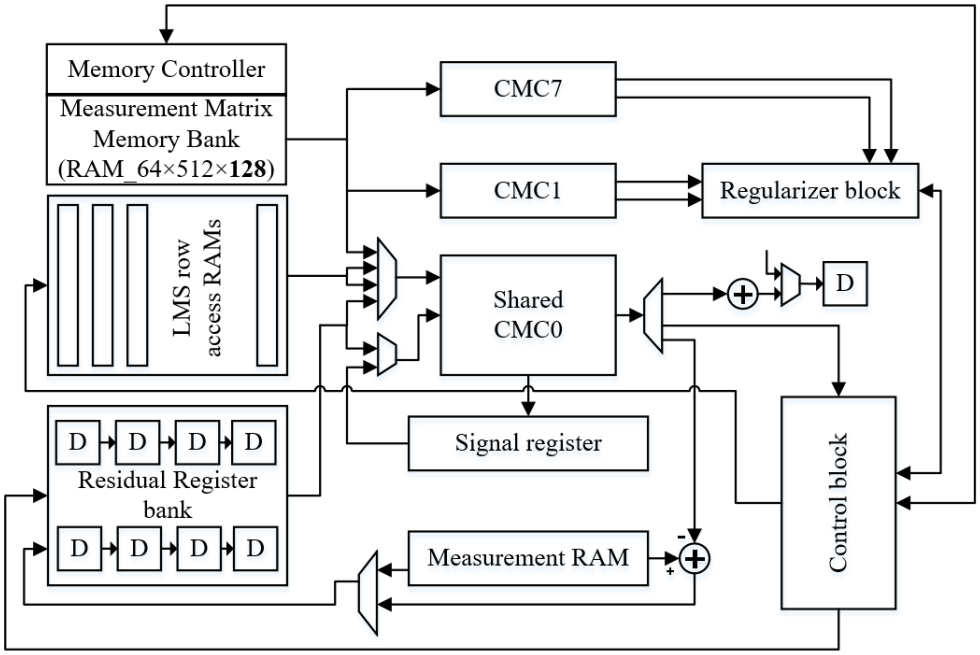


Figure 5.2: Proposed hardware architecture for the modified SIRP algorithm

5.3.1 Measurement matrix memory organization

In traditional hardware implementations of pursuit strategies, each row of the 256×1024 measurement matrix is stored in a separate RAM such that each column (from index 0 to 1023) can be accessed in 1024 successive clock cycles. The SIRP divide-and-conquer strategy permits parallel processing of 8 clusters, which can speed up the correlation process.

However, this would give rise to the requirement of 2048 block RAM instances to access the columns of 8 clusters concurrently, which is very significant. Further, the architecture would also entail 2048 parallel multipliers to compute the correlation of each cluster's columns with the residual, which considerably increases the hardware burden.

In order to mitigate the hardware burden involved in naively implementing the SIRP parallel pursuit strategy, the proposed architecture folds the correlation for 8 clusters by a factor of 4, bringing down the number of multipliers allotted for each cluster to 64. This in

turn relaxes the memory access requirements as only 64 elements of each cluster need to be accessed parallelly. In order to satisfy the parallel access requirements, the storage of $\mathbf{\Phi}$ is implemented on block RAMs of data width 128 (by combining the 16-bit data widths of the 8 clusters) and depth 512. Since each cluster consists of 128 columns and each column can now be processed in 4 clock cycles due to folding strategy, the chosen depth of 512 is justified. Since parallel access to every 64 elements of the columns of each cluster, 64 such RAMs would be sufficient to generate the required data for the parallel processing of the clusters. The structure of a 512×128 block RAM is shown in Fig. 5.3 to clearly highlight the memory organization.

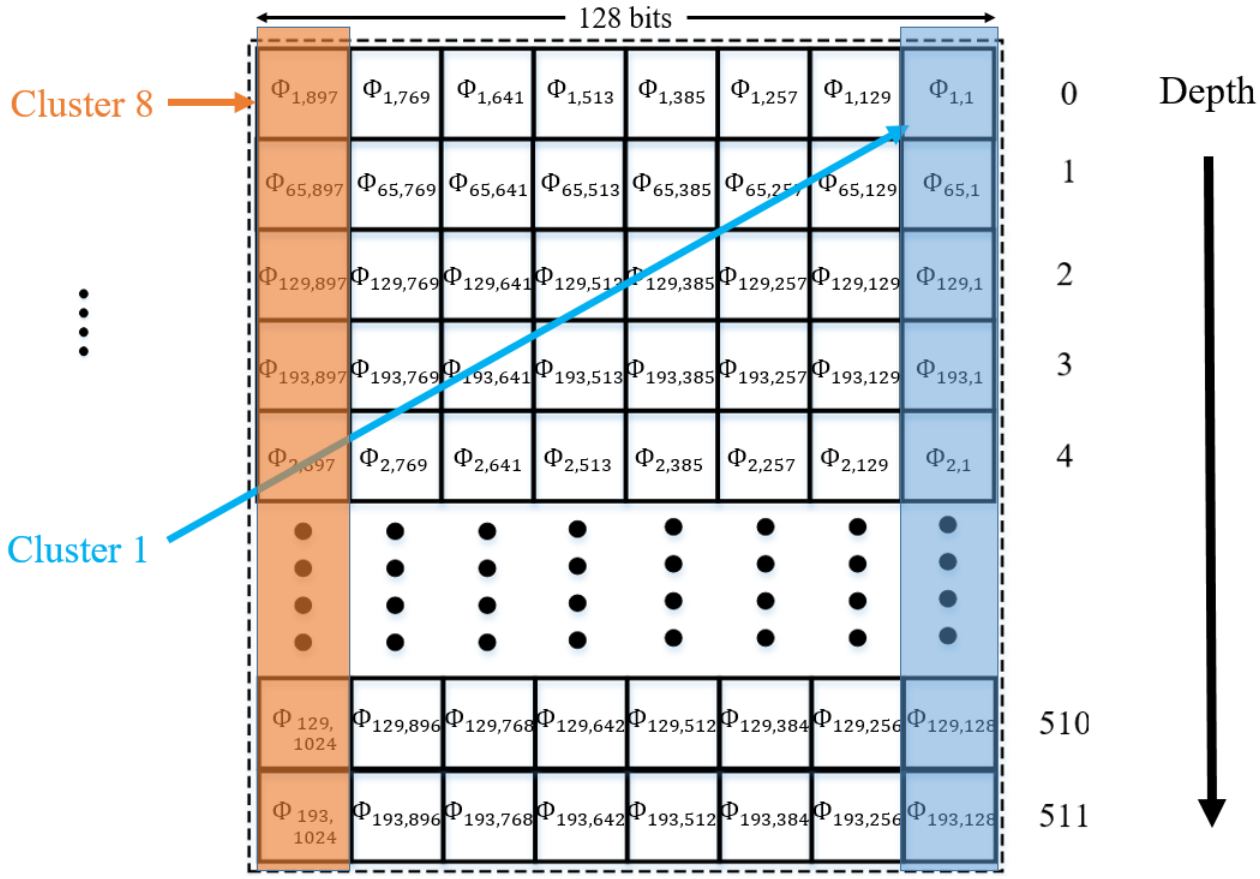


Figure 5.3: Internal organization of a 512×128 block RAM showing the stored elements

It can be seen that the lower 16 bits correspond to the 1st, 65th, 129th and 193rd rows of Cluster 1 containing columns 1-128. Similarly, the upper 16 bits correspond to the 1st, 65th, 129th and 193rd rows of Cluster 8 containing columns 897-1024. Similarly, another

such block RAM would be needed to store the 2nd, 66th, 130th and 194rd columns of the respective clusters. Therefore, the entire measurement matrix storage can be carried out with 64 such block RAMs, as shown in Fig. 5.4. The measurement matrix is stored in memory in this fashion so that the folded correlation operation of each column with the residual can be undertaken in 4 clock cycles.

