5-25-2016

Design of Hardware-Efficient Signal Reconstruction for Embedded Computing

Guoxian Huang
guoxian.huang@uconn.edu

Follow this and additional works at: http://digitalcommons.uconn.edu/dissertations

Recommended Citation
http://digitalcommons.uconn.edu/dissertations/1138
Design of Hardware-Efficient Signal Reconstruction for Embedded Computing

Guoxian Huang, Ph.D.
University of Connecticut, 2016

Conventional sensing techniques often acquire the signals entirely using a lot of resources and then just toss away a large portion of the obtained data during compression. This motivates an emerging research area called as Compressive Sensing (CS) that allows efficient signal acquisition under the sub-Nyquist rate while still able to promise reliable data recovery. Despite the benefits of compressive sensing, one critical issue in the practical applications of compressive sensing is how to reliably recover the original signals from only a few measurements in an efficient way. The Orthogonal Matching Pursuit (OMP) algorithm has shown a good capability for reliable recovery of compressed signals. Due to the simple geometric interpolation and good efficiency, the OMP-based greedy algorithms are often the preferable choice in hardware implementations for real-time signal recovery. However, practical applications of compressive sensing in hardware platforms are limited as signal reconstruction is still challenging due to its high computational complexity.

On the greedy algorithms for compressive signal reconstruction, we will first investigate the computation steps of the Orthogonal Matching Pursuit algorithm.
In the iterative computations, intermediate signal estimates and matrix inversions can be decoupled, thereby enabling parallel processing of these two time-consuming operations in the Orthogonal Matching Pursuit algorithm. Based on the observation, the implementation technique of an algorithmic transformation technique referred to as Matrix Inversion Bypass (MIB) is proposed to improve the signal recovery efficiency of the Orthogonal Matching Pursuit based CS reconstruction. The proposed MIB naturally leads to a parallel architecture for high-speed dedicated hardware implementations and the hardware implementations will be studied for verification.

For the OMP-based signal recovery, we find out that more significant elements of the signal are likely to be recovered first in the iterative OMP algorithm. On the other hand, as iteration order goes up, the OMP algorithm still suffers from significantly increasing computational complexity despite relatively low complexity of hardware implementation. Based on this, a Soft-thresholding Orthogonal Matching Pursuit (ST-OMP) technique is proposed for efficient signal reconstruction in compressive sensing applications. The proposed ST-OMP recovers less significant signal elements using a low-complexity procedure without much degradation in reconstruction quality. The proposed ST-OMP is applied in systems powered by non-deterministic renewable energy sources. The threshold of employing the efficient reconstruction is made dynamically adjustable according to the performance requirements and energy levels. Simulation results demonstrate that the
ST-OMP can achieve good recovery performance while significantly reducing the energy consumption as compared to the original OMP implementation.
Design of Hardware-Efficient Signal Reconstruction for Embedded Computing

Guoxian Huang

B.S. Huazhong University of Science and Technology, 2009
M.S. Huazhong University of Science and Technology, 2012

A Dissertation
Submitted in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy at the University of Connecticut

2016
Copyright by

Guoxian Huang

2016
APPROVAL PAGE

Doctor of Philosophy Dissertation

Design of Hardware-Efficient Signal Reconstruction for Embedded Computing

Presented by

Guoxian Huang, B.S., M.S.

Major Advisor

----------------------------------
Lei Wang

Associate Advisor

----------------------------------
John Chandy

Associate Advisor

----------------------------------
Shalabh Gupta

University of Connecticut

2016
To my family
ACKNOWLEDGEMENTS

First and foremost, I would like to express my sincere gratitude to my advisor, Professor Lei Wang for his supervision, patience and support during my Ph.D study. Without his great help, none of the progresses I made would not have been possible. His broad knowledge and sharp insight always guide me in the right direction. I also would like to thank Professor John Chandy, Shalabh Gupta, Faquir Jain and Liang Zhang for their valuable time and useful advices.

Secondly, I want to thank my fellow lab mates Junlin Chen, Ridvan Umaz, Dong Zhao, Menglong Guan, Wenjie Huang, Shuai Chen, Fengyu Qian and Yanping Gong for the insightful discussions, knowledge and assistance.

Finally, I must thank my parents for their constant love, support and encouragement. They have always been there for me.
## Acronyms and Notations

**Acronym**

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIC</td>
<td>analog-to-information converters</td>
</tr>
<tr>
<td>CS</td>
<td>Compressive Sensing</td>
</tr>
<tr>
<td>ADC</td>
<td>analog-to-digital converters</td>
</tr>
<tr>
<td>MRI</td>
<td>magnetic resonance imaging</td>
</tr>
<tr>
<td>LP</td>
<td>linear programming</td>
</tr>
<tr>
<td>BP</td>
<td>basis pursuit</td>
</tr>
<tr>
<td>OMP</td>
<td>orthogonal matching pursuit</td>
</tr>
<tr>
<td>MIB</td>
<td>matrix inversion bypass</td>
</tr>
<tr>
<td>RIP</td>
<td>restricted isometry property</td>
</tr>
<tr>
<td>RIC</td>
<td>restricted isometry constant</td>
</tr>
<tr>
<td>DMD</td>
<td>digital micromirror device</td>
</tr>
<tr>
<td>CD</td>
<td>Cholesky Decomposition</td>
</tr>
<tr>
<td>ACD</td>
<td>Alternative Cholesky Decomposition</td>
</tr>
<tr>
<td>PSNR</td>
<td>peak signal-to-noise ratio</td>
</tr>
<tr>
<td>Acronym</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>PEs</td>
<td>Processing Elements</td>
</tr>
<tr>
<td>RMSE</td>
<td>root mean square errors</td>
</tr>
<tr>
<td>DCT</td>
<td>discrete cosine transform</td>
</tr>
<tr>
<td>DFT</td>
<td>discrete fourier transform</td>
</tr>
<tr>
<td>MAC</td>
<td>multiply-accumulate</td>
</tr>
<tr>
<td>RF</td>
<td>radio frequency</td>
</tr>
<tr>
<td>DWT</td>
<td>discrete wavelet transform</td>
</tr>
<tr>
<td>SER</td>
<td>signal-to-error ratio</td>
</tr>
<tr>
<td>i.i.d.</td>
<td>identically and independently distributed</td>
</tr>
</tbody>
</table>
Notation

\((\cdot)^*\) \hspace{1cm} \text{conjugate transpose}

\(|\cdot|\) \hspace{1cm} \text{absolute value}

\(\|\cdot\|_n\) \hspace{1cm} \text{n-th norm}

\([\cdot]\) \hspace{1cm} \text{round towards minus infinity}
# TABLE OF CONTENTS

Chapter 1: Introduction 1

1.1 Overview ................................................. 1
1.2 Dissertation Contribution ............................ 4
1.3 Outline of the Dissertation ............................ 5

Chapter 2: The Background 7

2.1 Preliminaries of Compressive Sensing ................... 7
2.2 Sparse Signal Recovery ................................. 11
   2.2.1 Basis Pursuit ........................................... 11
   2.2.2 Greedy Recovery Algorithm ......................... 13
2.3 Applications .............................................. 14
   2.3.1 Compressive Imaging ................................. 14
   2.3.2 Analog-to-information Converters .................... 16
   2.3.3 Compressive Radar ..................................... 16

Chapter 3: The Matrix Inversion Bypass (MIB) Transform and Implementation 18

3.1 Introduction ............................................... 18
3.2 OMP algorithm ............................................ 21
3.3 The proposed Matrix Inversion Bypass (MIB) Transform 23
   3.3.1 Matrix Inversion Bypass .............................. 24
3.3.2 Complexity Comparison ............................................. 28

3.4 Architectural Design for Hardware Specialization ................. 31
  3.4.1 System Architecture ............................................. 31
  3.4.2 Data Structure ....................................................... 33
  3.4.3 Initialize $\Phi^T Y$ ................................................. 34
  3.4.4 Update $\alpha_k$ ..................................................... 35
  3.4.5 Update $I_k$ ........................................................ 37
  3.4.6 Update $G_{k,k}^{-1}$ : ............................................. 37

3.5 Performance Evaluation ................................................ 39
  3.5.1 Evaluation of MIB Transform .................................... 39
  3.5.2 Case study on Video Monitoring ................................. 42

3.6 Summary ................................................................. 47

Chapter 4: FPGA Architecture and Implementation of the MIB
  transform ................................................................. 49
  4.1 Overview ................................................................. 49
  4.2 FPGA Architecture and Implementation ............................. 51
    4.2.1 Architecture Design ............................................. 51
    4.2.2 Finding Index Unit ............................................. 54
    4.2.3 Bypassing Unit .................................................... 60
    4.2.4 Data Unit .......................................................... 62
  4.3 Results and Analysis ................................................. 65
4.3.1 Complexity Comparison ........................................... 66
4.3.2 Performance Evaluation ........................................... 68
4.3.3 Comparisons With the Existing Work ......................... 72
4.3.4 Case Study: Audio Signal Processing ......................... 74
4.4 Summary .................................................................. 76

Chapter 5:  Soft-thresholding OMP Algorithm and Implementation

5.1 Overview ................................................................... 77
5.2 The proposed Soft-thresholding OMP ......................... 78
  5.2.1 Motivation ......................................................... 78
  5.2.2 Soft-thresholding OMP ....................................... 85
  5.2.3 Convergence study of the Soft-thresholding OMP ....... 88
  5.2.4 Recovery performance of the Soft-thresholding OMP ... 93
  5.2.5 Implementation Architecture of the Soft-thresholding OMP 95
5.3 Evaluation ............................................................... 97
  5.3.1 Numerical Results .............................................. 97
  5.3.2 Case Study: DFT-Domain Signal Reconstruction ....... 103
5.4 Summary .................................................................. 108

Chapter 6:  Conclusions and Future Work

6.1 Conclusions ............................................................ 109
6.2 Future Work ............................................................ 110
# LIST OF TABLES

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The number of arithmetic operations needed to update $\hat{X}_k$</td>
<td>31</td>
</tr>
<tr>
<td>2</td>
<td>Signal parameters</td>
<td>39</td>
</tr>
<tr>
<td>3</td>
<td>Hardware-related measurements of the proposed MIB transform and Batch-OMP.</td>
<td>42</td>
</tr>
<tr>
<td>4</td>
<td>Stages of the finding index operation</td>
<td>54</td>
</tr>
<tr>
<td>5</td>
<td>Stages of the bypassing operation</td>
<td>62</td>
</tr>
<tr>
<td>6</td>
<td>FPGA resource utilization in the proposed MIB architecture</td>
<td>65</td>
</tr>
<tr>
<td>7</td>
<td>Arithmetic operations to update $\alpha_k$ in the $k^{th}$ iteration</td>
<td>67</td>
</tr>
<tr>
<td>8</td>
<td>Arithmetic operations of the MIB technique to update $\alpha_k$ in the</td>
<td>68</td>
</tr>
<tr>
<td></td>
<td>$k^{th}$ iteration</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Comparison Implementation Results</td>
<td>74</td>
</tr>
<tr>
<td>10</td>
<td>The Major Computations of the $k^{th}$ iteration of OMP algorithm</td>
<td>84</td>
</tr>
<tr>
<td>11</td>
<td>Hardware measurements of the proposed ST-OMP.</td>
<td>101</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

1 Flow chart of the MIB-based signal reconstruction algorithm. 28
2 Operation flow of (a) Batch-OMP algorithm and (b) the proposed MIB transform. 30
3 Top-level architecture of the proposed MIB transform. 32
4 Update $\Phi^T Y$ and $\alpha_0$. 34
5 Update $\alpha_k$. 35
6 Update $I_k$. 37
7 Update $G_{I_k}^{-1} I_k$. 38
8 Comparison of the number of clock cycles during the iteration of the proposed MIB transform and Batch-OMP. 40
9 Comparison of the number of clock cycles in the MIB transform for (a) updating $\alpha_k$ and $I_k$, (b) updating $[VMA^T, -VM]$, and (c) updating $G_{I_k}^{-1} I_k$. 41
10 System diagram of video processing using compressive sensing. 42
11 PSNR (in dB) and sparsity for a 12-hour video. 44
12 Performance of video monitoring. 45
13 Energy consumption of the two implementations. 46
14 Failure rate of the two implementations. 47
15 Top-level architecture of the proposed MIB signal reconstruction. 52
16 Control flow of the proposed MIB architecture. 53
17 Finding index unit ............................................. 55
18 Collective accumulation mode .......................... 56
19 Independent accumulation mode ...................... 57
20 Scheme to find the index with the largest value .... 60
21 Bypassing operation ........................................... 62
22 Memory data arrangement for the matrix $G$: (a) original arrangement and (b) improved arrangement facilitating single read access of all the components .................. 63
23 Complexity of updating $\alpha$ under the signal length $N = 256$ and $512$. 69
24 Timing breakdown of the finding index and bypassing operations. 70
25 Speedup ratio of the proposed MIB architecture over the OMP implementation ........................................ 71
26 The SNR of the recovered signals under different measurement ratios 72
27 The original and recovered audio signals under the measurement ratio of 0.5 ......................................... 73
28 Statistical results of signal recovery pattern ...................... 79
29 Implementation architecture of the proposed ST-OMP algorithm 98
30 Recovery accuracy for flat signals with different sparsities and threshold $L$ ......................................... 99
31 Grey image reconstruction via the proposed ST-OMP 100
32 Performance of the ST-OMP in a self-powered sensing system (solar cell panel area: $1.5\text{cm}^2$) ........................................ 102
33 Success rate of the detected DFT-frequencies by ST-OMP algorithm being closest to the occupied frequencies under different estimated sparsity $\hat{K}$ and list length $L$.