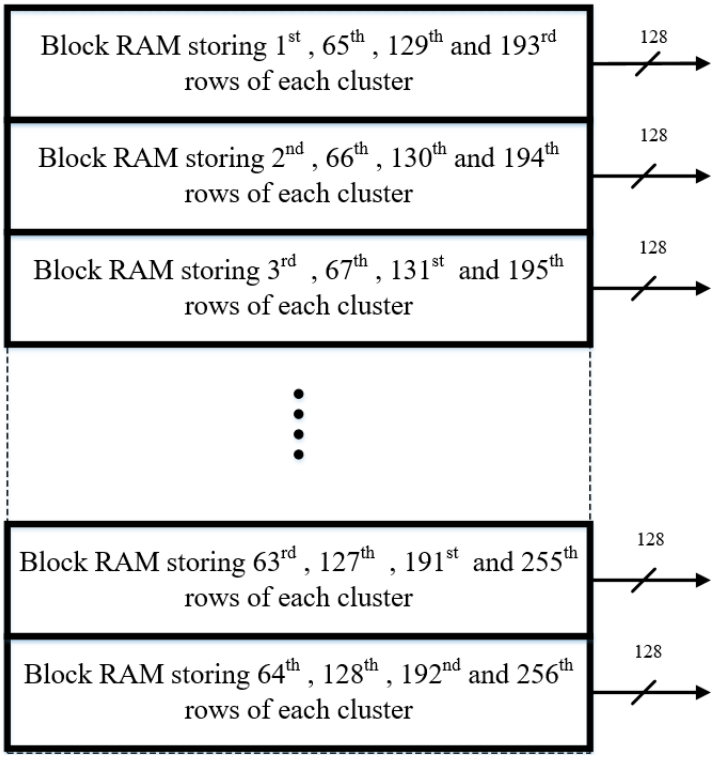


Figure 5.4: Memory bank structure of the measurement matrix Φ

In order to control the access to this complex memory bank structure, a dedicated memory controller is implemented by a 9-bit counter to access the 512 locations of the 64 block RAMs. The count enable and load signals of the counter are triggered by the type of operation being performed i.e., correlation or specific memory access for LMS write cycles.

5.3.2 Cluster maximum compute (CMC) block

In order to induce hardware savings, the parallel computation of the cluster maximums are folded by a factor of 4 in tandem with the memory organization of the measurement matrix.

The folding architecture of the CMC block is depicted in Fig. 5.5 for the case of the cluster 0 which includes the first 128 columns of the measurement matrix. Each CMC block consists of $M/4$ parallel multipliers with a latency of t_{mul}=1 clock cycles and an $M/4$-input pipelined adder tree (with a latency of t_{add}=3 clock cycles) that yields the partial correlation. For instance, in time slot t_0, the first 64 elements of the 1st column of $\mathbf{\Phi}$ are correlated with the first 64 elements of the residual vector (which is available from the residual register bank shown in Fig 5.2). This is registered after ($t_{mul} + t_{add}$) clock cycles and added to '0' to yield the partial correlation product pcp_0 which is again stored in 16-bit registers. Since the architecture is fully pipelined, the partial correlation product corresponding to the subsequent 64 elements of $\boldsymbol{\phi}_0$ and $\boldsymbol{r}$ from time slot t_1 will be available in the next clock cycle. The multiplexer ensures that the new partial pcp_1 is added to the previously computed pcp_0. In this way, the partial correlations accumulated for 4 clock cycles as well as the corresponding index are then passed to a comparator unit which computes the cluster maximum and its index, which are then passed to the regularizer unit. The control unit ensures that the accumulation register is cleared before the next column is inspected.

Since there are 128 columns in a cluster, the residual correlation for all clusters can be completed in 512+t_{mul}+t_{add} clock cycles. In order to promote hardware reuse, the cluster maximum compute block 0 is also utilized to perform the LMS update operations as will be discussed later.

5.3.3 Regularizer block

The regularizer block is tasked with determining the optimal indices to be augmented to the support set, which is done in two stages. The first stage determines the highest value among the chosen cluster maxima and the second stage uses this global maximum to partition the eight selected indices into groups based on the rule in step 4 of Algorithm 1. The cluster maximum finder is implemented by a simple comparator circuit in 8 clock cycles using a

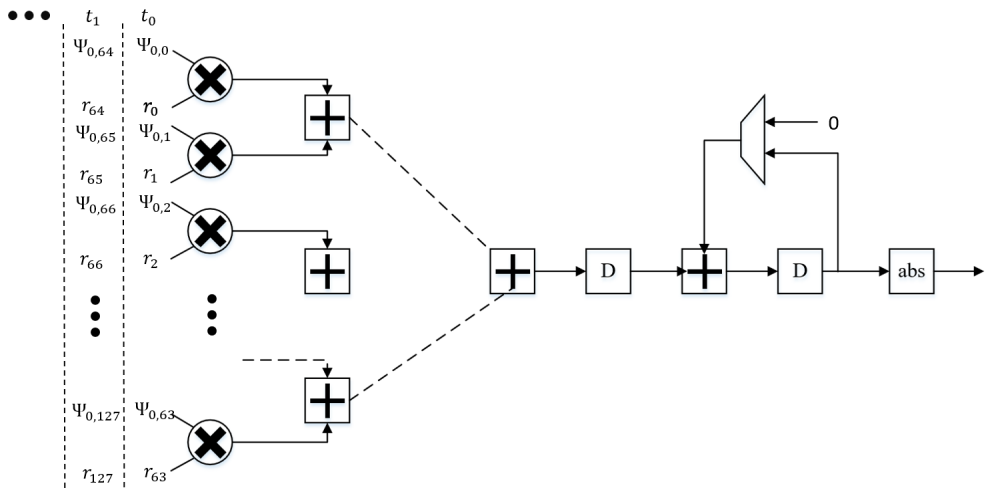


Figure 5.5: Folding architecture of the cluster maximum compute block

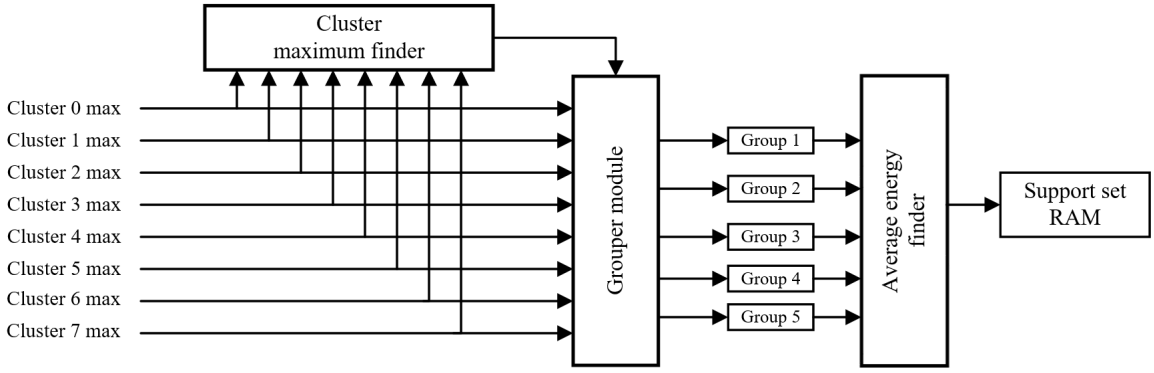


Figure 5.6: Architecture of the regularizer block

8:1 multiplexer and a 3-bit counter, as shown in Fig. 5.7.

The grouper module whose internal architecture is shown in Fig. 5.8 uses a set of comparators to check the range of the cluster maximum with respect to $\frac{1}{2}$ intervals of the global maximum and stores the cluster values in corresponding group RAMs.

After partitioning the indices, the regularizer block determines the group possessing the highest average energy using LUT based dividers whose architecure was shown in Fig. 4.10 and stores them into a support set RAM.

Case study

In the first iteration, the residual vector of size 256×1 is correlated with columns in clusters of $\mathbf{\Psi}$ concurrently, resulting in an overall latency of 512 clock cycles. The CMC block

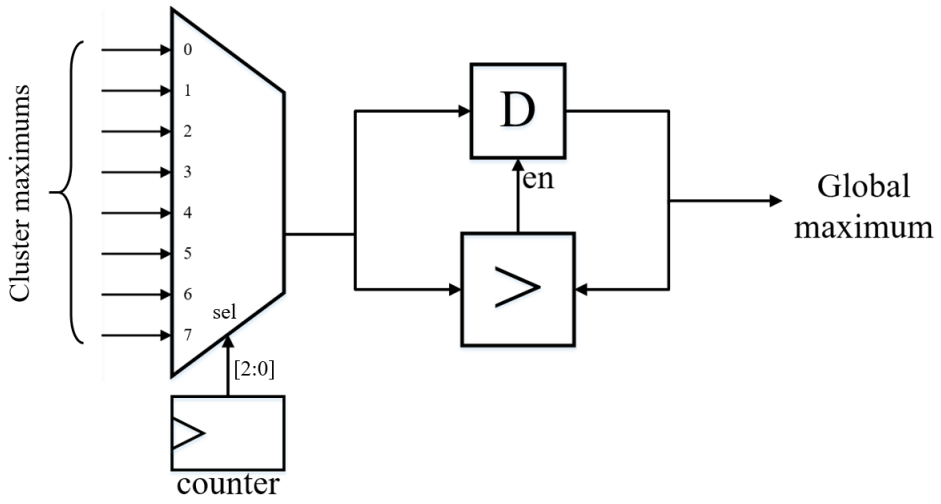


Figure 5.7: Architecture of the cluster maximum finder

stores the value and index of the column within each cluster exhibiting the highest correlation energy to form the initial candidate set of size 8, which in this case study is determined to be $\boldsymbol{J}$= {30, 95, 137, 376, 419, 532, 823, 914}. The cluster maximums are compared sequentially to determine the overall maximum at index 419. This is used to construct the subsets according to the following intervals $[u_{\max}, \frac{u_{\max}}{2}]$,$(\frac{u_{\max}}{2}, \frac{u_{\max}}{4}]$,$(\frac{u_{\max}}{4}, \frac{u_{\max}}{8}]$,$(\frac{u_{\max}}{8}, \frac{u_{\max}}{16}]$ and $(\frac{u_{\max}}{16}, 0]$. The subset with highest average correlation energy is found to be $\boldsymbol{\Pi}_1$={419, 95, 532, 914, 137, 376} corresponding to the interval $[u_{\max}, \frac{u_{\max}}{2}]$ and forms the final candidate set of the first iteration. It is observed that all the chosen indices belong to true support set of the sparse signal. These are written to a support memory for later processing and concludes the operation of the RU in 24 clock cycles. The subsequent LMS and residual update steps are carried out based on the updated set.