34 Success rate of the detected DFT-frequencies by ST-OMP algorithm being in the sensitive area under different estimated sparsity $\hat{K}$ and list length $L$.

35 Accuracy of recovering the closest DFT-frequencies and targeting the sensitive area under different variance.

36 DFT of original signal and estimated signal by ST-OMP.
Chapter 1

Introduction

1.1 Overview

Many embedded signal processing systems need to achieve real-time performance under strict physical and operational constraints. For example, distributed sensor networks need to acquire and transmit a large amount of data subject to a wide range of physical conditions such as coverage range, detection sensitivity, system robustness, and timing requirements [1, 2]. To intelligently correlate the computing process with physical resources, one effective way is to reduce the unnecessary or redundant data to be processed. Traditional sensing techniques often obtain far more data than necessary, but only a small portion of the acquired data are actually useful while tossing away the rest will not affect the performance in most cases.
This gives rise to a new research field referred to as *Compressive Sensing* (CS) [3, 4, 5, 6], which allows more efficient signal acquisition with fewer measurements or at the sub-Nyquist rate [7, 8]. Thus, the physical resources needed for data acquisition, computation, and transmission can be significantly reduced. Some recent applications include compressive imaging [9, 10, 11], which applied the theory of compressive sensing to attain sub-Nyquist image acquisition; far fewer measurements were performed than the conventional imaging while reliable image reconstruction was still possible. Medical imaging [12, 13, 14] was also benefited from compressive sensing with reduced magnetic resonance imaging (MRI) acquisition time. As an alternative to conventional analog-to-digital converters (ADC), compressive sensing based analog-to-information converters (AIC) [15, 16, 17, 18] focus only on the relevant information, which enables a great reduction of the digital data rate. In wireless communication systems, compressive sensing is also used in sparse channel estimation [19, 20, 21]. The radar imaging system can also see great potential of better resolution over classical radar [22] by using the tools of compressive sensing.

Despite the benefits of compressive sensing brought to those systems, practical applications have to deal with the critical issue of recovering the original signal from the fewer number of measurements in a reliable and efficient way. For this purpose, a lot of research efforts have been dedicated to the signal reconstruction algorithms [23, 24, 25, 26, 27, 28, 29]. Two major existing approaches are the
\( \ell_1 \)-minimization and greedy algorithms. The \( \ell_1 \)-minimization relaxes from the \( \ell_0 \)-minimization applying the convex optimization to reconstruct the signal and can be executed based on the Linear Programming (LP) in polynomial time [23]. The LP algorithms, also known as Basis Pursuit (BP), involve high complexity even though some fast algorithms have been proposed [25]. The other well-studied approach is based on the iterative greedy pursuit. The family of greedy algorithms includes the Orthogonal Matching Pursuit (OMP) [26], Regularized OMP (ROMP) [27], Stagewise OMP (StOMP) [28] and Subspace Pursuit (SP) [29]. The Orthogonal Matching Pursuit (OMP) algorithm [26] has shown a good capability for reliable recovery of compressed signals. Due to the simple geometric interpolation and good efficiency, the OMP-based greedy algorithms [26, 27, 28, 29] are often the preferable choice in hardware implementations [30, 31] for compressive sensing signal recovery.

While significant progress has been made in optimizing the hardware implementation of compressed signal reconstruction, the computational complexity still remains high [32]. This dissertation will focus on reliable and efficient signal reconstruction implementation based on Orthogonal Matching Pursuit (OMP) algorithm from the fewer measurements in the embedded signal processing systems.
1.2 Dissertation Contribution

The major contributions in this dissertation are as follows,

- Firstly, this dissertation studies the implementation techniques of the Orthogonal Matching Pursuit algorithm and proposes an algorithmic transformation technique referred to as Matrix Inversion Bypass (MIB) [33, 34]. The basic idea of MIB is to decouple the computations of intermediate signal estimates and matrix inversions, thereby enabling parallel processing of these two time-consuming operations in the OMP algorithm.

- This dissertation also develops an FPGA architecture based on MIB to eliminate the speed bottleneck with parallel computing and more efficient utilization of hardware resources [35]. In the architecture design, the hardware resources are dynamically allocated for high-speed computation, especially the dominant matrix-vector multiplications with varying sizes at the different stages of the iteration and the different iterations during the iterative signal recovery. The FPGA block memory overhead is also reduced using a new data storage structure.

- This dissertation observes that the last rounds of the iterative OMP signal recovery are of higher computational complexity involving matrix operations of larger size and introduce more energy overhead. For applications that are operated under severe energy constraints, this may lead to failure of the whole signal recovery or energy-insufficiency for other
tasks considering the varying domain-specific constraints for compressive signal recovery. This dissertation proposes *Soft-thresholding OMP* technique (ST-OMP) [36, 37] that allows efficient recovery of less significant but computation-intensive signal components while maintaining high reconstruction quality.

### 1.3 Outline of the Dissertation

This dissertation is organized as follows,

- Chapter 2 briefly reviews the theory of compressive sensing, the algorithmic approaches to solve the sparse signal recovery problem and some applications that can be benefitted using the under-determined systems provided by compressive sensing.

- Chapter 3 covers the proposed algorithm transformation technique referred to as *Matrix Inversion Bypass* (MIB) to improve the reconstruction time and reduce the computational complexity of OMP algorithm by decoupling two timing-critical operations.

- Chapter 4 presents the FPGA implementation of MIB technique to eliminate the speed bottleneck with parallel computing and more efficient utilization of hardware resources.
• Chapter 5 covers the proposed *Soft-thresholding Orthogonal Matching Pursuit* (ST-OMP) technique for efficient signal reconstruction in compressive sensing applications.

• Chapter 6 concludes the dissertation.
Chapter 2

The Background

In this chapter, preliminaries of compressive sensing theory will first be briefly reviewed. Two major algorithmic approaches to solving the sparse signal recovery problem are then explained. The first approach is based on an optimization problem solvable using linear programming while the second approach is developed from the greedy algorithms. At last, some exemplary applications of compressive sensing are presented.

2.1 Preliminaries of Compressive Sensing

Compressive sensing can be expressed mathematically by multiplying the original signal $X$ of length $N$ with a sensing matrix $\Phi \in R^{M \times N}$ to obtain fewer measurements $Y$, i.e.,

$$Y = \Phi \ast X + e,$$  \hspace{1cm} (1)
where \( M \ll N \) such that the system is under-determined and infinite number of feasible solutions exist; \( e \in \mathbb{R}^M \) is additive measurement noise. To reliably recover the original signal, the CS theory relies first and foremost on the sparsity of signal \( X \), of which most coefficients need to be zero or relatively insignificant in some basis such that they can be tossed away without much loss.

Many natural or man-made signals \( X \) either are sparse in the time domain or are known to be sparse in some basis, and it enables reliable recovery of \( X \) from fewer measurements \( Y \). The representation of \( \Psi \)-basis sparse signal \( X \) in the sparsity basis \( \Psi = [\psi_1 \, \psi_2 \, \ldots \, \psi_N] \) is,

\[
X = \sum_{i=1}^{N} f_i \psi_i \quad \text{or} \quad X = \Psi f
\]  

(2)

where a large fraction of the coefficient set \( f \) can be thrown away without much loss. For example, most wavelet coefficients of images in a wavelet basis are small while the rest few are large capturing most of the information and only keeping the few large coefficients will not cause much perceptual loss. The number of those non-zero or significant entries in the time domain or \( \Psi \)-basis is denoted as \( K \) in this dissertation.

In addition, the sparsity of signal \( X \) can also be quantified by finding the best sparse approximation to the signal \( X \) [38]. The optimal \( K \)-sparse approximation is defined as achieving the lowest distortion of the signal \( X \) by selecting \( K \) coefficients:

\[
\delta_\Psi(X, K) = \min_{\|Z\|_0 \leq K} \|X - \Psi Z\|_2.
\]  

(3)
Consider the coefficients $f$ of signal $X$, when sorted in order of decreasing magnitude, decay according to the power law,

$$|f(\nu(n))| \leq R \cdot n^{-1/r}, \quad n = 1, \ldots, N,$$

where $\nu$ is the index of the sorted coefficients. Then, such signals can be approximated by $K$-sparse signals due to the rapid decay of the coefficients as the approximation error obeys,

$$\delta_{\Psi}(X, K) \leq C_r \cdot R \cdot K^{-s}, \quad s = \frac{1}{r} - \frac{1}{2},$$

where $C_r$ is a constant depending on $r$. In other words, the signal’s best approximation error has a power-law decay with exponent $s$ as $K$ increases and such a signal is called $s$-compressible.

In order to enable good estimation of $X$ from the $M$ measurements, the sensing matrix $\Phi$ should satisfy the Restricted Isometry Property (RIP) [23]. A matrix $\Phi$ is said to satisfy the RIP if there exists a constant $0 < \delta_K < 1$ for all $K$-sparse vectors $x$ such that,

$$(1 - \delta_K)\|x\|_2^2 \leq \|\Phi x\|_2^2 \leq (1 + \delta_K)\|x\|_2^2,$$

where $\|\cdot\|_2$ is the $\ell_2$-norm of a vector and coefficient $\delta_K$ is denoted as the Restricted Isometry Constant (RIC) defined as the smallest constant for which the RIP property holds.

In addition, different metrics can be used to evaluate the performance of CS in particular cases. It is already reported [39, 40] that the independence of
columns of the sensing matrix $\Phi$ decides the sparse signal recovery performance. In other words, an admissible sensing matrix for the sparse signal recovery needs its columns to be statistically independent. The coherence parameter $\mu(\Phi)$ has been introduced,

$$\mu(\Phi) = \max_{i \neq j} |\langle \Phi_i, \Phi_j \rangle|,$$

where $\mu(\Phi)$ represents the maximum absolute inner product between the two distinct columns of $\Phi$.

Natural choices of admissible sensing matrix [26] are normalized Bernoulli and Gaussian distributions:

- Bernoulli distribution: Independently select each entry of $\Phi$ to be $\pm \frac{1}{\sqrt{M}}$ with equal probability.

- Gaussian distribution: Independently select each entry of $\Phi$ from $NORMAL(0, M^{-1})$ distribution. The density of function $p(x)$ of the distribution is as follows,

$$p(x) = \frac{1}{\sqrt{2\pi M}} e^{-x^2M/2}, \text{ for } x \in \mathbb{R}.$$  

The RIP property is proven [38] to hold for such randomly chosen matrices with high probability when

$$K \leq C \frac{M}{\log(N/M)},$$

where $C$ is a function of the RIC. In addition, it has been proven that the coherence parameter $\mu(\Phi)$ is fairly small [26] for a $M \times N$ normalized Bernoulli
matrix,
\[ P\{\mu(\Phi) \leq \sqrt{4M^{-1}ln(N/\varepsilon)}\} \geq 1 - \varepsilon^2, \]
(10)
where \( \varepsilon \in (0, 1) \). The coherence parameter \( \mu(\Phi) \) for a normalized Gaussian matrix has also been proven to be a small number with high probability. In practice, a pseudo-random \( \pm 1 \) Bernoulli generator [41, 42] is commonly used to facilitate an efficient hardware realization obtaining the compressed signal \( Y \) from original signal \( X \).

2.2 Sparse Signal Recovery

As fewer measurements are obtained using the tools provided by compressive sensing, a method is needed to acquire the knowledge of the locations and values of those significant elements or coefficients of the signal. There exist two major algorithmic approaches to solving the compressive sensing signal reconstruction problem. The first approach is to use linear programming method to solve the optimization problem, and the second approach is developed from the greedy algorithms.

2.2.1 Basis Pursuit

It has been proven that using \( \ell_0 \)-minimization, signal \( X \) can be reconstructed from the measured signal \( Y \) with a high probability.
\[ \hat{X} = \arg \min_Z \|Z\|_0 \quad s.t. \quad \Phi * Z = Y, \]
(11)
where

$$\|Z\|_0 = \sum_{i=1}^{n} |Z_i|^0$$  \hspace{1cm} (12)

denotes the $\ell_0$-norm of the vector $Z$ and is the number of nonzero entries of the vector it operates on.

However, the $\ell_0$-minimization method must search all possible combinations to locate the non-zero components in $X$ and the computational complexity is combinatorial. In other words, this problem is generally known to be NP-hard [43, 44] due to the exhaustive search method.

Fortunately, there exist numerically feasible alternatives to this NP-Hard problem in compressed sensing. A lot of research effort [45, 46, 47, 48, 49, 50, 51] is then directed towards finding alternative solutions. One classical approach of signal reconstruction for compressive sensing relaxes the $\ell_0$-minimization to the $\ell_1$-minimization and applies the convex optimization to reconstruct the original signal in an over-complete dictionary, such as

$$\hat{X} = \arg \min_{Z} \|Z\|_1 \hspace{1cm} s.t. \hspace{1cm} \Phi * Z = Y,$$  \hspace{1cm} (13)

where

$$\|Z\|_1 = \sum_{i=1}^{n} |Z_i|$$  \hspace{1cm} (14)

denotes the $\ell_1$-norm of the vector $Z$.

The $\ell_1$-minimization approach is a convex optimization problem and can be solved based on the Linear Programming (LP) [47, 49, 51]. The LP algorithms, also known as Basis Pursuit (BP), are executed in polynomial time [23] and
involve high computational complexity even though some fast algorithms have been proposed [25, 52].