5.3.4 LMS row access memory bank

In order to carry out the LMS operation for the selected columns in each iteration, access is required to each row of the corresponding sub-matrix. This cannot be directly obtained from the $\boldsymbol{\Phi}$ storage as it has been configured to provide parallel column access. Therefore, a set of LMS row access RAMs are employed to provide the row data in parallel. However,

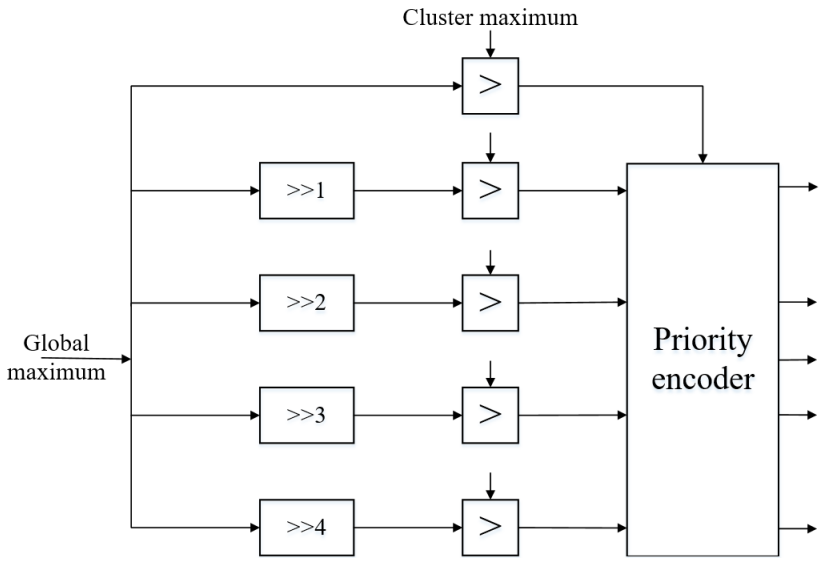


Figure 5.8: Architecture of the cluster maximum finder

writing the data retrieved from the $\mathbf{\Phi}$ matrix would require M clock cycles for each newly added column. However, this data writing process is cleverly interleaved with the LMS update operation so as to not have any detrimental effect on the processing latency, as will explained later.

5.3.5 LMS update operation of the CMC block

The complex task in the algorithm is the LMS update which is governed by two major update equations given below

$$\epsilon_\lambda = d_\lambda - \tilde{\mathbf{\Phi}}_\lambda \tilde{\mathbf{b}}_\lambda \tag{5.9}$$

$$\tilde{\mathbf{b}}_\lambda = \tilde{\mathbf{b}} + \mu \epsilon_\lambda \tilde{\mathbf{\Phi}}_\lambda^T \tag{5.10}$$

From equation 5.9, the term $\tilde{\mathbf{\Phi}}_\lambda \tilde{\mathbf{b}}_\lambda$ is the inner product between the sub-matrix restricted to the current support set $\mathbf{\Phi}_\lambda$ and the corresponding estimated signal. Since $\mathbf{\Phi}_\lambda$ changes in respect to each row of $\mathbf{\Phi}_\lambda$, the CMC block 0 which performs an inner product operation is reconfigured to support the LMS update operations as well to save hardware resources.

This is deemed sufficient as the sparsity is assumed to remain below 64 which implies the shared CMC block can be used to estimate signals with 64 non-sparse coefficients at most.

Equation 5.10 performs a scalar multiplication first on the row of $\mathbf{\Phi}_\lambda$ followed by updating the previous estimate. The shared CMC block is configured to support this operation as well, to ensure re-utilization of resources. Since (5.9) and (5.10) need to be recursively computed, the signal estimate $\hat{\mathbf{b}}$ can be obtained every 3 clock cycles [71]. However, the critical path would be dominated by the size of the adder tree which would be $K_{\max}$ in this case. In order to reduce the critical path delay, a three stage pipelined $K_{\max}$ parallel multiply adders and a $K_{\max}$-input adder tree unit is designed as shown in Fig. 5.9, which would also function as the cluster maximum compute (CMC) block for cluster 0 during the correlation stage. The parallel multiply adders first compute $\mathbf{\Phi_\Gamma}$ in 3 clock cycles to yield ϵ_j which is then used to compute the term $\mu\epsilon_j$ in the subsequent clock cycle. The signal estimate $\mathbf{b}$ is then updated in 3 additional clock cycles using the same parallel multiply adders contributing to a latency overhead of 8 clock cycles for each LMS update of the rows of $\mathbf{\Phi_\Gamma}$. Since it is required to iteratively update M rows of $\mathbf{\Phi_\Gamma}$, the total cost of the LMS process amounts to $8M$ clock cycles. Since the LMS process requires parallel access

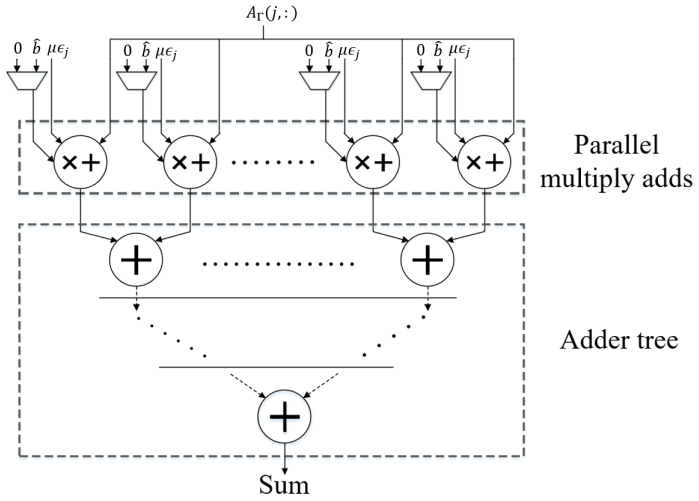


Figure 5.9: Structure of the shared cluster maximum compute block

to each row of $\mathbf{\Phi_\Gamma}$, a novel technique is needed to transform the column access nature of

the selected columns from the memory bank to a row access nature. Since the maximum sparsity supported in [45] is very low, it employed a 2-D register bank to perform this transformation without performing any transpose. [71] employs global multi-task buffers in their design to transpose the newly selected columns using additional clock cycles at considerable area reduction by using cache, and the LMS process is designed to execute in $3M$ cycles after all the columns are stored in cache.

Since the LMS process in the proposed pipelined design costs $8M$ cycles and the upper bound on the number of newly chosen columns is 8, the system is designed to simultaneously store entries of newly selected columns to the LMS row access dual-port RAMs and read the previously stored row for the LMS computation. This technique helps us to avoid any additional hardware or clock cycles to perform column transpose by exploiting the latency associated with the proposed pipelined LMS update. On completion of the LMS process, the shared CMC block sequentially computes the new residual in $M + 3$ clock cycles and then the squared norm is computed in 4 clocks to determine algorithm termination.

In order to enhance the throughput of the proposed reconstruction engine by reducing the cycles per iteration, an LMS alternate row update strategy is proposed whose reconstruction performance was experimentally verified to be near that of the standard LMS. This slight trade-off in recovery performance brings a 2-fold reduction in the LMS cycle cost.

5.3.6 Residual register bank

The residual register bank is designed in order to support the proposed folding architecture by self-feeding 4-stage register chain with the added capability of updating the bank with new residual values at the end of each iteration. Fig. 5.10 clearly depicts the operation of the residual register bank.