### 2.2.2 Greedy Recovery Algorithm

To recover the original signal $X$, the other alternative approach to compressed sensing is the use of greedy algorithms [26, 50, 51, 53, 54], which iteratively compute the support of the sparse signal and the pseudo-inverse of the corresponding columns in the measurement matrix. Among the greedy algorithms, Orthogonal Matching Pursuit-based reconstruction as **Algorithm 1** shows is getting popular because of its simple geometric interpolation and relatively low complexity of hardware implementation.

**Input**: measurement matrix $\Phi$; measurement signal $Y = \Phi X$; target sparsity $K$

**Output**: $\hat{X}$ and $I$

**Initialize**: Index set $I = \emptyset$; Residual Vector $R = Y$; Vector $\alpha = \Phi^T Y$;
Repeat the following $K$ times:

**Support Identification**: Add the coordinate of the entry with the largest absolute value in $\alpha = \Phi^T R$ to Index set $I$.

**Signal Update**: $\hat{X} = \arg\min_Z(\| Y - \Phi_f Z \|)$, $R = Y - \Phi \hat{X}$

**Algorithm 1**: Orthogonal Matching Pursuit-based reconstruction algorithm.

Several researches have been done to make sure that OMP algorithm is reliable to recover the sparse signals. Tropp has shown [39] that condition $\mu(\Phi) < \frac{1}{2K-1}$ suffices for the OMP algorithm to exactly recover the support of $K$-sparse signals. Also, Kunis's empirical work [55] suggests that the number measurements at
the level of $O(K \ln N)$ is sufficient for OMP to recover an $K$-sparse signal and OMP can achieve exact support recovery of $K$-sparse signals without noise under condition $\delta_{K+1} < \frac{1}{\sqrt{K+1}}$.

In addition, the OMP algorithm can also work with an error bound on the residual vector as a halting criterion [56, 57, 58] due to the noise in measurement $Y$ or that sparsity $K$ remains unknown. To simplify the theoretical analysis in some parts of the dissertation, as done in [26, 56, 59], the sparsity $K$ or its estimation $\hat{K}$ is assumed known in this dissertation, either through error bound, pre-estimation [60] or other ways [61, 62]. Still, this dissertation will incorporate the fact of inaccurately estimating $K$ for some simulation results.

2.3 Applications

The sparse signal recovery of compressive sensing relies first and foremost on the signal sparsity, and fortunately in practice many signals are naturally sparse in certain domain. Thus, compressive sensing theory can be well applied to many applications. The applications include compressive imaging, analog-to-information converters and compressive radar.

2.3.1 Compressive Imaging

Images are known to be sparse either in nature or some basis and thus many imaging applications [9, 10, 11, 63, 64, 65] can be benefitted from the tools provided by compressive sensing. Applications of the theory of compressive sensing
to the imaging enables sub-Nyquist image acquisition; far fewer measurements were performed than the conventional imaging while reliable image reconstruction was still possible. Everyday digital cameras record and store every pixel which operate in the mega-pixel range but most of this abundance of data is thrown. Instead, a new single-pixel camera [9], based on the framework of compressive sensing, is proposed acquire fewer measurements. The single-pixel camera can also operate at broader range of the light spectrum than silicon-based traditional cameras. Another work [10] developed computational sensor systems capable of implementing compressive imaging operations. The compressive sensing-based compression technique [63, 64] also offers the astronomy an efficient way of handling astronomical data compression and multiple observations of the same field view.

Many magnetic resonance images are actually sparse when represented in pixels and others can be transformed to some basis to be sparse. As speed of acquiring magnetic resonance image in traditional way is very limited leading to time-costly imaging, medical imaging [12, 13, 14, 66, 67] can be extremely benefited from compressive sensing with reduced magnetic resonance imaging (MRI) acquisition time.
2.3.2 Analog-to-information Converters

Many radar and communication systems, especially those are operated in the radio frequency (RF) bands, are of very high bandwidth. Then, the analog-to-digital converter (ADC) technologies are greatly stressed to be faster and more precise. As an alternative, compressive sensing based analog-to-information converters (AIC) [15, 16, 17, 18, 68] focus only on the relevant information, which enables a great reduction of the digital data rate. As a result, application of CS to the signal acquisition receivers can increase the input bandwidth and greatly lower the cost of the signal receiver. In wireless communication systems, compressive sensing is also used in sparse channel estimation [19, 20, 21] improving the spectrum and energy-efficiency.

2.3.3 Compressive Radar

Compressive radar imaging is one additional application of compressive sensing. There are two types of conventional radar systems [69, 70]: those systems detecting the targets using an analog matched filter to correlate the signal received with some sort of transmitted pulse; those systems that first sample and quantize the reflected radar signals and then process it as digital signals. But these conventional radar systems are expensive and of high complexity considering the sampling rate and quantization levels of ADCs, and the resolution is limited due to the time-frequency uncertainty principles.
By using the tools of compressive sensing, the radar imaging system can see great potential of better resolution over classical radar [22, 71]. The time-frequency plane in Compressive Radar Imaging is discretized into a grid and each possible target scene is considered as a matrix. Assuming the number of targets is sufficiently small, transmitted pulse is incoherent enough and then the target scene can be reconstructed employing the compressive sensing.
Chapter 3

The Matrix Inversion Bypass (MIB) Transform and Implementation

3.1 Introduction

In many embedded signal processing systems, one fundamental challenge is how to achieve satisfactory performance using scarce physical resources. For example, distributed in-field sensors need to perform autonomous data acquisition and wireless transmission subject to a wide range of physical conditions such as coverage range, detection sensitivity, system robustness, and energy availability [72, 73, 74]. This requires intelligent management of computing process with physical resources. In order to achieve this goal, one effective way is to reduce the unnecessary or redundant data to be processed. Traditional sensing techniques often obtain far more data than necessary; i.e., only a small portion of the obtained data are actually useful while the rest can be tossed away without affecting
the performance in most cases. This motivates a new research field called Compressive Sensing (CS) [3], which allows efficient data acquisition based on the observation that most signals obtained from the physical world can be viewed as sparse or transformed into some domains that are sparse. Exploiting compressive sensing, the physical resources needed for data acquisition and transmission can be significantly reduced.

One critical issue in compressive sensing is to recover the original signal from a few number of measurements in a reliable and efficient way. A lot of research effort has been directed towards investigating compressive sensing signal reconstruction algorithms [23, 24, 25, 26, 27, 28, 29] for this purpose. Two major existing approaches are the $\ell_1$-minimization and greedy algorithms. The $\ell_1$-minimization can be executed based on the Linear Programming (LP) [23], also known as Basis Pursuit (BP). Many fast algorithms have been proposed [25] to reduce the BP algorithm complexity. Another class of algorithms based on the iterative greedy pursuit has also received significant attention recently. The family of greedy algorithms include the Orthogonal Matching Pursuit (OMP) [26], Regularized OMP (ROMP) [27], Stagewise OMP (StOMP) [28] and Subspace Pursuit (SP) [29]. Although typically requiring more measurements than the $\ell_1$-minimization in order to attain successful signal reconstruction with a high probability [26], the OMP-based greedy algorithms have demonstrated many good features such as simple geometric interpolation and low computational complexity. Despite these
benefits, the reconstruction time of the OMP-based greedy algorithms is still formidable as the size of the original signal increases.

While significant progress has been made in algorithm optimization, only a few results were reported on the implementation of signal reconstruction algorithms. In general, dedicated hardware implementations are energy-efficient and thus are preferable in many resource-constrained embedded signal processing systems. A high-speed architecture based on the $\ell_1$-minimization was proposed in [75] for image reconstruction. An integrated circuit implementation of OMP algorithm was presented in [30], where the OMP algorithm was directly mapped into hardware units to fulfill the procedures sequentially. Adopting a similar design, the work in [76] improved the implementation of compressive sensing by finding the fast inverse square root of fixed point numbers. In order to improve the signal reconstruction speed, both designs in [30, 76] applied the pre-determined size and sparsity for the original signal while the parameters of the signal were varied under the same sampling technique.

In this chapter, an algorithmic transformation technique referred to as Matrix Inversion Bypass (MIB) is proposed to improve the reconstruction time and reduce the computational complexity of OMP algorithm. The key idea is to decouple the update operation of intermediate estimates and the time-consuming matrix inversion operation required in the OMP algorithm, thereby eliminating the speed bottleneck in hardware implementations. The proposed MIB technique
naturally leads to a parallel architecture for dedicated hardware implementations. It is shown that by reducing the reconstruction time, the energy consumption needed to recover the original signal can be reduced as well. This is important for many embedded signal processing systems that are deployed under severe physical/energy constraints.

In addition, a dedicated architectural design of the proposed MIB technique is developed. This design can benefit application-specific hardware platforms such as dedicated hardware accelerators for embedded signal processing systems. The proposed MIB technique is also applied in a CS-based video application to demonstrate the performance improvement.

The rest of this chapter is organized as follows. Section 3.2 briefly introduces the procedures of the OMP algorithm. In Section 3.3, the Matrix Inversion Bypass (MIB) transformation is developed. The architectural implementation of the proposed technique is presented in Section 3.4. Section 3.5 evaluates the proposed technique and Section 3.6 summarizes this chapter.

3.2 OMP algorithm

The procedures of OMP algorithm [26] can be summarized in Algorithm 2. In this algorithm, \( I_k \in \mathbb{R}^k \) is the index set of the selected columns in the measurement matrix \( \Phi \) at the \( k^{th} \) iteration, \( \tilde{I}_k \in \mathbb{R}^{N-k} \) is its complementary set, and \( \Theta_{I_k} (\Phi_{\tilde{I}_k}) \) is the matrix containing only the columns with the indices in \( I_k \).
Note that the matrix $\Phi$ can be replaced by $\Theta = \Phi \Psi$ if the signal is sparse in the basis $\Psi$.

**Input**: measurement matrix $\Theta$; sampled signal $Y$; target sparsity $K$;

**Initialize**: Iteration counter $k = 1$; Index set $I_0 = \emptyset$; Residual Vector $R_0 = Y$; Vector $\alpha_0 = \Theta^T Y$;

while $k < K$ do

2.1 $i_k = \arg\max_j(|\alpha_k|_j)$

2.2 $I_k = I_{k-1} \cup i_k$

2.3 $\hat{X}_k = \arg\min_z(\|Y - \Theta_{I_k} Z\|)$

2.4 $R_k = Y - \Theta_{I_k} \hat{X}_k$

2.5 $\alpha_k = \Theta_{I_k}^T R_k$

2.6 $k = k + 1$

**Output**: $\hat{X}_K$ and $I_K$

**Algorithm 2**: Procedures of the Orthogonal Matching Pursuit reconstruction algorithm.

To start with, $I_0$ is an empty set and $R_0$ is initialized with $Y$. The OMP algorithm determines which entries of the signal are non-zero through finding the columns in $\Theta$. At each iteration, the column of matrix $\Theta$ most correlated with the residual vector $R_k \in \mathbb{R}^N$ is chosen and its coordinate is added into $I_k \in \mathbb{R}^k$. After index set $I_k$ is updated, the algorithm minimizes approximation error in step 2.3 by solving a least square problem [77] and updates the estimate of the original signal $\hat{X}_k \in \mathbb{R}^k$ as follows,

$$\hat{X}_k = \Theta_{I_k}^\dagger Y, \quad s.t. \quad \Theta_{I_k}^\dagger = \left(\Theta_{I_k}^* \Theta_{I_k}\right)^{-1} \Theta_{I_k}^*,$$

(15)
where \( \hat{X}_k \in \mathbb{R}^k \) is current estimate of signal \( X \) in the sparsity basis \( \Psi \). Then, the residual vector \( R_k \), and the inner products of \( \Phi^T I_k \) and \( R_k \) are updated, respectively. After \( K \) iterations, the estimated \( K \) non-zero components of the original signal \( X \) are obtained along with the corresponding index set showing the coordinates of these non-zero components in the signal \( X \).

### 3.3 The proposed Matrix Inversion Bypass (MIB) Transform

In the signal recovery of the OMP algorithm, the amount of computations can be reduced to improve the efficiency of the OMP algorithm implementation using the same measuring matrix to recover a large number of signals. For the OMP algorithm (see Algorithm 2), the column selection in the step 2.3 does not need to know \( R_k \) explicitly and thus it calculates \( \alpha_k \) directly. In addition, the Batch-OMP algorithm [57] pre-computes the product of \( \Phi^T \) and \( \Phi \) to reduce the runtime workload in solving the least square problem in step 2.5 of the OMP algorithm.

However, the signal reconstruction computation still requires intensive matrix operations and matrix inversion is still needed in solving the least square problem (step 2.5 of Algorithm 2). This becomes the speed bottleneck in the OMP-based signal reconstruction, in particular when the sparsity of the signal is large. In this section, the Matrix Inversion Bypass (MIB) transform is proposed to address this problem.
3.3.1 Matrix Inversion Bypass

Throughout the rest of the chapter, denote $G = \Phi^T \Phi$ and $G_{I,J} = \Phi_I^T \Phi_J$, where $I$ and $J$ can be any index set. A speed-limiting step in the OMP algorithm is to estimate the value of the reconstructed signal at each iteration by solving the following problem

$$\hat{X}_k = \arg \min_Z (\| Y - \Phi_{I_k} Z \|).$$

(16)

This problem involves a least square minimization that is usually solved by the Moore-Penrose pseudo-inverse [77], expressed as

$$\hat{X}_k = \Phi_{I_k}^\dagger Y, \quad s.t. \quad \Phi_{I_k}^\dagger = G_{I_k,I_k}^{-1} \Phi_I^{T}.$$  

(17)

where $G_{I_k,I_k} \in \mathbb{R}^{k \times k}$ is a symmetric positive definite (SPD) matrix. One way to obtain the inverse of a SPD matrix is to use the Cholesky Decomposition (CD) [78], which decomposes the matrix into the product of sub-matrices in some canonical forms, inverts these matrices, and then multiplies them to obtain $G_{I_k,I_k}^{-1}$. This procedure requires a lot of operations such as multiplications, divisions, and even finding the square root in some cases, which introduce large computational complexity. To avoid the square root calculation, the Alternative Cholesky Decomposition (ACD) [30] was proposed but it still needs many complex operations such as divisions.

Another effective way to implement the Moore-Penrose pseudo-inverse is referred to as the Updated Pseudo-inverse method [79], which utilizes the matrix $G_{I_k-1,I_{k-1}}^{-1}$ available from the previous iteration to obtain the current matrix.
Specifically, this method calculates $G_{I_k,I_k}^{-1}$ using the Schur-Banachiewicz block-wise inversion [80], as shown below

$$G_{I_k,I_k}^{-1} = \begin{bmatrix}
G_{I_{k-1},I_{k-1}} & G_{I_{k-1},i_k} \\
G_{i_k,I_{k-1}} & G_{i_k,i_k}
\end{bmatrix}^{-1}$$

$$= \begin{bmatrix}
G_{I_{k-1},I_{k-1}}^{-1} + VA^T A & -VA^T \\
-V A & V
\end{bmatrix},$$

(18)

where

$$V = \frac{1}{G_{i_k,i_k} - G_{i_k,I_{k-1}} G_{I_{k-1},I_{k-1}} G_{I_{k-1},i_k}} \in \mathbb{R},$$

(19)

and

$$A = G_{i_k,I_{k-1}} G_{I_{k-1},I_{k-1}}^{-1} \in \mathbb{R}^{k-1}.$$  

(20)

To solve equation (17), the original OMP-based algorithms have to compute $G_{I_k,I_k}^{-1}$, $\Phi_{i_k}^\dagger$ and $\hat{X}_k$ in sequence. Different from these algorithms, the update of signal estimate $\hat{X}_k$ and the calculation of matrix $G_{I_k,I_k}^{-1}$ can be decoupled in the proposed MIB transform; i.e., through bypassing the matrix inversion operation, the speed of signal reconstruction can be greatly improved.

It is observed that when $k > 1$, the matrix $\Phi_{I_k}$ in the $k^{th}$ iteration can be expressed as

$$\Phi_{I_k}^T = \begin{bmatrix}
\Phi_{I_{k-1}}^T \\
\Phi_{i_k}^T
\end{bmatrix} \in \mathbb{R}^{k \times M}.$$  

(21)
Substituting $\Phi_{I_k}^T$ from equation (21) into (17) and utilizing equation (18) and (19), the computation of $\hat{X}_k$ at each iteration can be transformed to

\[
\hat{X}_k = G_{I_k,I_k}^{-1} \Phi_{I_k}^T Y
\]

\[
= \begin{bmatrix}
G_{I_{k-1},I_{k-1}}^{-1} + VA^T A & VA^T \\
-VA & V
\end{bmatrix}
\begin{bmatrix}
\Phi_{I_{k-1}}^T Y \\
\Phi_{I_k}^T Y
\end{bmatrix}
\]

\[
= \hat{X}_{k-1} + VA^T A \Phi_{I_{k-1}}^T Y - VA^T \Phi_{I_k}^T Y \\
-VA \Phi_{I_{k-1}}^T Y + V \Phi_{I_k}^T Y
\]

Define a scalar $M$ as

\[
M = A \Phi_{I_{k-1}}^T Y - \Phi_{I_k}^T Y \in R,
\]

then

\[
\hat{X}_k = \begin{cases}
G_{I_k,I_k}^{-1} \Phi_{I_k}^T Y, & k = 1 \\
\hat{X}_{k-1} + VMA^T, & k > 1
\end{cases}
\]

As shown in equation (24), except for the initial iteration $k = 1$, the computation of $\hat{X}_k$ can be transformed by only utilizing the results from the $(k - 1)^{th}$ iteration. In other words, the computation of $G_{I_k,I_k}^{-1}$ for the current iteration can be bypassed. As $G_{I_k,I_k}^{-1}$ is now needed only in the $(k + 1)^{th}$ iteration, it can be computed in parallel with the computation of $\hat{X}_k$ so that the speed of signal reconstruction can be improved.
Note that other variables in equation (24) such as $M$, $A$ and $V$ are easy to obtain, as $G_{i_k,i_{k-1}}, G_{i_k,i_k}$ and $G_{I_{k-1},i_k}$ in equation (19) and (20) are the sub-matrices of $G$. Thus, the computational complexity of equation (24) is much less than that of equation (17) in the OMP algorithm.

After $\hat{X}_k$ is obtained, the value of $\alpha_k$ at the step 2.5 of the OMP algorithm (see Algorithm 2) can be updated according to

$$
\alpha_k = \begin{cases} 
\Phi_{I_k}^T Y & , k = 0 \\
\Phi_{I_k}^T Y - G_{I_k,i_k} \hat{X}_k & , k > 0
\end{cases}
$$

(25)

Employing the proposed MIB transform, the computation of $\alpha_k$ can be further simplified as

$$
\alpha_k = \Phi_{I_k}^T Y - G_{I_k,i_k} \hat{X}_k
$$

$$
= \Phi_{I_k}^T Y - \Phi_{I_k}^T [\Phi_{I_{k-1}} \Phi_{i_k}] \cdot \begin{bmatrix} \hat{X}_{k-1} + VMA^T \\
-VM
\end{bmatrix}
$$

$$
= \Phi_{I_k}^T Y - \Phi_{I_k}^T (\Phi_{I_{k-1}} \hat{X}_{k-1} + \Phi_{I_{k-1}} VMA^T - \Phi_{i_k} VM)
$$

$$
= \Phi_{I_k}^T (Y - \Phi_{i_k} \hat{X}_{k-1}) - \Phi_{I_k}^T (\Phi_{I_{k-1}} VMA^T - \Phi_{i_k} VM)
$$

$$
= \alpha_{k-1} - \Phi_{I_k}^T [\Phi_{I_{k-1}} \Phi_{i_k}] \cdot \begin{bmatrix} VMA^T \\
-VM
\end{bmatrix}
$$

$$
= \alpha_{k-1} - G_{I_k,i_k} \cdot \begin{bmatrix} VMA^T \\
-VM
\end{bmatrix}
$$

(26)
Thus, equation (25) can be implemented in an equivalent way as

\[
\alpha_k = \begin{cases} 
\Phi^T_{I_k} Y, & k = 0 \\
\alpha_{k-1} - G_{\tilde{I}_k, I_k} \cdot \begin{bmatrix} VMA^T \\ -VM \end{bmatrix}, & k > 0
\end{cases}
\]  

(27)

where the matrix \(G_{\tilde{I}_k, I_k}\) can be pre-computed as in the Batch-OMP [57]. As a result, the computational complexity of updating \(\alpha_k\) at each iteration is also reduced by the proposed MIB transform.

### 3.3.2 Complexity Comparison

![Flow chart of the MIB-based signal reconstruction algorithm.](image)

The proposed MIB transform is summarized in Figure 1. The parameters are initialized as follows: iteration counter \(k = 0\), matrix \(G = \Phi^T\Phi\); \(\alpha_0 = \Phi^T Y\),
\[ i = \arg \max_j (|\alpha_0|), \quad I_0 = \emptyset, \quad I_1 = \{i_k\}, \quad \text{and} \quad G^{-1}_{I_1,I_1} = \frac{1}{\Phi_{I_1}^T \Phi_{I_1}}. \] Some key operations are explained below.

- When \( k = 0 \), the initial vector \( \Phi^T Y \) can be saved for computations in the subsequent iterations.

- When \( k = 1 \), the vector \([VMA^T, -VM]\) only contains \(-VM = \Phi_{I_1}^T Y G^{-1}_{I_1,I_1}\) as the vector \( A^T \) is empty.

- When \( k > 1 \), the computation of \( \alpha_k \) and the matrix inversion \( G^{-1}_{I_k,I_k} \) are decoupled and both need to use some variables such as the scalar \( V \) and vector \( A \). Thus, these two time-consuming computations can be performed in parallel. This enables a high-speed architecture in implementing the proposed MIB transform, as discussed in Section 3.4.

- To compute \( \alpha_k \) only needs to update \([VMA^T, -VM]\); computing vector \( \hat{X}_k \) is not necessary for the iteration to continue. However, \( \hat{X}_k \) still needs to be updated through a simple addition of two vectors

\[
\hat{X}_k = \begin{bmatrix}
\hat{X}_{k-1} \\
0 \\
\end{bmatrix} + \begin{bmatrix}
VMA^T \\
-VM \\
\end{bmatrix}.
\] (28)

Figure 2 compares the operation flow of the proposed MIB transform and the original Batch-OMP method. The computational complexity of these two methods is then evaluated here. As shown in equation (27), vector \([A^T, -1]\) needs to be multiplied by the product of \( V \) and \( M \) to obtain \([VMA^T, -VM]\).
Thus, the acquisition of $\hat{X}_k$ needs one addition between two $k \times 1$ vectors. In comparison, the Batch-OMP algorithm computes equation (17) using $G_{I_k,I_k}^{-1}$ and $\Phi^T_{I_k} Y$ is a product between a $k \times k$ matrix and a $k \times 1$ vector. Clearly, the proposed MIB transform introduces much smaller computational complexity in updating $\hat{X}_k$.

The number of arithmetic operations needed to update $\hat{X}_k$ in each iteration is summarized in Table 1. Compared with the Batch-OMP method, the proposed MIB transform requires $(k^2 - 2k + 1)$ fewer multiplications and $(k^2 - 2k + 1)$ fewer additions in the $k^{th}$ iteration. Furthermore, by parallelizing the computations
of $\alpha_k$, $\hat{X}_k$, and $G_{I_k,I_k}^{-1}$, the critical path can be shortened by $\frac{1}{2}(k^2 - 3k + 2)$ multiplications and $\frac{1}{2}(k^2 - k)$ additions in the $k^{th}$ iteration. This leads to a high-speed architecture that will be discussed in the next section.

Table 1: The number of arithmetic operations needed to update $\hat{X}_k$

<table>
<thead>
<tr>
<th>$k^{th}$ step</th>
<th>Number of multiplications</th>
<th>Number of additions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Batch-OMP</td>
<td>$k \times k$</td>
<td>$k \times (k - 1)$</td>
</tr>
<tr>
<td>MIB-OMP</td>
<td>$2 \times (k - 1) + 1$</td>
<td>$k - 1$</td>
</tr>
</tbody>
</table>

3.4 Architectural Design for Hardware Specialization

In this section, the architectural design of the proposed MIB transform is presented. This design can benefit application-specific platforms such as dedicated hardware accelerators for embedded signal processing systems.

3.4.1 System Architecture

The top-level architecture based on the proposed MIB transform is shown in Figure 3. It can be divided into several functional units that update $\Phi^T Y$, $[VMA^T, -VM]$, $G_{I_k,I_k}^{-1}$, $\alpha_k$, $\hat{X}_K$ and $I_k$. One system memory is adopted to store the intermediate results accessible by these units. During the iterative signal reconstruction, updating $G_{I_k,I_k}^{-1}$ and $I_k$ needs to be finished before the start of the next round of iteration as shown in Figure 2. The estimated signal $\hat{X}_K$ and the corresponding index vector $I_K$ are the recovered signals. Note that the elements
in $I_k$ indicate the rows and columns in $G$ and $\Phi^T Y$, respectively, that are needed to update $[VMA^T, -VM]$, $\alpha_k$ and $\hat{X}_k$ in each iteration. Employing the proposed MIB transform, updating $G_{I_k,I_k}^{-1}$ is performed in parallel with updating $\alpha_k$, $\hat{X}_k$ and $I_k$, which reduces the computation time of each iteration. This is shown in Figure 3, where the two shadow blocks are operated in parallel. Also, updating $G_{I_k,I_k}^{-1}$ only needs to be finished before the update of $[VMA^T, -VM]$ for the next iteration. Once $\hat{X}_k$ and $I_k$ are updated, the current round of iteration is finished and the iteration number $k$ is increased by one.

The measured signal $Y$ and the corresponding sparsity $K$ are fed into the system from the input memory to initialize the signal reconstruction. Note that the signal sparsity $K$ can be estimated as in [26, 56, 59], either through the
error bound, pre-estimation [60] or other methods [61, 62]. This topic, however, is beyond the scope of this dissertation. During the signal reconstruction, the iteration number \( k \) is compared with the sparsity \( K \). Once they are equal, signal reconstruction is finished and the estimated signal \( \hat{X}_K \) and index vector \( I_K \) are written into the output memory. The iteration number \( k \) is then reset to 0 so that the next signal can be processed.

### 3.4.2 Data Structure

This chapter considers the measuring matrix \( \Phi \) to follow the Bernoulli distribution that consists of 0 and 1 only. Since \( \Phi \) is only used when updating \( \Phi^T Y \) at the beginning of signal reconstruction, it is stored in a memory in the unit that updates \( \Phi^T Y \) and \( \alpha_0 \) (see Figure 4). Note that matrix \( \Phi \) can be utilized repeatedly to recover different sets of signals obtained by the same measuring matrix \( \Phi \).