There are three principal modes of operation for the register bank which are controlled

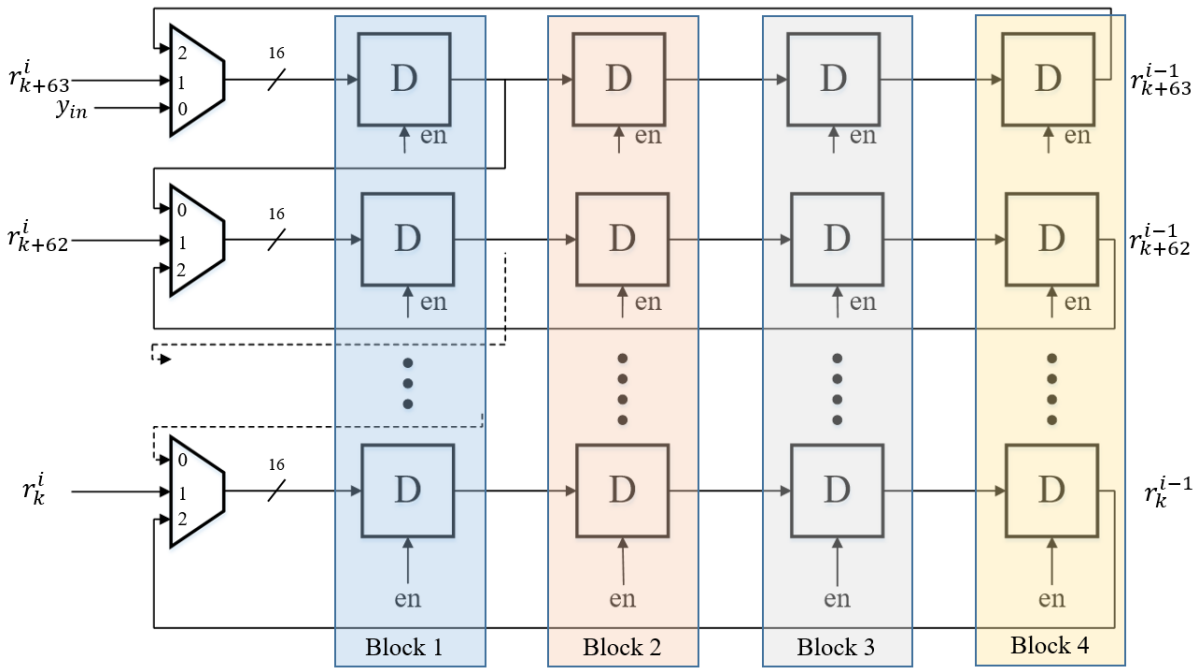


Figure 5.10: Architecture of the residual register bank

by the select signals to the 3:1 multiplexers. Mode 0 is configured when the system is initialized and begins receiving the external measurements sequentially. The first measurement $\boldsymbol{y}_{\text{in}}(0)$ is registered after the first clock cycle and passed to bottom adjacent register as input. This implies that at the end of 64 clock cycles, 64 measurements will be available at the output of Block 1 registers. In the subsequent clock cycle, the enable signals of Block 2 will be asserted so that all 64 measurements are registered by the block and then the enable signals are de-asserted. The next 64 measurements are sequentially filled up in Block 1 registers in a likely fashion and at the end of 64 more clock cycles, the enable signals of blocks 2 and 3 are asserted so that the corresponding shift can take place. This process continues until all 256 measurements are loaded in to the register bank.

Mode 1 is configured by the control block when the new residual produced by the shared CMC block in 4 successive cycles is loaded into the register bank in 4 similar cycles by appropriately asserting the enable signals.

The residual bank operates in mode 2 when it is being parallelly correlated with the columns of each cluster and it is needed to continuously cycle 64 residual elements at a

time according to the folding architecture of the CMC blocks.

5.3.7 Control Block

The control block is tasked with controlling the operations of all the above mentioned blocks. The complex scheduling involved in the implementation of this algorithm is made possible by a carefully designed Moore finite state machine (FSM) that controls the underlying hardware resources through a series of well planned states. The state machine diagram is shown in Fig. 5.11 with 8 principal states of operation. State S0 is the system

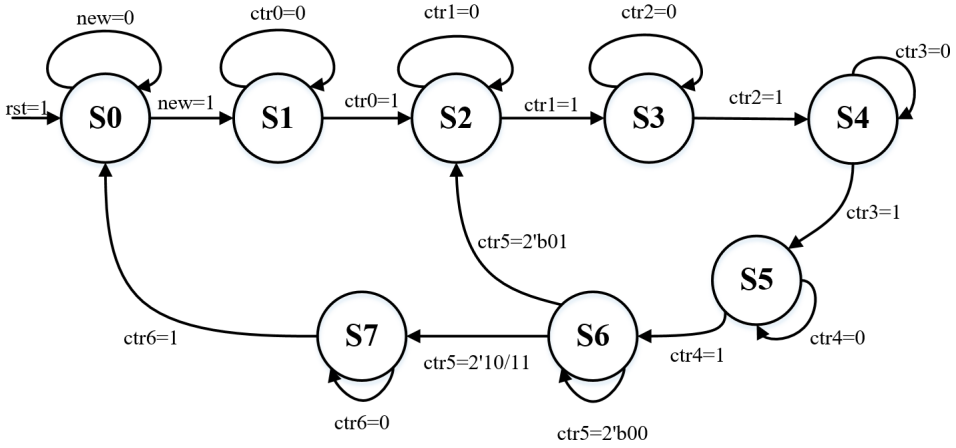


Figure 5.11: State machine diagram of the control block

initialization state in which the internal registers and memories are initialized upon receiving the system reset signal. The FSM waits in this state until the new signal is triggered which indicates that new measurements are available to be loaded into the system.

State S1 waits till all the residual vector elements are loaded in to the residual register bank. The register bank is controlled by the FSM in a self-feeding structure such that it produces 64 elements of the residual vector in subsequent clock cycles when operations are required to be performed using the residual. The latency of state S1 is 256 clock cycles corresponding to the number of elements in the residual vector.

State S2 corresponds to the correlation phase where all the cluster maximum blocks work parallelly to determine each cluster's maximum value and index. The FSM also pro-

duces the necessary control signals to cycle the residual register bank values for the folded architecture operation. This process takes 512 clock cycles to complete before the control signal is asserted to proceed to the subsequent state. State S3 is tasked with controlling the regularizer block to yield the optimal subset in 24 clock cycles.

State S4 is responsible for controlling the LMS update operation of the shared CMC block. Prior to that, the data necessary for this needs to be written to the LMS row access memory. As mentioned in the previous section, the data write and read cycles to and from the LMS row access memory are interleaved in a way to avoid memory collisions. These memory access cycles are managed by two separate read and write counters that address the specific locations of the dual port LMS RAMs. Since the LMS update operation takes $8M$ clock cycles overall, the writing mechanism cleverly overlaps this timeline.

State S5 is responsible for handling the residual update operation which consumes 260 clock cycles. The control block issues the necessary control signals to the shared CMC block and the residual register bank to correctly perform the residual update operation. State S6 manages the residual norm computation task by again manipulating the shared CMC block with the appropriate control signals. If the residual norm remains above the threshold, the FSM goes to state S2 to initiate a new iteration to detect more indices of the support set. State S7 is tasked with generating the estimated signal and proceeds to the original wait state S0 for processing the next measurement vector. Fig. 5.12 depicts the timing diagram of the finite state machine showing the latencies of each state described above.

5.4 Implementation results

5.4.1 Fixed-point analysis

The performance of the proposed algorithm is evaluated in terms of the success rate versus SNR for different fractional precision widths. The fixed point simulation is executed for

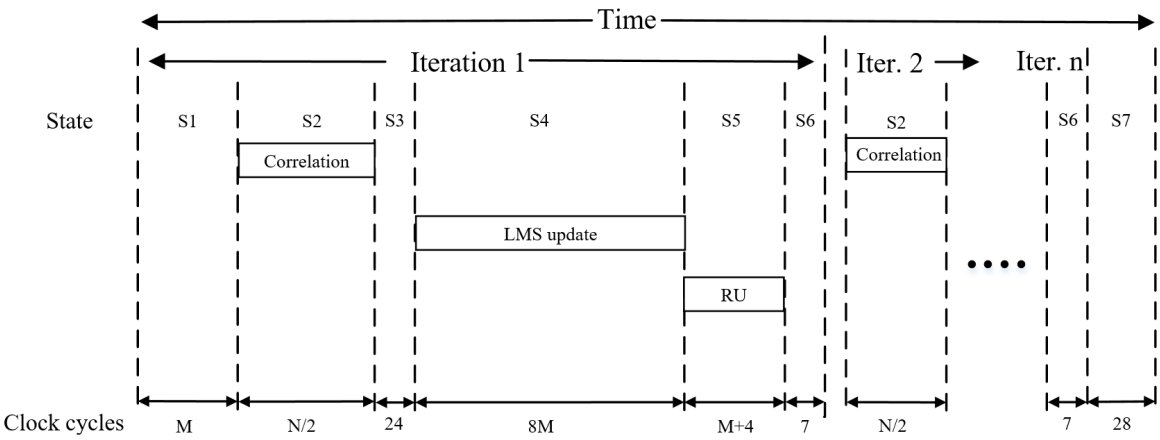


Figure 5.12: Timing diagram showing the latencies of each state of the FSM

10000 trials keeping all other parameters same as in Fig. 5.1. It can be seen that as the fractional precision decreases below 11 bits, the performance begins to degrade and the algorithm completely fails for fractional precision of 8 bits and below. Based on the experiments, the entries of $\mathbf{\Phi}$ are quantized to 16-bit word length with 13 fractional bits before storing into the memory bank. The word length of $\mathbf{y}$, $\hat{\mathbf{b}}$, $\mathbf{r}$ and other intermediate computations are assigned word lengths based on the distribution of the non-zero coefficients of $\mathbf{b}$ with 13 fractional bits.