During the signal reconstruction, \( G, \Phi^T Y \) and \( \alpha_k \) have relatively large sizes and need to be accessed repeatedly in different units. Thus, these data are stored in the system memory. The size of the memory for \( \Phi^T Y \) and \( \alpha_0 \) is \( 2 \times N \times B \), where \( N \) is the length of original signal vector and \( B \) is the bit width of each signal. The size of the memory for symmetric \( G \) can be reduced by about a half to \( \frac{1}{2} \times N \times (N - 1) \times BM \), and \( BM \) is the bit width of each element is \( G \). Vector \( \Phi^T Y \) is calculated once the measured signal \( Y \) is fed in, and its sub-set \( \Phi^T_{I_k} Y \) needs to be read at each iteration to update the vector \( [VMA^T, -VM] \). Vector
\( \alpha_k \) needs to be updated at each iteration to update the vector \( I_k \), which is used to locate the sub-matrix of \( G \) and \( \Phi^T Y \) during signal reconstruction. The size of \( I_k \) directly determines the length of the vectors read from the memory, such as \( \Phi^T I_k Y \) and \( G_{I_k} \).

![Figure 4: Update \( \Phi^T Y \) and \( \alpha_0 \).](image)

Other data, such as \( G^{-1}_{I_k,I_k} \), \([VMA, -VM]\) and \( \hat{X}_k \), have much smaller sizes than \( G \), \( \Phi^T Y \) and \( \alpha_k \). These data are stored in the internal data register so that they are readily accessible by the related operations.

### 3.4.3 Initialize \( \Phi^T Y \)

For each signal to be recovered, the measured signal \( Y \) is first multiplied with the matrix \( \Phi \). Since the matrix \( \Phi \) is in the Bernoulli distribution, the computation of \( \Phi^T Y \) only involves addition operations. The hardware architecture for updating \( \Phi^T Y \) is shown in Figure 4. The column elements of \( \Phi \), stored in the internal memory, are used in sequence as the control signal of the multiplexer,
which selects the corresponding elements in $Y$ to be accumulated. Note that a zero in $\Phi$ indicates the subtraction operation.

### 3.4.4 Update $\alpha_k$

As updating $A$, $M$, $V$, $[VMA^T, -VM]$ and $\alpha_k$ are performed in sequence, the hardware cost can be reduced by reusing the multiplication and addition units. The architecture of updating $\alpha_k$ and $\hat{X}_k$ is shown in Figure 5. The main processing unit is designed to be able to perform either multiply, add or multiply-accumulate operations [81]. The inverse unit obtains the multiplicative inverse of the input number. Data read from the memory or the register are fed to the processing unit and the calculated results are stored to the registers. The procedure is as follows:

![Figure 5: Update $\alpha_k$.](image-url)
Matrix $G_{I_{k-1},I_{k-1}}^{-1}$ obtained from the previous iteration is first multiplied by $G_{i_k,I_{k-1}}$ (from the system memory) to derive the vector $A$. The obtained $A$ is then used to calculate scalar $1/V$ and $M$ by multiplying with $G_{I_{k-1},i_k}^{-1}$ and $\Phi_{I_{k-1}}^{T}Y$, respectively. The operations above are all inner products of two vectors.

Once updated, $1/V$ undergoes the inversion operation to generate $V$. Then, $M$ obtained from the above step and $V$ are fed into the multiplier to obtain $VM$, followed by $A$ and $VM$ to calculate $VMA$. The operations here are scalar multiplications. $A$ and $-V$ will also be used to calculate $G_{I_k,I_k}^{-1}$ for the next iteration (see Section 3.4.6).

Vector $[-VMA^{T}, VM]$ obtained from the above step is multiplied by the rows of $G_{I_{k-1}}$ and the inner products undergo the addition operation with $\alpha_{k-1}$ to obtain $\alpha_k$. The updated elements of $\alpha_k$ will be used for updating the vector $I_k$ (see Section 3.4.5). Meanwhile, $[-VMA^{T}, VM]$ and $[\hat{X}_{k-1}, 0]$ are added to update vector $\hat{X}_k$.

Note that at the beginning of reconstruction, vector $\Phi^{T}Y$ is directly assigned to $\alpha_k$ as $\alpha_0 = \Phi^{T}Y$. Thus, the adder in Figure 4 and the processing unit in Figure 5 have no time overlapping during the signal reconstruction, i.e., the processing unit can be shared. Also, both $G_{I_{k-1},I_{k-1}}^{-1}$ and $A$ are initially empty and only $-VM$ is computed to update $\alpha_1$ and $\hat{X}_1$. 
3.4.5 Update $I_k$

The unit for updating the index vector $I_k$ is shown in Figure 6. As $I_k$ is being updated, $\alpha_k(n)$ and the corresponding index $n$ are fed in. At the beginning, $\alpha_k(1)$ and 1 are assigned to $\alpha_{\text{temp}}$ and $i_{\text{temp}}$, respectively. As $n$ increases to $N$, $\alpha_k(n)$ is compared with $\alpha_{\text{temp}}$ based on the absolute value. If $\alpha_k(n)$ is larger, $\alpha_{\text{temp}}$ and $i_{\text{temp}}$ will be replaced by $\alpha_k(n)$ and $n$, respectively. When $n$ reaches $N$, $i_{\text{temp}}$, which is the index of the largest element in $\alpha_k$, is added into $I_{k-1}$.

![Figure 6: Update $I_k$.](image)

3.4.6 Update $G_{I_k,I_k}^{-1}$:

Figure 7 shows the unit that updates $G_{I_k,I_k}^{-1}$ based on equation (18). The parallel updating process starts as soon as $A$ and $-V$ are updated as shown in Section 3.4.4.
When $k = 1$, $G_{I_{k},I_{k}}^{-1}$ is equal to $V$; when $k > 1$, updating $G_{I_{k},I_{k}}^{-1}$ requires more operations. First, each element in the vector $A$ is multiplied with $-V$ and the result $-VA$ is then multiplied by $A$ to obtain the matrix $-VA^T A$. Note that both operations here are scalar multiplications, and thus the scalar multiplier can be reused.

The updated $-VA$ and $V$ are saved into $G_{I_{k},I_{k}}^{-1}$ as the new row and column of the matrix addition result. In addition, matrix $G_{I_{k-1},I_{k-1}}^{-1}$ is replaced by the addition result of matrix $-VA^T A$ and $G_{I_{k-1},I_{k-1}}^{-1}$. Therefore, matrix $G_{I_{k},I_{k}}^{-1}$ is formed with the size increasing from $(k - 1) \times (k - 1)$ to $k \times k$.

Due to the proposed MIB transform, updating $G_{I_{k},I_{k}}^{-1}$ is performed in parallel with updating $\alpha_k$, $\hat{X}_k$ and $I_k$ to improve the speed of signal reconstruction. Since the proposed MIB technique also reduces the computational complexity, the
proposed architecture is energy-efficient as well. More results will be discussed in the next section.

3.5 Performance Evaluation

The MIB-based signal reconstruction architecture was synthesized in a 65nm CMOS process with a clock frequency of 500MHz. The original Batch-OMP was also implemented for the purpose of comparison. The hardware-related results were further verified by MATLAB simulations to ensure the correctness in signal reconstruction.

3.5.1 Evaluation of MIB Transform

Table 2: Signal parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Size of original signal</th>
<th>Number of measurements</th>
<th>Sparsity of signal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simulation A</td>
<td>512</td>
<td>150</td>
<td>64</td>
</tr>
<tr>
<td>Simulation B</td>
<td>256</td>
<td>150</td>
<td>32</td>
</tr>
</tbody>
</table>

Table 2 shows the different sets of signal parameters used to evaluate the proposed MIB transform. These input signals are obtained from a random number generator for the purpose of demonstration. Figure 8 compares the speed in terms of the number of execution clock cycles of the proposed MIB transform and the original Batch-OMP. Both implementations show the increase in clock cycles with the length and the sparsity of input signals. The proposed MIB transform
significantly improves the speed of signal reconstruction for all the input signals considered. This is contributed by two factors: (1) the parallel operations of signal update and matrix inversion via the MIB transform and (2) the reduced complexity in updating \( \hat{X}_k \) and \( \alpha_k \) (see equation (24) and (27)).

![Figure 8: Comparison of the number of clock cycles during the iteration of the proposed MIB transform and Batch-OMP.](image)

Figure 9 shows the details of the major operations in the proposed MIB transform. Updating \( \alpha_k \) and \( I_k \) are shown together as they are performed at the same time. Initializing \( \Phi^T Y \) and \( \alpha_0 \) is not shown as they take much less time, i.e., only involving some additions for each measured signal. As \( G_{I_k}^{-1} I_k \) is calculated in
parallel, as long as it is finished before the index set $I_k$ is updated, the total computation time will not increase. Thus, the matrix inversion operation is no longer a part of the speed bottleneck in the compressive sensing signal reconstruction.

![Figure 9: Comparison of the number of clock cycles in the MIB transform for (a) updating $\alpha_k$ and $I_k$, (b) updating $[VMA^T, -VM]$, and (c) updating $G_{I_k,I_k}$.

The hardware-related results of signal reconstruction implemented in a 65nm CMOS process are shown in Table 3. Compared with the Batch-OMP, the proposed MIB transform costs a little bit more areas and power consumption due to the additional hardware units needed for matrix inversion bypass and parallel updating. However, as the number of arithmetic operations is greatly reduced, the total energy consumed in signal reconstruction is actually reduced by about 15%, indicating the improvement in energy efficiency by the proposed MIB transform.
Table 3: Hardware-related measurements of the proposed MIB transform and Batch-OMP.

<table>
<thead>
<tr>
<th></th>
<th>Area ($mm^2$)</th>
<th>Total power ($\mu W$)</th>
<th>Energy Consumption ($\mu J$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Batch-OMP</td>
<td>0.050</td>
<td>130</td>
<td>2.37</td>
</tr>
<tr>
<td>MIB-OMP</td>
<td>0.061</td>
<td>151</td>
<td>2.03</td>
</tr>
</tbody>
</table>

This is important for many energy-constrained applications such as distributed on-sensor signal processing, which are typically powered by batteries or renewable energy sources.

3.5.2 Case study on Video Monitoring

Figure 10: System diagram of video processing using compressive sensing.
The proposed MIB transform is further evaluated for video processing applications. In such applications, especially remote wireless video monitoring systems under the severe energy constraint, compressive sensing is a promising technique that can reduce the amount of video signals being transmitted, thereby improving the energy efficiency of these systems. Existing work [82] chose one key video frame with full-scale measurements and applied compressive sensing to the following frames to reduce the number of measurements. The results are then transmitted wireless to a processing center for video recovery and analysis. Since the amount of transmitted data is reduced, the energy consumption of wireless transmission can be significantly reduced. One major challenge, however, is that the stream of compressively processed image frames needs to be recovered at high speed for real-time applications. The proposed MIB transform can effectively address this challenging problem.

In the experiment setup (see Figure 10), the video is recorded by the security camera from UConn Bookstore at a rate of 1 frame/second, which is sufficient for long-term security monitoring. The frame size of the video is 600 × 512 pixels. The key frame is the original image obtained from the camera and is updated every 30 seconds. Since the background of the video images is relative steady, the moving objects captured by the security camera can be seen as sparse signals. The difference between the key frame and the following frames is processed
by compressive sensing so that fewer measurements are needed. Note that signal transforms such as discrete cosine transform and discrete wavelet transform (DWT) \cite{83} are not needed for signal recovery \cite{84, 85}.

As the difference between the key frame and non-key frames is the sparse signal to be recovered, the signal sparsity will vary reflecting the different levels of object activities (pedestrians, cars, etc.) in the recorded video. This will affect the CS-based video reconstruction quality \cite{86, 87}. The average sparsity during the 12-hour recorded video and the peak signal-to-noise ratio (PSNR) \cite{88} after the signal reconstruction via the proposed MIB transform are shown in Figure 11. A low sparsity usually leads to a high PSNR as expected. Figures 12(a) and (b) show the sampled images of one key frame and one recovered non-key frame. It

Figure 11: PSNR (in dB) and sparsity for a 12-hour video.
can be seen that the recovered image has high quality and the objects in the recovered image are well recognizable.

The comparison of energy consumption for the conventional OMP and the MIB-based implementations is shown in Figure 13. It can be seen that the energy consumption varies with the signal sparsity. The proposed MIB transform reduces the energy consumption by 20 - 50%. Note that this energy comparison does not include wireless video transmission. If included, the system using the proposed technique can achieve even more energy savings due to fewer video signals being transmitted.

Another important performance measurement for video monitoring applications is the failure rate in image reconstruction under the strict real time constraint. With the varying signal sparsity, both implementations are observed to
Figure 13: Energy consumption of the two implementations.

suffer from reconstruction failures for the 1 frame/second video stream. These failures are mainly due to the sudden increase in sparsity in some frames, which requires more computations to recover these frames. The failure rates for both implementations are shown in Figure 14. The MIB-based implementation can finish signal recovery with a much lower failure rate, i.e., less than 4% during most of the time. This is typically acceptable for various video monitoring applications. Further improvement can be achieved by using more hardware resources for a higher level of parallel processing at the cost of more energy consumption in signal reconstruction.
Figure 14: Failure rate of the two implementations.

3.6 Summary

In this chapter, an algorithmic transformation technique, *Matrix Inversion Bypass* (MIB), is proposed to improve the OMP-based signal reconstruction in compressive sensing applications. By decoupling two timing-critical operations in the signal recovery iteration, the speed of signal reconstruction can be greatly improved. The architectural design of the proposed MIB transform is also presented in this chapter. This design targets application-specific hardware platforms such as dedicated hardware accelerators for embedded signal processing applications. The implementation of the proposed MIB transform is optimized to reduce hardware overheads and improve energy efficiency in a wireless video
monitoring system. Future work is directed towards identifying other suitable applications for the proposed MIB technique.
Chapter 4

FPGA Architecture and Implementation of the MIB transform

4.1 Overview

In order to achieve higher performance, researchers have been working on developing hardware implementation of compressed signal reconstruction. The OMP algorithm was implemented in an integrated circuit design in [30], which directly mapped the algorithm into hardware units to fulfill the computation sequentially. By adopting a similar design, the work in [76] improved the implementation by finding the fast inverse square root of fixed-point numbers. Both designs applied the pre-determined size and sparsity of the original signal to improve the signal recovery speed, while the parameters of the signal will vary under the same sampling technique. An FPGA implementation of the OMP algorithm
was proposed in [89], where the fixed-point implementation targets specific applications with a certain range of supported signal sparsity. A single-precision FPGA implementation was proposed in [81] for high-speed processing. The hardware resources were designed to support the possible maximum size in the parallel matrix-vector multiplications.

While significant progress has been made in optimizing the hardware implementation of compressed signal reconstruction, the computational complexity still remains high [32]. The algorithmic transformation technique referred to as Matrix Inversion Bypass (MIB) has been presented in chapter 3 reducing the computational complexity and improving the reconstruction time. The basic idea of the MIB technique is to decouple the computations of intermediate signal estimates and matrix inversions, thereby enabling parallel processing of these two time-consuming operations in the OMP algorithm.

In this chapter, an FPGA architecture based on the MIB technique is developed to eliminate the speed bottleneck with parallel computing and more efficient utilization of hardware resources. In the architecture design, the decoupled computations of the MIB are broken down to separate stages and implemented with shared computing sources. The hardware resources are dynamically allocated for high-speed computation, especially the dominant matrix-vector multiplications with varying sizes at the different stages of the iteration and the different iterations in the signal recovery. The FPGA block memory overhead is also reduced using a new data storage structure. A comprehensive suite of experiments
is performed with measured results from the FPGA to evaluate the proposed implementation such as hardware resource overheads, signal recovery speed, along with a case study on the audio signal recovery. The proposed architecture can benefit many application-specific hardware platforms with high-speed signal recovery requirements by efficient utilization of computing resources.

The rest of this chapter is organized as follows. The FPGA-based architecture leveraging the MIB technique is developed in Section 4.2. The proposed architecture is evaluated in Section 4.3, and the summary is given in Section 4.4.

4.2 FPGA Architecture and Implementation

In this section, the architecture and implementation of the MIB technique in an FPGA platform are discussed. The proposed architecture is optimized specifically for the high-speed data processing applications.

4.2.1 Architecture Design

The proposed architecture consists of two processing units that operate in parallel: the **Finding Index Unit** updates the vector $\alpha_k$ (see equation (27)) and identifies the maximum value in $|\alpha_k|$ as the new index to the set $I_k$ (see Step 2.2 in **Algorithm 2**), and the **Bypassing Unit** performs matrix inversion $G_{i_k,I_k}^{-1}$ and signal estimation $\hat{X}_k$ (see equation (18) and (22)). To feed and store the data to and from the above processing units, there is a **Data Unit** to manage
the intermediate results among different memories. The top-level diagram of this architecture is shown in Figure 15.

In the Data Unit, the same measurement matrix $\Phi$ can be used to recover various signals to reduce the amount of computations [57]. Thus, $G = \Phi^T\Phi$ is pre-computed and stored in the memory to reduce the runtime workload of updating $\alpha_k$ and the intermediate results $A$ and $V$. At the beginning, $\Phi^T Y$ is calculated and assigned to $\alpha_k$ to initialize the signal reconstruction. Also, $\Phi^T Y$ is stored in the memory to simplify the computation of $M$ in equation (23). During the signal reconstruction, the intermediate results from the two processing units will be stored into the memory. The updated vector $I_k$ at each iteration are the address set for reading the corresponding entries in $G_{i_k,I_k}$, $\Phi^T I_k Y$, $G_{\tilde{i}_k,I_k}$, and $\alpha_{\tilde{i}_k}$. 

Figure 15: Top-level architecture of the proposed MIB signal reconstruction.
For the two processing units shown in Figure 15, the Bypassing Unit involves primarily the multiplication and addition operations, while the Finding Index Unit utilizes multiplications, additions, reciprocal operations along with finding the index of an entry with the largest absolute value. To improve the hardware utilization in the two processing units, the hardware resources are configured to be reusable at the different stages of one iteration. The control flow is illustrated in Figure 16, where MA-i, i = 1, 2, ⋯ 5, denote the different stages of the finding index operation that will be explained in the next subsection.
4.2.2 Finding Index Unit

The operations in the Finding Index Unit can be generally represented as the following matrix-vector multiplication:

\[ Px + c = y, \]  

(29)

where \( P \) is an \( m \times n \) matrix, \( x \) is a \( n \times 1 \) vector, \( y \) and \( c \) are \( m \times 1 \) vectors. Note that the values of \( m \) and \( n \) are different at the different stages of the iteration. Table 4 lists these values for the \( k^{th} \) iteration along with the input data for \( P, x \) and \( c \). The multiplier and adder arrays need to be designed to accommodate the varying data sizes during the iteration.

Table 4: Stages of the finding index operation.

<table>
<thead>
<tr>
<th>Stage</th>
<th>( P )</th>
<th>( x )</th>
<th>( c )</th>
</tr>
</thead>
<tbody>
<tr>
<td>MA-1 ( (m=k-1, n=k-1) )</td>
<td>( G_{i_{k-1},i_{k-1}}^{-1} )</td>
<td>( G_{i_{k},i_{k-1}} )</td>
<td>0</td>
</tr>
<tr>
<td>MA-2 ( (m=2, n=k-1) )</td>
<td>[ -G_{i_{k},i_{k-1}} \Phi_{i_{k-1}} Y ]</td>
<td>( A^T )</td>
<td>[ -G_{i_{k},i_{k}} \Phi_{i_{k}} Y ]</td>
</tr>
<tr>
<td>MA-3 ( (m=1, n=1) )</td>
<td>( V )</td>
<td>( M )</td>
<td>0</td>
</tr>
<tr>
<td>MA-4 ( (m=k-1,n=1) )</td>
<td>( A^T )</td>
<td>( VM )</td>
<td>0</td>
</tr>
<tr>
<td>MA-5 ( (m=N-k, n=k) )</td>
<td>( G_{i_{k},i_{k}} )</td>
<td>[ -VMA^T ]</td>
<td>( \alpha_{k-1} )</td>
</tr>
</tbody>
</table>

Figure 17 shows the hardware architecture to perform the computations in Table 4, where the computations are carried out by the shared multiplier and
adder arrays. As shown in Table 4, the values of $m$ and $n$ in the matrix-vector multiplications change at the different stages. The maximum number of the entries in the dot product is the iteration number $k$, which changes from 1 to the maximum iteration number $K$. Some previous works utilized multiple Processing Elements (PEs) in parallel for this computation, where the number of sparse coefficients is limited by the number of PEs. As $n$ gradually increases to the maximum iteration number $K$ in stages MA-1, MA-2 and MA-5, in order to support a wide range of sparsity, the number of PEs usually is over-designed. On the other hand, if PEs only work in series as independent multiply-accumulate (MAC) units, the signal reconstruction will take a much longer time to finish.
In order to efficiently perform the complex tasks of matrix-vector multiplications with different sizes, the design of shared multiplier-adder arrays is used that can be operated flexibly under two different modes as explained below.

Figure 18: Collective accumulation mode.

- In the **Collective Accumulation Mode** as shown in Fig. 18, the multipliers are followed by adders working as a dynamically-configurable adder tree to compute the dot product between two vectors. For example, with \( p_0 \) and \( x \) (both are \( n \times 1 \) vectors) as the inputs to the multipliers, the \( n \) multiplication results are fed to the adders at the first level of the adder tree, and then the addition results go through the following levels of the adder tree to obtain the dot product \( p_0 x \), which will be added with \( c_0 \) to obtain \( y_0 \). In the case that \( n \) is larger than the number of the available multipliers \( T \), the dot product is obtained through several steps, with each
step getting the intermediate result between \( T \) components of \( y_0 \) and \( x \). On the other hand, if \( n < T \), the adders can be partitioned into more adder trees dedicated to obtain multiple dot products simultaneously to efficiently utilize the hardware resources. Also, after the results are saved, the adders at each level of the adder tree can be used as part of another adder tree for a different computation.

- In the Independent Accumulation Mode as shown in Fig. 19, multiple sets of multipliers, adders and multiplexers are configured as independent MACs. For example, the upper MAC in Fig. 19 calculates \( p_0x + c_0 = y_0 \) in (29), where \( p_0 \) is the first row in the matrix \( P \), \( c_0 \) and \( y_0 \) are the first components in the vectors \( c \) and \( y \), respectively. Thus, multiple rounds (e.g., the row
size \( n \) are needed to compute the dot product \( p_0 x \), after which \( c_0 \) is added to obtain \( y_0 \). With a number of these MACs working independently, the components in the vector \( y \) are computed in parallel. Note that the Independent Accumulation Mode can also be considered as a special case of the Collective Accumulation Mode with all adders working independently.

At each stage of the finding index operation, the multiplier-adder array can be dynamically configured to one of the two modes at runtime in order to improve the reconstruction speed and the utilization of the multipliers and adders. Note that the number of multipliers and adders allocated to the Finding Index Unit is fixed. Thus, the runtime decision on which mode the multiplier-adder array is configured to work is based on the size of the matrix and vector in the multiplication. Assume that the implementation allocates \( T \) multiplier-adder pairs to carry out the computation in (29). Other than the stages MA-3 and MA-4 which only involve scalar multiplications, the general guideline of allocating the multipliers and adders is

- the first \( \lfloor m/T \rfloor \times T \) components of the vector \( y \) in (29) are calculated using the Independent Accumulation mode;

- the rest components of the vector \( y \) are obtained in the Collective Accumulation mode.

As an example, if \( T = 16 \) and \( m = 34 \), the first 32 components of the vector \( y \) are calculated using the Independent Accumulation mode, with 16 components
obtained in parallel at each time. The remaining 2 components are obtained by
the Collective Accumulation mode. If only the Independent Accumulation mode
is available, two pairs of MACs will be used to calculate these 2 components while
the rest 14 MACs will be unused wasting hardware resources and increasing signal
recovery time. Thus, considering the value of $m$ shown in Table 4, the multiplier-
adder array at the stage MA-1, MA-2 and MA-5 of the $k_{th}$ iteration is configured
as follows.

- Stage MA-1: the multiplier-adder array is in the Independent Accumulation
  mode to compute the first $\lfloor (k - 1)/T \rfloor \times T$ components of the vector $A$.
The remaining $k - 1 - \lfloor (k - 1)/T \rfloor \times T$ components are obtained in the
  Collective Accumulation mode. If $k - 1$ is smaller than $T$, only Collective
  Accumulation mode is used.

- Stage MA-2: with $m = 2 < T$, both $V$ and $M$ are obtained in the Collective
  Accumulation mode.

- Stage MA-5: the multiplier-adder array is in the Independent Accumulation
  mode to compute the first $\lfloor (N - k)/T \rfloor \times T$ components of the vector $A$. The
  remaining $N - k - \lfloor (N - k)/T \rfloor \times T$ components are obtained in Collective
  Accumulation mode.

Clearly, the proposed two-mode approach allows multiple dot products to be
computed at the same time to maximize the hardware utilization and reduce the
computation time, especially when $n$ is relatively large in (29). Furthermore,
the supported sparsity in the signal recovery is not limited by the number of multipliers and adders. This provides more flexibility to the system design.

Also in this unit, the vector \( \alpha_k \) computed at stage MA-5 will be fed into the scalar comparison logic to find the index of the component with the largest absolute value. This is achieved in Fig. 20, where the result is added to the index set \( I_k \) after the updating of \( \alpha_k \) is finished.

### 4.2.3 Bypassing Unit

The Bypassing Unit computes the matrix inversion \( G_{I_k,I_k}^{-1} \) and estimates the signal \( \tilde{X}_k \). The operations involved are scalar multiplications and vector additions. According to (18), the matrix inversion first adds the new column/row in \( G_{I_k,I_k}^{-1} \), obtained from the multiplication between vector \( A \) and scalar \( V \). Then,
the elements in VA are multiplied by the vector A to obtain the matrix $VA^T A$.

Meanwhile, the elements in $G_{I_{k-1}}^{-1}I_{k-1}$ are updated by adding $VA^T A$. After the matrix $G_{I_k}^{-1}I_k$ is updated, the vector $\hat{X}_{k-1}$ is updated by the addition between $\hat{X}_{k-1}$ and $VMA$.