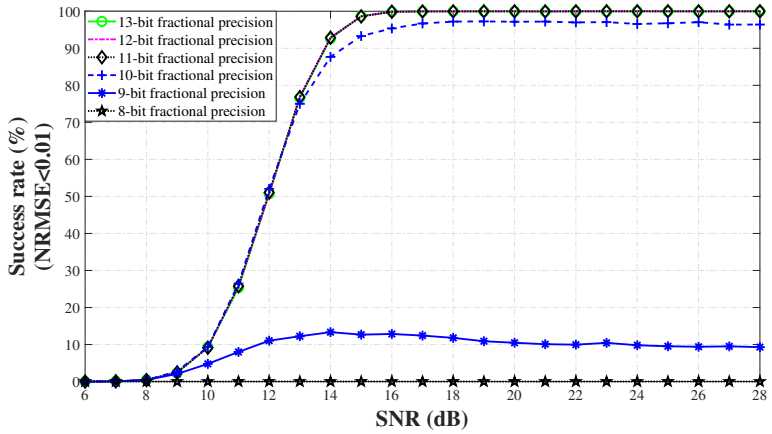


Figure 5.13: Recovery performance of the proposed algorithm under varying fractional precision

Table 5.3: Comparison with state-of-art designs

Reference & Year	**Algorithm**	**Problem Size**			**Time**	**Frequency**	**Format**	**Target**	**Resource Utilization**			**Process**	**RSNR / PSNR**	
		M	N	K	(μs)	(MHz)	Qm.n	**FPGA**	BRAM	DSP	Slices	**cycles**	(dB)	
[57] ('12)	OMP	256	1024	36	622	100	25-bits	Virtex 6	258	268	32001	-	23.5	-
[59] ('15)	OMP	256	1024	36	340	119	Q9.9.	Virtex 6	576	589	6208	40788	-	38.9
[67] ('18)	Imp. OMP	256	1024	36	170	135.4	Q12.12	Virtex 6	342	1544	7860	23009	31.04	-
[68] ('19)	OMP	256	1024	36	327	133.33	Q6.12	Virtex 6	430	386	10902	43604	18.34	-
[69] ('19)	OMP	256	1024	36	238	210	Q6.12	Kintex 7	386	523	28443	50030	-	28.55
[73] ('20)	SP	58	256	8	22.5	15.4	Q15.13	Virtex 7	0	3324	46268	-	28.2	-
Proposed	SIRP	256	1024	36	185	143	Q3.13	Virtex U+	321	518	3600	26244	26.31	39.86
Proposed alternate LMS	SIRP	256	1024	36	132	143	Q3.13	Virtex U+	321	518	3600	26244	22.93	30.67

5.4.2 Implementation results

The proposed CS reconstruction engine supporting a signal length N=1024 and M=256 is implemented on a Xilinx Virtex Ultrascale+ xcvu9p FPGA in order to comapare with existing designs targeting a similar problem size. The implementation and resource statistics are detailed in Table 5.3. The results for recovery of a 36-sparse random gaussian signal show that the proposed engine can operate at a maximum frequency of 125 MHz consuming 26,244 clock cycles resulting in a total reconstruction time of 210 μs. Due to replacement of the complex LS update with the LMS process, the hardware complexity is greatly reduced as seen from Table 4.4. Compared to prior designs, the configurable logic block (CLB) and DSP slice usage is considerably reduced owing to the simple structure of the LMS update. For instance, the improved OMP implementation on a Virtex 6 device in [67] requires considerably higher DSP and logic resources to achieve a reconstruction time of 170 μs. The OMP implementation in [68] consumes slightly lesser DSP resource and higher logic slices on a Virtex 6 FPGA to complete sparse recovery in 327 μs which is outperformed by a factor of roughly 2 in the first variant of the proposed method. The square-root free LS version of OMP implemented in [**Ge'19**] consumes much higher logic slices on the Kintex 7 device. The hardware sharing approach adopted for the various steps in the algorithm contribute to the reduced hardware burden.

The low latency variant of the proposed method formulated by allowing alternate row updates to the signal estimate improves reconstruction time by a factor of approximately 1.5 over the primary variant. Due to the fewer number of rows that are used to estimate the signal, a slight degradation in reconstruction quality is observed with respect to the full update version. The dynamic power consumed by the implementation is estimated by the Vivado Power Analyzer to be 1.56 W which translates to a dynamic power efficiency of 288 μW/vector.

Table 5.4 tabulates the reconstruction quality and time taken by the proposed full and

alternate row LMS versions for varying degrees of sparsity. It is observed that the optimization improves reconstruction time by mildly scarifying the RSNR, which is still above the acceptable RSNR limits for CS applications like ECG [71]. The decrease in SNR does not correspond to distortion in the principal features of the signal, but merely to the difference in coefficient values as in Fig. 5.14.

Table 5.4: Performance under varying K with fixed N=1024 and M=256

	Parameters	K**=8**	K**=16**	K**=24**
Proposed	RSNR (dB)	26.6	29.45	28.52
	Recovery time (μs)	73.3	117.5	161.6
Proposed alternate LMS	RSNR(dB)	25.2	25.5	25.8
	Recovery time (μs)	62.6	90.3	118

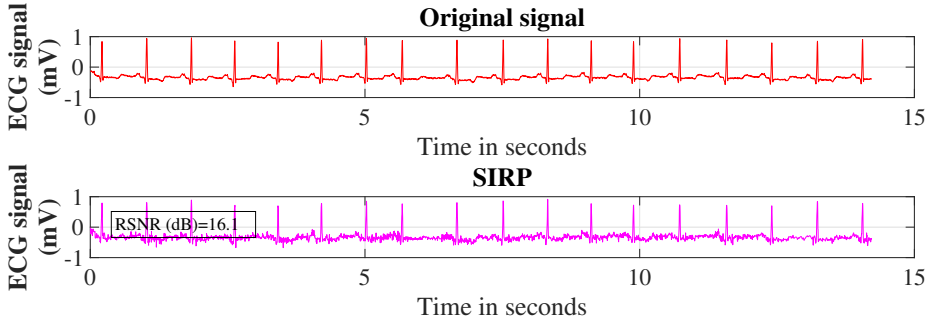


Figure 5.14: Reconstruction of ECG signals in MIT-BIH database from 50% measurements with wavelet sparsifying basis[82]

5.5 Summary

In this chapter, a modified sparsity independent regularized pursuit is presented that substitutes the complex LS update in the original algorithm with the iterative LMS update to significantly reduce the computational burden. A pipelined hardware-sharing architecture is proposed to implement the interdependent steps of the algorithm. The key feature of the proposed architecture is the interleaving of the LMS memory data write operations with the LMS update operations, thereby bypassing the need for additional buffers to perform transpose operations on the measurement matrix columns. The proposed implementation

operating at 143 MHz can complete the reconstruction of 36-sparse signals in approximately 185 μs which is competitive with the state-of-art. An alternate row based LMS update sheme is proposed to reduce the latency cost and leads to a 28% improvement in the recovery time with a 12% degradation in the RSNR with respect to the full LMS update.

Chapter 6

Conclusion and future work

6.1 Conclusion

In recent years, the area of Compressed Sensing has garnered significant attention in various application domains like communication, radar, biomedical and imaging applications etc. CS offers a radical approach to information acquisition by sampling the signal in a basis incoherent with its sparsifying basis. While CS has contributed to lowering the storage and energy requirements of the sampling nodes, fast and efficient reconstruction is necessary to take timely decisions by analysing the reconstructed signals. In this context, the choice of CS recovery algorithms and their hardware feasibility to facilitate the demand of quick reconstruction becomes very crucial.