Figure 21 shows the implementation of the basic operation in the Bypassing Unit, where multipliers and adders are reused to perform the tasks as described above. The multiplexers select the appropriate input data from the memory units, where the input data at the different stages of the bypass operation are shown in Table 5. The multiplication results can be directly used by the adders (e.g., additions between $G_{I_{k-1}}^{-1}I_{k-1}$ and $VA^T A$). The final results of $G_{I_k}^{-1}I_k$ and $\hat{X}_k$ are saved to the corresponding memory units. Note that the multipliers and adders are reserved and only used in Bypassing Unit. This is because sharing MACs with the Finding Index Unit will incur more hardware overheads. As discussed in the next section, bypassing unit operations are less complex than finding index operations. Thus, fewer multipliers and adders will be used in the Bypassing Unit.

With the range of the signal sparsity usually known, the number of multipliers and adders for the two units can be per-determined in the implementation, so that the execution time of bypassing unit operations will always be less than that of finding index unit operations.

Employing the proposed MIB transform, the Bypassing Unit operates in parallel with the Finding Index Unit to improve the execution time. The matrix inversion starts as soon as $V$ is obtained at the stage MA-3 in the Finding Index
Figure 21: Bypassing operation.

Unit. It needs to be finished before the new index is found at the stage MA-5 in the Finding Index Unit. For signal estimation, updating the vector $\hat{X}_k$ can start as soon as $[VMA^T, -VM]$ is obtained at the stage MA-4 in the Finding Index Unit, or it can be performed after the matrix inversion is finished, as this operation is not needed in the current iteration.

Table 5: Stages of the bypassing operation

<table>
<thead>
<tr>
<th>Stage</th>
<th>MUL</th>
<th>ADD</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>BY-1</td>
<td>$A \times V$</td>
<td>$-$</td>
<td>$VA$</td>
</tr>
<tr>
<td>BY-2</td>
<td>$A \times VA$</td>
<td>$VA^T A + G_{I_{k-1},I_{k-1}}^{-1}$</td>
<td>$G_{I_{k},I_{k}}^{-1}$</td>
</tr>
<tr>
<td>BY-3</td>
<td>$-$</td>
<td>$VMA + \hat{X}_{k-1}$</td>
<td>$\hat{X}_k$</td>
</tr>
</tbody>
</table>

4.2.4 Data Unit

The input data of the matrix-vector multiplications shown in Table 4 need to be provided to the multiplier arrays in Figs. 18 and 19 at the same time. Except
for the matrix $G$, all the other data have relatively small sizes and thus can be stored in the corresponding registers where parallel data access is possible.

The size of $N \times N$ matrix $G$ is rather large (usually $N = 512$ or 1024) and thus each column or row needs to be stored in the block memory of the FPGA. Note that each memory block is individually accessible and there are usually two read ports for each memory block. By exploiting the symmetry of $G$ (e.g., $g_{ij} = g_{ji}$), the memory size can be reduced by half. However, as there are only two read ports in each block memory (here Xilinx Kintex-7 XC7K325T-FBG900 FPGA is used, see Section 5), a straightforward memory data arrangement as shown in Fig. 22(a) makes it impossible to access the components of $G$ at the same time.

Figure 22: Memory data arrangement for the matrix $G$: (a) original arrangement and (b) improved arrangement facilitating single read access of all the components.
For example, in the upper half of $G$, to read all the components in $G(i_k,:)$ (e.g., the dashed line, $i_k = 3$) at the same time is impossible, because the two read ports cannot read out $g_{1,3}$ (i.e., $g_{3,1}$), $g_{2,3}$ (i.e., $g_{3,2}$), and $g_{3,3}$ at once. In this case, the matrix-vector multiplications in stages MA-1, 2 and 5 of the Finding Index Unit that involve matrix $G$ (see Table 4) may have to wait until all the needed elements of matrix $G$ are obtained from the block memories. Since the data in $G$ are used frequently during the signal recovery iteration, using multiple read operations to access $G$ will significantly degrade the hardware performance.

To address this issue, every component of the $i^{th}$ row in the upper half of the matrix $G$ is shifted left by $(i - 1)$ columns. Then, the triangle data pattern is transformed into a rectangle organization as shown in Figure 22(b). In this new data arrangement, all the data are stored column-wise in the block memory. It is feasible to access either $G_{i_k,i_{k-1}}$ at stage MA-1 or each row of $G_{i_k,1:N}$ at stage MA-5 by one read operation through the two read ports of the block memory.

As the computations at the different stages are performed by the shared hardware resources, a design using fixed-point data format will have large hardware overhead in order to maintain the accuracy [81]. Thus, in the proposed architecture, the 32-bit single-precision floating-point format is adopted at all internal computations in the signal reconstruction to support a wide dynamic range and maintain computation accuracy.

As discussed in the next section, the large matrix $G$ can be used to store in other data formats to reduce the block memory usage. Thus, the proposed
architecture is also designed with a format converter (as shown in Fig. 15) to convert the data of matrix $G$ to 32-bit single-precision floating-point format before being used in the signal reconstruction.

4.3 Results and Analysis

The MIB-based signal reconstruction architecture is implemented on a Xilinx Kintex-7 XC7K325T-FBG900 FPGA (speed grade-2). In this design, the size of the matrix $G$ is $512 \times 512$, 16 floating-point multiplier and adders are used in the Finding Index Unit, and one pair of multiplier and adder is used in the Bypassing Unit.

<table>
<thead>
<tr>
<th>Data Format of Matrix G</th>
<th>8-bit fixed</th>
<th>18-bit fixed</th>
<th>32-bit float</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSP48s</td>
<td>93(11%)</td>
<td>93(11%)</td>
<td>93(11%)</td>
</tr>
<tr>
<td>LUTs</td>
<td>88.9K(44%)</td>
<td>90.5K(44%)</td>
<td>92.4K(45%)</td>
</tr>
<tr>
<td>Block RAMs</td>
<td>64(14%)</td>
<td>64(14%)</td>
<td>128(29%)</td>
</tr>
</tbody>
</table>

The matrix $G$ is stored in the block RAMs. The proposed architecture supports the following data formats of $G$: (1) 18-bit fixed-point numbers or 32-bit floating-point numbers when all the elements in $G$ are rational numbers (e.g., elements of $G$ are all rational numbers when $\Phi$ follows the Gaussian distribution or the applied basis $\Psi$ consists of rational numbers such as DCT, see equation (1)); and (2) 8-bit fixed-point numbers when all the elements in $G$ are integers (e.g.,
elements of $G$ are all integer numbers when $\Phi$ follows the Bernoulli distribution and the transformation basis is not needed). Table 6 summarizes the resource utilization under these formats with a system frequency of 87.3MHz. Most LUT resources are used for the configurable parallel computing and temporary data storage. As each block RAM of the FPGA board consists of two 18Kb banks that can be used independently, the implementation under the 32-bit floating-point format needs 128 block RAMs, whereas those under the 8-bit and 18-bit fixed-point formats use 64 block RAMs. The symmetry of $G$ has been exploited to reduce the use of block RAMs by half. Note that the proposed design utilizes a small amount of the hardware resources. This allows other related functions to be easily implemented on the same FPGA. Except for the number of block memories, the implementation under the three data formats of matrix $G$ consumes almost the same resources as the differences only lie in the data format converter and memory I/O interfacing circuits, while all internal computations are based on the 32-bit floating-point format.

4.3.1 Complexity Comparison

The complexity of the proposed MIB-based implementation is first compared with the conventional OMP implementation. In both implementations, the matrix $G$ and vector $\Phi Y$ are stored in the memory. The number of arithmetic operations (multiplications and additions) needed to update $\alpha_k$ for new index selection in the $k^{th}$ iteration is summarized in Table 7 and Table 8. By directly
updating $\alpha_k$ using the proposed MIB method, the complexity of finding the new index is reduced by $\frac{1}{2}(3k^2 - k - 2)$ multiplications and $\frac{3}{2}(k^2 - k)$ additions in the $k^{th}$ iteration. Combining the finding index and bypassing operations together, the proposed architecture requires $(k^2 - k - 3)$ fewer multiplications and $(k^2 - k)$ fewer additions in the $k^{th}$ iteration as compared with the conventional OMP. Depending on the specific end use or application, this reduction in complexity can be leveraged for either high-speed (as in this chapter) or low-power purposes.

Table 7: Arithmetic operations to update $\alpha_k$ in the $k^{th}$ iteration

<table>
<thead>
<tr>
<th>Result</th>
<th>Number of Multiplications</th>
<th>Number of Additions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$</td>
<td>$k^2 - 2k + 1$</td>
<td>$k^2 - 3k + 2$</td>
</tr>
<tr>
<td>$V$</td>
<td>$k - 1$</td>
<td>$k - 1$</td>
</tr>
<tr>
<td>$VA$</td>
<td>$k - 1$</td>
<td>$0$</td>
</tr>
<tr>
<td>$G_{I_k,I_k}^{-1}$</td>
<td>$\frac{1}{2}k^2 - \frac{1}{2}k$</td>
<td>$\frac{1}{2}k^2 - \frac{1}{2}k$</td>
</tr>
<tr>
<td>$\hat{X}_k$</td>
<td>$k^2$</td>
<td>$k^2 - k$</td>
</tr>
<tr>
<td>$\alpha_k$</td>
<td>$Nk - k^2$</td>
<td>$Nk - k^2$</td>
</tr>
</tbody>
</table>

Figure 23 compares the arithmetic operations of updating the vector $\alpha_k$ under two signal lengths $N = 256$ and 512. In each case, the numbers of multiplications and additions are approximately the same as the computations mainly consist of vector and matrix multiplications. Although the arithmetic operations increase with the iteration number, the proposed design increases at a slower rate. This
Table 8: Arithmetic operations of the MIB technique to update $\alpha_k$ in the $k^{th}$ iteration

<table>
<thead>
<tr>
<th>Finding Index Operation</th>
<th>Bypassing Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Result</td>
<td>Number of Multiplications</td>
</tr>
<tr>
<td>$A$</td>
<td>$k^2 - 2k + 1$</td>
</tr>
<tr>
<td>$[V, M]$</td>
<td>$2k - 2$</td>
</tr>
<tr>
<td>$-VM$</td>
<td>$1$</td>
</tr>
<tr>
<td>$VMA$</td>
<td>$k - 1$</td>
</tr>
<tr>
<td>$\alpha_k$</td>
<td>$Nk - k^2$</td>
</tr>
</tbody>
</table>

indicates that the efficiency of signal reconstruction, especially for signals with a large sparsity $K$, is greatly improved.

4.3.2 Performance Evaluation

The execution time of the Finding Index Unit and the Bypassing Unit are measured with respect to the iteration number (signal length $N = 512$) are shown in Figure 24. It can be seen that the execute time of the Finding Index Unit increases linearly with the iteration number. For the Bypassing Unit, the execute time increases quadratically and is getting closer to that of the Finding Index Unit as the iteration number increases. These different rates of increase correlate well with Table 7 and Table 8, where the numbers of multiplications and additions of the two units increase linearly and quadratically with the iteration.
Due to the dynamic configuration between the Independent Accumulation Mode and Collective Accumulation Mode in the finding index operation, the real-time utilization of the multiplier and adder resources is close to 100% during all the iterations. This along with the matrix inversion bypass significantly

Figure 23: Complexity of updating $\alpha$ under the signal length $N = 256$ and 512.

number $k$, respectively. For typical applications requiring a relative small number of iterations, the finding index operation is expected to be the dominant operation of the signal recovery. Thus, the total execute time of the signal recovery increases almost linearly with the iteration number. Note that it is possible to reduce the gap between the Finding Index Unit and Bypassing Unit by reallocating the hardware resources on the FPGA platform.
Figure 24: Timing breakdown of the finding index and bypassing operations.

improves the speed of signal recovery. Figure 25 shows the speed improvement over the conventional OMP-based FPGA implementation. In the OMP-based implementation, the 16 floating-point multipliers and adders from the Finding Index Unit of the MIB implementation is used for both finding index and matrix inversion operations. The proposed implementation uses only about 6% more hardware resources due to the parallel processing; however, it improves the speed of signal reconstruction by up to 40%.

Finally, to show the effect of the different formats of the matrix $G$ on the accuracy of signal recovery, the FPGA results are compared with the results obtained from MATLAB (double precision format). The root mean square errors
Figure 25: Speedup ratio of the proposed MIB architecture over the OMP implementation.

(RMSE) between the two sets of results are used to quantify the accuracy. As the proposed implementation adopts the floating-point data format, the level of errors introduced by the FPGA design is very low. For the matrix $G$ in the 32-bit floating-point format and 8-bit fixed-point format, the RMSE is at the level of $10^{-7}$, whereas for $G$ in the 18-bit fixed-point number, the RMSE increases to $10^{-5}$. This increase for the 18-bit fixed-point data format is mainly due to the quantization errors of converting rational numbers to 18-bit fixed-point numbers. This introduces additional signal reconstruction errors. On the other hand, the quantization errors of using 32-bit floating-point format for rational numbers are
much smaller, and there is no quantization error when using 8-bit fixed-point format to represent integer numbers. It can be seen that the proposed FPGA design achieves almost the same level of the accuracy as the software-based implementation (e.g. MATLAB), even when the matrix $G$ uses the 18-bit fixed-point format. This makes the proposed design a promising solution when the block RAM resources are limited in FPGA implementations.

![Figure 26: The SNR of the recovered signals under different measurement ratios.](image)

**4.3.3 Comparisons With the Existing Work**

The proposed design is compared with some existing FPGA implementations for the OMP algorithm [81, 89, 90]. The results are summarized in Table 9.