In order to address the challenges of low recovery speed and hardware complexity in CS reconstruction, this work presents a two-pronged approach of developing a novel sparsity independent regularized pursuit algorithm and designing high-throughput and low complexity architectures by appropriate hardware specific optimizations to the proposed algorithm.

6.1.1 Contributions

The contributions of this research are summarized as follows.

The first part of the research work focuses on novel sparse recovery algorithm development that is indispensable to designing efficient real-world hardware architectures. In this respect, the sparsity independent regularized pursuit algorithm is proposed which employs

a novel identification and enhanced regularization step to considerably speed up pursuit while ensuring recovery performance at par with the state-of-art algorithms. The rigorous experimental evaluation establishes it's independence of the sparsity prior, robustness in noisy frameworks, significant speed-up and feasibility for real world problems. These merits pave the way for realizing efficient hardware implementations in the future that can offer swifter off-line recovery for real-world applications.

In order to significantly improve the reconstruction speed, the work presents a sparse reconstruction engine implementation on FPGA for fast CS recovery. A hardware friendly version of the SIRP algorithm is proposed by simplifying the regularization and LS update. The architectural design incorporates a linear iterative QRD block to exploit parallelism in the MGS and fasten the augmentation of newly selected columns. The implementation results on a field programmable gate array show that design contributes to a two-fold improvement in the reconstruction speed over the existing works tackling identical problem sizes. The implementation is also scalable for different sparsity levels and does not require prior knowledge of signal sparsity. It also contributes to significant energy savings by achieving a low dynamic power efficiency per reconstructed vector. The work also presents an application specific integrated circuit (ASIC) implementation which has undergone post-synthesis validation and can achieve further reduction in the dynamic power consumption.

An approach towards the low complexity implementation of the SIRP algorithm with a least mean squares (LMS) strategy is proposed in order to minimize the hardware consumption. A pipelined hardware-sharing architecture is proposed to implement the interdependent steps of the algorithm. The key feature of the proposed architecture is the interleaving of the LMS memory data write operations with the LMS update operations, thereby bypassing the need for additional buffers to perform transpose operations on the measurement matrix columns. Compared to prior designs, the logic and DSP slice usage is considerably reduced owing to the simple structure of the LMS update. Also, an alternate row LMS

scheme is introduced in the design requiring no additional hardware resources, reducing the cycles per iteration to improve reconstruction speed by approximately one-third permitting a slight reduction in the reconstructed signal-to-noise ratio (RSNR) compared to the full row LMS update.

6.2 Future scope

Although significant work has been carried out towards developing a novel fast decoding CS algorithm and designing efficient architectures for achieving the desired design goals of reconstruction speed and low hardware complexity respectively, there is considerable scope for further work in this domain. A brief discussion of this is given below.

1. The proposed SIRP algorithm is well suited to the general CS framework regardless of the nature of the signal or the sensing mechanism. However, the signals in certain niche applications like spectral occupancy estimation [87] exhibit block sparse nature. A future direction would be to investigate modifications to the algorithm that could exploit this structured sparsity to improve signal estimates.

2. Another possible direction of research could be to tune the proposed algorithm and optimize the hardware for specific CS applications like radar detection or wireless channel estimation.

3. The proposed hardware can currently handle reconstruction of real data, but it would be interesting to further understand the challenges and address the issues in the reconstruction of complex data, which would be necessary when the sparsifying domain is the Fourier basis for instance.

4. Since l_1 minimization provides stable reconstructions in noisy frameworks with strong theoretical guarantees, another interesting direction would be to identify or propose hardware friendly basis pursuit decoding algorithms for CS

List of Publications

Journal publications

1. Thomas James Thomas, J. Sheeba Rani, Recovery from compressed measurements using Sparsity Independent Regularized Pursuit, Signal Processing, Volume 172, 2020, 107508, ISSN 0165-1684

2. T. J. Thomas and J. S. Rani, "FPGA Implementation of Sparsity Independent Regularized Pursuit for Fast CS Reconstruction," in IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 69, no. 4, pp. 1617-1628, April 2022, doi: 10.1109/TCSI.2021.3139164.

3. T. J. Thomas and J. S. Rani, "Gradient pursuit architecture for reduced complexity sparsity independent CS recovery," in IEEE Transactions on Circuits and Systems II: Express Briefs, 2022, doi: 10.1109/TCSII.2022.3223400.

4. Arun A., Thomas T.J., Rani J.S., Gorthi R.K.S.S. "Efficient directionality-driven dictionary learning for compressive sensing magnetic resonance imaging reconstruction", Journal of Medical Imaging, January 2020

Conference publications

1. Thomas T.J., Arun A. and Rani J.S. (2018) "CS recovery using modified Newton GP algorithm and application to ECG with Denoising", International Conference on Signal Processing and Communications (SPCOM) 2018, IISc Bangalore

2. Mubarak M., Thomas T.J., Rani J.S. and Mishra D. (2019) “Higher order Dictionary

Learning for Compressed Sensing based Dynamic MRI reconstruction", 30th British Machine Vision Conference (BMVC) 2019, Cardiff University, UK

3. Mubarak M., Thomas T.J., Rani J.S. and Mishra D. (2019) "Tensor Based Dictionary Learning for Compressive Sensing MRI Reconstruction", Fourth International Conference on Computer Vision and Image Processing (CVIP) 2019

4. Arun A., Thomas T.J., Rani J.S., Gorthi R.K.S.S. "Variable Patch Dictionaries for efficient compressed sensing based MRI Reconstruction", 11th Indian conference on Computer Vision Graphics and Image Processing, ICVGIP 2018

References

[1] H. Nyquist, "Certain topics in telegraph transmission theory," *Transactions of the American Institute of Electrical Engineers*, vol. 47, no. 2, pp. 617–644, 1928.

[2] C. Shannon, "Communication in the presence of noise," *Proceedings of the Institute of Radio Engineers*, vol. 37, no. 1, pp. 10–21, 1949.

[3] R. Walden, "Analog-to-digital converter survey and analysis," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 4, pp. 539–550, 1999.

[4] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM Journal on Scientific Computing*, vol. 20, no. 1, pp. 33–61, 2006.

[5] E. Candes and T. Tao, "Decoding by linear programming," *IEEE Transactions on Information Theory*, vol. 51, no. 12, pp. 4203–4215, 2005.

[6] E. J. Candes and T. Tao, "Near-optimal signal recovery from random projections: Universal encoding strategies?" *IEEE Transactions on Information Theory*, vol. 52, no. 12, pp. 5406–5425, 2006.

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

[8] M. B. Wakin *et al.*, "An architecture for compressive imaging," in *2006 International Conference on Image Processing*, 2006, pp. 1273–1276.

[9] M. Lustig, D. Donoho, and J. M. Pauly, "Sparse mri: The application of compressed sensing for rapid mr imaging," *Magnetic Resonance in Medicine*, vol. 58, no. 6, pp. 1182–1195, eprint: `https://onlinelibrary.wiley.com/doi/pdf/10.1002/mrm.21391`.

[10] K. T. B. D. K. S. R. O. H. C. L. Feng T. Benkert, "Compressed sensing for body mri," *Journal of Magnetic Resonance Imaging*, vol. 45, no. 4, pp. 966–987, 2017.

[11] M. A. Herman and T. Strohmer, "High-resolution radar via compressed sensing," *IEEE Transactions on Signal Processing*, vol. 57, no. 6, pp. 2275–2284, 2009.

[12] S.-H. Jung, Y.-S. Cho, R.-S. Park, J.-M. Kim, H.-K. Jung, and Y.-S. Chung, "High-resolution millimeter-wave ground-based sar imaging via compressed sensing," *IEEE Transactions on Magnetics*, vol. 54, no. 3, pp. 1–4, 2018.

[13] K. Wu, W. Cui, and X. Xu, "Superresolution radar imaging via peak search and compressed sensing," *IEEE Geoscience and Remote Sensing Letters*, vol. 19, pp. 1–5, 2022.

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

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

[16] R. Monika, S. Dhanalakshmi, R. Kumar, and R. Narayanamoorthi, "Coefficient permuted adaptive block compressed sensing for camera enabled underwater wireless sensor nodes," *IEEE Sensors Journal*, vol. 22, no. 1, pp. 776–784, 2022.

[17] J. Chen, S. Sun, L.-b. Zhang, B. Yang, and W. Wang, "Compressed sensing framework for heart sound acquisition in internet of medical things," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 3, pp. 2000–2009, 2022.

[18] A. M. R. Dixon, E. G. Allstot, D. Gangopadhyay, and D. J. Allstot, "Compressed sensing system considerations for ecg and emg wireless biosensors," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 6, no. 2, pp. 156–166, 2012.

[19] Z. Zhang, T.-P. Jung, S. Makeig, and B. D. Rao, "Compressed sensing of eeg for wireless telemonitoring with low energy consumption and inexpensive hardware," *IEEE Transactions on Biomedical Engineering*, vol. 60, no. 1, pp. 221–224, 2013.

[20] F. Ren and D. Marković, "18.5 a configurable 12-to-237ks/s 12.8mw sparse-approximation engine for mobile exg data aggregation," in *2015 IEEE International Solid-State Circuits Conference - (ISSCC) Digest of Technical Papers*, 2015, pp. 1–3.

[21] T. R. Braun, "An evaluation of GPU acceleration for sparse reconstruction," in *Signal Processing, Sensor Fusion, and Target Recognition XIX*, I. Kadar, Ed., International Society for Optics and Photonics, vol. 7697, SPIE, 2010, pp. 430 –439.

[22] D. Gangopadhyay, E. G. Allstot, A. M. R. Dixon, K. Natarajan, S. Gupta, and D. J. Allstot, "Compressed sensing analog front-end for bio-sensor applications," *IEEE Journal of Solid-State Circuits*, vol. 49, no. 2, pp. 426–438, 2014.

[23] J. Zhang, Y. Yan, L. J. Chen, M. Wang, T. Moscibroda, and Z. Zhang, "Impression store: Compressive sensing-based storage for big data analytics," in *Proceedings of the 6th USENIX Conference on Hot Topics in Cloud Computing*, ser. HotCloud'14, Philadelphia, PA: USENIX Association, 2014, p. 1.

[24] D. E. Bellasi, L. Bettini, C. Benkeser, T. Burger, Q. Huang, and C. Studer, "Vlsi design of a monolithic compressive-sensing wideband analog-to-information converter," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 3, no. 4, pp. 552–565, 2013.

[25] M. Fornasier and H. Rauhut, "Compressive sensing," in *Handbook of mathematical methods in imaging*, Springer, 2011, pp. 187–228.

[26] E. J. Candès, J. K. Romberg, and T. Tao, "Stable signal recovery from incomplete and inaccurate measurements," *Communications on Pure and Applied Mathematics*, vol. 59, no. 8, pp. 1207–1223, 2006. eprint: `https://onlinelibrary.wiley.com/doi/pdf/10.1002/cpa.20124`.

[27] H. Rauhut, K. Schnass, and P. Vandergheynst, "Compressed sensing and redundant dictionaries," *IEEE Transactions on Information Theory*, vol. 54, no. 5, pp. 2210–2219, 2008.

[28] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[29] E. Hale, W. Yin, and Y. Zhang, "A fixed-point continuation method for '1-regularized minimization with applications to compressed sensing," *CAAM Technical Report TR07-07*, Jan. 2007.

[30] S. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries," *IEEE Transactions on Signal Processing*, vol. 41, no. 12, pp. 3397–3415, Dec. 1993.

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

[32] T. Blumensath and M. E. Davies, "Stagewise weak gradient pursuits," *IEEE Transactions on Signal Processing*, vol. 57, no. 11, pp. 4333–4346, 2009.

[33] D. Needell and R. Vershynin, "Uniform uncertainty principle and signal recovery via regularized orthogonal matching pursuit," *Foundations of computational mathematics*, vol. 9, no. 3, pp. 317–334, 2009.

[34] D. Needell and J. Tropp, "Cosamp: Iterative signal recovery from incomplete and inaccurate samples," *Applied and Computational Harmonic Analysis*, vol. 26, no. 3, pp. 301 –321, 2009.

[35] W. Dai and O. Milenkovic, "Subspace pursuit for compressive sensing signal reconstruction," *IEEE Transactions on Information Theory*, vol. 55, no. 5, pp. 2230–2249, 2009.

[36] H. Huang and A. Makur, "Backtracking-based matching pursuit method for sparse signal reconstruction," *IEEE Signal Processing Letters*, vol. 18, no. 7, pp. 391–394, 2011.

[37] T. T. Do, L. Gan, N. Nguyen, and T. D. Tran, "Sparsity matching pursuit algorithm for practical compressed sensing," in *2008 42nd Asilomar Conference on Signals, Systems and Computers*, 2008, pp. 581–587.

[38] X. Li, Y. Liu, S. Zhao, and W. Chu, "A modified regularized adaptive matching pursuit algorithm for linear frequency modulated signal detection based on compressive sensing," *Journal of Communications*, vol. 11, no. 4, 2016.

[39] T. Blumensath and M. E. Davies, "Iterative hard thresholding for compressed sensing," *Applied and computational harmonic analysis*, vol. 27, no. 3, pp. 265–274, 2009.

[40] D. L. Donoho, A. Maleki, and A. Montanari, "Message-passing algorithms for compressed sensing," *Proceedings of the National Academy of Sciences*, vol. 106, no. 45, pp. 18 914–18 919, 2009.

[41] S. Ji, Y. Xue, and L. Carin, "Bayesian compressive sensing," *IEEE Transactions on Signal Processing*, vol. 56, no. 6, pp. 2346–2356, 2008.

[42] S. Jafarpour, W. Xu, B. Hassibi, and R. Calderbank, "Efficient and robust compressed sensing using optimized expander graphs," *IEEE Transactions on Information Theory*, vol. 55, no. 9, pp. 4299–4308, 2009.

[43] M. Lotfi and M. Vidyasagar, "A fast noniterative algorithm for compressive sensing using binary measurement matrices," *IEEE Transactions on Signal Processing*, vol. 66, no. 15, pp. 4079–4089, 2018.

[44] M. Raginsky, S. Jafarpour, Z. T. Harmany, R. F. Marcia, R. M. Willett, and R. Calderbank, "Performance bounds for expander-based compressed sensing in poisson noise," *IEEE Transactions on Signal Processing*, vol. 59, no. 9, pp. 4139–4153, 2011.

[45] Y.-M. Lin, Y. Chen, N.-S. Huang, and A.-Y. Wu, "Low-complexity stochastic gradient pursuit algorithm and architecture for robust compressive sensing reconstruction," *IEEE Transactions on Signal Processing*, vol. 65, no. 3, pp. 638–650, 2017.

[46] D. Takhar *et al.*, "A new compressive imaging camera architecture using optical-domain compression," in *Computational Imaging IV*, C. A. Bouman, E. L. Miller, and I. Pollak, Eds., International Society for Optics and Photonics, vol. 6065, SPIE, 2006, pp. 43 –52.

[47] J. A. Tropp, J. N. Laska, M. F. Duarte, J. K. Romberg, and R. G. Baraniuk, "Beyond nyquist: Efficient sampling of sparse bandlimited signals," *IEEE Transactions on Information Theory*, vol. 56, no. 1, pp. 520–544, 2010.

[48] H. Mamaghanian, N. Khaled, D. Atienza, and P. Vandergheynst, "Design and exploration of low-power analog to information conversion based on compressed sensing," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 2, no. 3, pp. 493–501, 2012.

[49] R. Baraniuk and P. Steeghs, "Compressive radar imaging," in *2007 IEEE Radar Conference*, 2007, pp. 128–133.

[50] J. H. Ender, "On compressive sensing applied to radar," *Signal Processing*, vol. 90, no. 5, pp. 1402–1414, 2010, Special Section on Statistical Signal and Array Processing.

[51] A. Kulkarni and T. Mohsenin, "Accelerating compressive sensing reconstruction omp algorithm with cpu, gpu, fpga and domain specific many-core," in *2015 IEEE ISCAS*, 2015, pp. 970–973.

[52] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[53] B. Benson, A. Irturk, J. Cho, and R. Kastner, "Survey of hardware platforms for an energy efficient implementation of matching pursuits algorithm for shallow water networks," in *Proceedings of the Third ACM International Workshop on Underwater Networks*, ser. WuWNeT '08, San Francisco, California, USA: Association for Computing Machinery, 2008, 83–86, ISBN: 9781605581859.

[54] A. Septimus and R. Steinberg, "Compressive sampling hardware reconstruction," in *Proceedings of 2010 IEEE ISCAS*, IEEE, 2010, pp. 3316–3319.

[55] J. L. V. M. Stanislaus and T. Mohsenin, "High performance compressive sensing reconstruction hardware with qrd process," in *2012 IEEE ISCAS*, 2012, pp. 29–32.

[56] J. Stanislaus and T. Mohsenin, "Low-complexity fpga implementation of compressive sensing reconstruction," in *2013 International Conference on Computing, Networking and Communications (ICNC)*, 2013, pp. 671–675.

[57] L. Bai, P. Maechler, M. Muehlberghuber, and H. Kaeslin, "High-speed compressed sensing reconstruction on fpga using omp and amp," in *2012 19th IEEE ICECS*, 2012, pp. 53–56.

[58] P. Blache, H. Rabah, and A. Amira, "High level prototyping and fpga implementation of the orthogonal matching pursuit algorithm," in *2012 11th International Con-*

ference on Information Science, Signal Processing and their Applications (ISSPA), 2012, pp. 1336–1340.

[59] H. Rabah, A. Amira, B. K. Mohanty, S. Almaadeed, and P. K. Meher, "Fpga implementation of orthogonal matching pursuit for compressive sensing reconstruction," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 23, no. 10, pp. 2209–2220, 2015.

[60] F. Ren, R. Dorrace, W. Xu, and D. Marković, "A single-precision compressive sensing signal reconstruction engine on fpgas," in *2013 23rd International Conference on Field programmable Logic and Applications*, 2013, pp. 1–4.

[61] B. Knoop, J. Rust, S. Schmale, D. Peters-Drolshagen, and S. Paul, "Rapid digital architecture design of orthogonal matching pursuit," in *2016 24th European Signal Processing Conference (EUSIPCO)*, 2016, pp. 1857–1861.

[62] Y. Cheng, P. Tsai, and M. Huang, "Matrix-inversion-free compressed sensing with variable orthogonal multi-matching pursuit based on prior information for ecg signals," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 10, no. 4, pp. 864–873, 2016.

[63] F. Ren and D. Marković, "A configurable 12–237 ks/s 12.8 mw sparse-approximation engine for mobile data aggregation of compressively sampled physiological signals," *IEEE Journal of Solid-State Circuits*, vol. 51, no. 1, pp. 68–78, 2016.

[64] G. Huang and L. Wang, "An fpga-based architecture for high-speed compressed signal reconstruction," *ACM Trans. Embed. Comput. Syst.*, vol. 16, no. 3, 89:1–89:23, May 2017.

[65] A. Kulkarni and T. Mohsenin, "Low overhead architectures for omp compressive sensing reconstruction algorithm," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 64, no. 6, pp. 1468–1480, 2017.

[66] M. Fardad, S. M. Sayedi, and E. Yazdian, "A low-complexity hardware for deterministic compressive sensing reconstruction," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 10, pp. 3349–3361, 2018.

[67] S. Liu, N. Lyu, and H. Wang, "The implementation of the improved omp for aic reconstruction based on parallel index selection," *IEEE Transactions on VLSI Systems*, vol. 26, no. 2, pp. 319–328, 2018.

[68] S. Roy, D. P. Acharya, and A. K. Sahoo, "Low-complexity architecture of orthogonal matching pursuit based on qr decomposition," *IEEE Transactions on VLSI Systems*, vol. 27, no. 7, pp. 1623–1632, 2019.

[69] X. Ge, F. Yang, H. Zhu, X. Zeng, and D. Zhou, "An efficient fpga implementation of orthogonal matching pursuit with square-root-free qr decomposition," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 3, pp. 611–623, 2019.

[70] J. Li, P. Chow, Y. Peng, and T. Jiang, "Fpga implementation of an improved omp for compressive sensing reconstruction," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, pp. 1–14, 2020.

[71] T. Chen, H. Kuo, and A. Wu, "A 232–1996-ks/s robust compressive sensing reconstruction engine for real-time physiological signals monitoring," *IEEE Journal of Solid-State Circuits*, vol. 54, no. 1, pp. 307–317, 2019.

[72] Y.-Z. Wang, Y.-P. Wang, Y.-C. Wu, and C.-H. Yang, "A 12.6 mw, 573–2901 ks/s reconfigurable processor for reconstruction of compressively sensed physiological signals," *IEEE Journal of Solid-State Circuits*, vol. 54, no. 10, pp. 2907–2916, 2019.

[73] Y. Liu, T. Song, and Y. Zhuang, "A high-throughput subspace pursuit processor for ecg recovery in compressed sensing using square-root-free mgs qr decomposition," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 28, no. 1, pp. 174–187, 2020.

[74] O. Polat and S. K. Kayhan, "High-speed fpga implementation of orthogonal matching pursuit for compressive sensing signal reconstruction," *Computers and Electrical Engineering*, vol. 71, pp. 173–190, 2018.

[75] L. Bai, P. Maechler, M. Muehlberghuber, and H. Kaeslin, "High-speed compressed sensing reconstruction on fpga using omp and amp," in *2012 19th IEEE International Conference on Electronics, Circuits, and Systems (ICECS 2012)*, 2012, pp. 53–56.

[76] B. Knoop, J. Rust, S. Schmale, D. Peters-Drolshagen, and S. Paul, "Rapid digital architecture design of orthogonal matching pursuit," in *2016 24th European Signal Processing Conference (EUSIPCO)*, 2016, pp. 1857–1861.

[77] S. Kunis and H. Rauhut, "Random sampling of sparse trigonometric polynomials: Orthogonal matching pursuit versus basis pursuit," *Found. Comput. Math.*, vol. 8, no. 6, 737–763, 2008.

[78] S. K. Sahoo and A. Makur, "Signal recovery from random measurements via extended orthogonal matching pursuit," *IEEE Transactions on Signal Processing*, vol. 63, no. 10, pp. 2572–2581, 2015.

[79] T. Blumensath and M. E. Davies, "Stagewise weak gradient pursuits," *IEEE Transactions on Signal Processing*, vol. 57, no. 11, pp. 4333–4346, 2009.

[80] D. Donoho and J. Tanner, "Observed universality of phase transitions in high-dimensional geometry, with implications for modern data analysis and signal processing," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 367, 2009.

[81] T. J. Thomas and J. Sheeba Rani, "Recovery from compressed measurements using sparsity independent regularized pursuit," *Signal Processing*, vol. 172, p. 107 508, 2020.

[82] G. Moody and R. Mark, "The impact of the mit-bih arrhythmia database," *IEEE Eng in Med and Biol*, vol. 20, no. 3, pp. 45–50, 2001.

[83] J. Hormigo, M. Ortiz, F. Quiles, F. J. Jaime, J. Villalba, and E. L. Zapata, "Efficient implementation of carry-save adders in fpgas," in *2009 20th IEEE International Conference on Application-specific Systems, Architectures and Processors*, 2009, pp. 207–210.

[84] Wikipedia contributors, *Fast inverse square root — Wikipedia, the free encyclopedia*, [Online; accessed 19-July-2022], 2022.

[85] M. Jacobsen, Y. Freund, and R. Kastner, "Riffa: A reusable integration framework for fpga accelerators," vol. 8, Apr. 2012, pp. 216–219, ISBN: 978-1-4673-1605-7.

[86] J. Jin, Y. Gu, and S. Mei, "A stochastic gradient approach on compressive sensing signal reconstruction based on adaptive filtering framework," *IEEE Journal of Selected Topics in Signal Processing*, vol. 4, no. 2, pp. 409–420, 2010.

[87] M. Wakin *et al.*, "A nonuniform sampler for wideband spectrally-sparse environments," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 2, no. 3, pp. 516–529, 2012.

List of Publications

Journal publications

1. Thomas James Thomas, J. Sheeba Rani, Recovery from compressed measurements using Sparsity Independent Regularized Pursuit, Signal Processing, Volume 172, 2020, 107508, ISSN 0165-1684

2. T. J. Thomas and J. S. Rani, "FPGA Implementation of Sparsity Independent Regularized Pursuit for Fast CS Reconstruction," in IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 69, no. 4, pp. 1617-1628, April 2022, doi: 10.1109/TCSI.2021.3139164.

3. T. J. Thomas and J. S. Rani, "Gradient pursuit architecture for reduced complexity sparsity independent CS recovery," in IEEE Transactions on Circuits and Systems II: Express Briefs, 2022, doi: 10.1109/TCSII.2022.3223400.

4. Arun A., Thomas T.J., Rani J.S., Gorthi R.K.S.S. "Efficient directionality-driven dictionary learning for compressive sensing magnetic resonance imaging reconstruction", Journal of Medical Imaging, January 2020

Conference publications

1. Thomas T.J., Arun A. and Rani J.S. (2018) "CS recovery using modified Newton GP algorithm and application to ECG with Denoising", International Conference on Signal Processing and Communications (SPCOM) 2018, IISc Bangalore

2. Mubarak M., Thomas T.J., Rani J.S. and Mishra D. (2019) “Higher order Dictionary

Learning for Compressed Sensing based Dynamic MRI reconstruction", 30th British Machine Vision Conference (BMVC) 2019, Cardiff University, UK

3. Mubarak M., Thomas T.J., Rani J.S. and Mishra D. (2019) "Tensor Based Dictionary Learning for Compressive Sensing MRI Reconstruction", Fourth International Conference on Computer Vision and Image Processing (CVIP) 2019

4. Arun A., Thomas T.J., Rani J.S., Gorthi R.K.S.S. "Variable Patch Dictionaries for efficient compressed sensing based MRI Reconstruction", 11th Indian conference on Computer Vision Graphics and Image Processing, ICVGIP 2018