As 32-bit floating-point for internal computations is used in this design, more complicated data-path is needed for dynamical configuration using shared multipliers and adders, and more LUT slices are also needed. Thus, the maximum
Figure 27: The original and recovered audio signals under the measurement ratio of 0.5.

frequency is lower than the existing works. However, through the proposed Matrix Inversion Bypass (MIB) transform, the complexity of signal reconstruction can be significantly reduced. In addition, the proposed work is more flexible and able to support different levels of signal sparsity and measurements rather than being limited to one specific setting as in some existing works.

Although far fewer multipliers and adders are used in this design than all three existing works, the reconstruction time of 36 iterations measured in the proposed design is still smaller due to the following reasons:

- The proposed MIB technique enabling parallel computations;
Table 9: Comparison Implementation Results

<table>
<thead>
<tr>
<th>Implementation Design</th>
<th>[89]</th>
<th>[90]</th>
<th>[81]</th>
<th>This Work</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computation Precision</td>
<td>fixed point: 18-bit inputs 42-bit intermediate 18-bit outputs</td>
<td>floating point: 18-bit</td>
<td>floating point: 32-bit</td>
<td>floating point: 32-bit</td>
</tr>
<tr>
<td>Supported Maximum Sparsity K</td>
<td>40</td>
<td>36</td>
<td>320</td>
<td>Configurable</td>
</tr>
<tr>
<td>Signal Size N</td>
<td>1024</td>
<td>1024</td>
<td>up to 1740</td>
<td>512</td>
</tr>
<tr>
<td>Number of Measurements M</td>
<td>256</td>
<td>256</td>
<td>up to 640</td>
<td>Configurable</td>
</tr>
<tr>
<td>MAX Frequency: (MHz)</td>
<td>100</td>
<td>120</td>
<td>53.7</td>
<td>87</td>
</tr>
<tr>
<td>Reconstruction time of 36 iterations (µs)</td>
<td>630</td>
<td>340</td>
<td>about 1436</td>
<td>250</td>
</tr>
<tr>
<td>Block RAMs</td>
<td>258</td>
<td>576</td>
<td>382</td>
<td>64 or 128</td>
</tr>
<tr>
<td>LUT Slices</td>
<td>$32K$</td>
<td>$6K$</td>
<td>$18K$</td>
<td>$\sim 92K$</td>
</tr>
</tbody>
</table>

- More efficient and configurable utilization of hardware resources.

4.3.4 Case Study: Audio Signal Processing

The MIB-based FPGA implementation of signal reconstruction is applied to a distributed sensor network, where a sound source is surrounded by multiple sensors to form an audio detection system. The model of this system is given by [91]

$$y_p(t) = \Phi(\alpha_p x(t)) + n(t)$$

(30)
where Φ is the measurement matrix which is a product of the discrete cosine transform (DCT) matrix \([92]\) and Gaussian matrix (see equation (1)), \(\alpha_p\) is the attenuation coefficient at the \(p^{th}\) sensor and \(n(t)\) is the noise. For the sake of simplicity, the signal reflection and phase delay are not considered.

Signals sampled at the frequency of 44.1\(KHz\) are used and the Gaussian noise \(n(t)\) is also introduced with the standard deviation \(\sigma_n = 0.02\). Different attenuation coefficients are applied to each set of the received signals. After that, the audio signals are segmented into the frames of \(N = 512\) samples for signal recovery. In the signal reconstruction, the sparsity is assumed to be 40 for each set of the signal. Five sets of the signals are recovered by the proposed FPGA implementations and the weighted average of the recovered signals is obtained. Since the recovery performance under the above mentioned formats of matrix \(G\) are quite similar, only the results of the 32-bit floating-point format implementation are presented here. The SNRs of the recovered signals under different measurement ratios \(M/N\) are shown in Figure 26, where \(M\) is number of measurements and \(N\) is the length of original signal. As the number of measurements increases, the quality of signal reconstruction shows a great improvement. The recovered signal of about 14 seconds at the measurement ratio of 0.5 is shown in Figure 27. When the sparsity of the audio signals is 40, the signal reconstruction performed by the FPGA takes only about 1.5 ms to recover the total 5 sets of audio signals. This allows the compressed signals to be recovered in real time because each set of the compressed samples is obtained every 11.61 ms. This example
clearly demonstrates the advantages of the proposed architecture for high-speed applications.

4.4 Summary

In this chapter, an FPGA-based implementation of the proposed algorithmic transformation referred to as Matrix Inversion Bypass (MIB) is presented for the OMP-based compressively sampled signal recovery. The proposed MIB technique reduces the computational complexity and improves the signal recovery speed by performing the signal estimation \( \hat{X}_k \), matrix inversion \( G^{-1}_{I_k,I_k} \), and the finding index operation in parallel. An FPGA-based architecture is realized for high-speed signal recovery by efficiently utilizing the hardware resources through system-level optimization. Without introducing large hardware overheads, the proposed implementation improves the speed of signal recovery by up to 1.4× while maintaining the same level of algorithmic performance. A higher speed-up ratio can be achieved in the less sparse signals. Further work is directed towards further improving the hardware utilization efficiency as well as the data pipelining efficiency, and applying the proposed MIB technique and architecture to complex valued systems in which complex sparsity basis is needed, such as channel estimation and radar imaging.
Chapter 5

Soft-thresholding OMP Algorithm and Implementation

5.1 Overview

It is already known that the Orthogonal Matching Pursuit (OMP) based greedy algorithms are preferable choices in hardware implementations of compressed sensing signal recovery for its efficiency and simplicity in geometric interpolation. However, the computation complexity is still formidable limiting the practical applications where the energy is the critical factor. This chapter will make an observation and deduction to the fact that more significant elements of the signal are likely to be recovered first in the iterative OMP algorithm. Based on this, a Soft-thresholding Orthogonal Matching Pursuit (ST-OMP) technique
is proposed for efficient signal reconstruction in compressive sensing applications. The proposed ST-OMP recovers less significant signal elements using a low-complexity procedure without much degradation in reconstruction quality. The proposed ST-OMP technique is applied in systems powered by non-deterministic renewable energy sources. The threshold of employing the efficient reconstruction is made dynamically adjustable according to the performance requirements and energy levels. Simulation results demonstrate that the ST-OMP can achieve good recovery performance while significantly reducing the energy consumption as compared to the original OMP implementation.

The rest of this chapter is organized as follows. In Section 5.2, the Soft-thresholding OMP technique is developed for efficient signal reconstruction of compressive sensing. The simulation results of the proposed technique are presented in Section 5.3 and the summary is given in Section 5.4.

5.2 The proposed Soft-thresholding OMP

5.2.1 Motivation

In the iterative signal recovery of the OMP algorithm, it is interesting to observe that the OMP algorithm is likely to first recover the non-zero elements with larger magnitude. To illustrate this observation, an experiment is run to recover a sparse signal with non-zero elements of values randomly chosen from 1 to 100. To obtain the statistical results of recovery accuracy, the experiments are repeated 1000 times with the measurement matrix $\Phi$ (normalized Bernoulli
distribution) generated at each time. Results in Figure 28 show how the signal is recovered with respect to iteration order. It can be clearly seen that elements with larger values are likely to be recovered earlier. For example, elements with a value of 90 are mostly recovered before 20\textsuperscript{th} iteration, whereas elements with a value of 10 are most possibly recovered after 40\textsuperscript{th} iteration. The observation leads to the following proposition and the corresponding proof.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure28.png}
\caption{Statistical results of signal recovery pattern.}
\end{figure}

**Lemma 1**: Define \( \Lambda \) as index set of non-zero elements of the sparse signal to recover. If all the \( K \) non-zero elements of the sparse signal are recovered in the
first $K$ iterations of OMP algorithm, which holds that,

$$I_K = \Lambda,$$

(31)

then the non-zero elements of which the indices first added into the index set $I_K$ are likely to be of larger magnitude.

**Proof:** As the OMP algorithm adds one new element into the index set $I_k$ based on $\alpha_k = \Phi^* R_k$ at each iteration, the assumption of being able to locate all $K$ non-zero entries in signal $X$ means that the element with the maximum absolute value in $\alpha_k$ matches one of the $K$ non-zero elements at the first $K$ iterations, which establishes that, for $k = 1, ..., K$,

$$\text{arg max}_{i=1,...,N}(|\Phi_i^* R_k|) \in \Lambda$$

(32)

where $\Lambda$ is denoted as the index set of the $K$ non-zero elements in signal $X$, and $R_k$ is the residual vector updated at each iteration (see Section 5.1). In other words, it is assumed here that it holds that $I_K = \Lambda$ after $K$ iterations of signal recovery in OMP algorithm. To simplify the proof, the following proof only talks about the elements in vector $\alpha_k = \Phi^* R_k$ for which the indices that will be added to the index set $I_k$ at the first $K$ iterations, which can be presented as the set $\{\Phi_i^* R_k : i \in \Lambda\}$. In addition, the measurement noise is first neglected.
In the beginning of OMP algorithm, \( R_k = Y = \Phi X \) and \( \forall i \in \Lambda \), it follows that,

\[
\Phi_i^* R_k = \Phi_i^* \Phi X \\
= \Phi_i^* \Phi_{\Lambda} X_{\Lambda} \\
= \Phi_i^* \Phi_i X_i + \Phi_i^* \Phi_{\Lambda-i} X_{\Lambda-i} \\
= \Phi_i^* \Phi_i X_i + \sum_{j \in \Lambda - i} \Phi_i^* \Phi_j X_j
\]

where the second equation is based on \( X_i \neq 0 \) iff \( i \in \Lambda \), i.e. the element in signal \( X \) is non-zero only when the index for the element is in the index set \( \Lambda \). For simplicity, the Normalized Bernoulli distribution (see section 5.1) is adopted for matrix \( \Phi \), then the dot product of any column in the matrix by itself is strictly equal to 1 and the equation above is as follows,

\[
\Phi_i^* R_k = X_i + \sum_{j \in \Lambda - i} \Phi_i^* \Phi_j X_j
\]

After the first iteration, \( R_k = Y - \Phi_{I_k} \hat{X}_k \) (see Section 5.1) and \( \forall i \in \Lambda \), it follows that,

\[
\Phi_i^* R_k = \Phi_i^* Y - \Phi_i^* \Phi_{I_k} \hat{X}_k \\
= \Phi_i^* \Phi X - \Phi_i^* \Phi_{I_k} \Phi_{I_k}^\dagger \Phi X,
\]

where the second term in the equation is based on equation (15). Then, it establishes that in the vector \( \alpha_k = \Phi^* R_k \), the elements already chosen in the previous iterations, i.e. the elements already added into the index set \( I_k \) are as
follows,

\[
\Phi_{I_k}^* R_k = \Phi_{I_k}^* R_k \\
= \Phi_{I_k}^* \Phi X - \Phi_{I_k}^* \Phi I_k (\Phi_{I_k}^* \Phi I_k)^{-1} \Phi_{I_k}^* \Phi X \\
= \Phi_{I_k}^* \Phi X - \Phi_{I_k}^* \Phi X \\
= 0
\] (36)

As a result, indices already added in index set \(I_k\) will not be picked up again, and in cases of the noise existing, the index selection can still be made deterministically without selecting the chosen indices. Then, only the unchosen elements \(\{\Phi_{I_k}^* r_k : i \in \Lambda - I_k\}\) are considered, and \(\forall i \in \Lambda - I_k\),

\[
\Phi_i^* R_k = \Phi_i^* \Phi X - \Phi_i^* \Phi I_k \Phi_{I_k}^\dagger \Phi X \\
= X_i + \sum_{j \in \Lambda - i} (\Phi_i^* \Phi)_{i,j} X_j - \Phi_i^* \Phi I_k \Phi_{I_k}^\dagger \Phi X.
\] (37)

As \(i\) is not in the index set \(I_k\), for the third term \(\Phi_i^* \Phi I_k \Phi_{I_k}^\dagger \Phi X\) of the above equation,

\[
|\Phi_i^* \Phi I_k \Phi_{I_k}^\dagger \Phi X| \leq \mu(\Phi) \sum_{j=1}^k |(\Phi_{I_k}^\dagger \Phi X)_j| \\
\leq \mu(\Phi) \| (\Phi_{I_k}^\dagger \Phi X) \|_1
\] (38)

where the inequality is based on the definition of the coherence parameter \(\mu(\Phi)\) which is the upper bound of inner products among distinct columns of matrix \(\Phi\) (see equation (7)); \(\| (\Phi_{I_k}^\dagger \Phi X) \|_1\) is the the \(\ell_1\)-norm of the \(k\)-entry vector \((\Phi_{I_k}^\dagger \Phi X)\).

In the above equation, the upper bound for \(\| (\Phi_{I_k}^\dagger \Phi X) \|_1\) is given in [39] and can be regarded as fairly small as reliable signal recovery is assumed here; the coherence parameter \(\mu(\Phi)\) is also known to be a small number as discussed in Section 5.1.
Based on the above properties, $\Phi_i^* \Phi_{I_k} \Phi_{I_k}^* \Phi X$ is likely to be of smaller magnitude compared to other terms in equation (37).

Recall equation (33), (34) and (37) and incorporate the measurement noise $e$ (see equation (1)), and $\forall i \in \Lambda - I_k$, it follows that,

$$\Phi_i^* R_k \approx X_i + \sum_{j \in \Lambda - i} (\Phi_i^* \Phi)_{i,j} X_j + \Phi_i^* e$$  

(39)

where $k = 0, \ldots, K - 1$.

As the magnitude of $(\Phi_i^* \Phi)_{i,j}$ here is known to be equal to or smaller than $\mu(\Phi)$, which is a fairly small number (see equation (10)); also, the term $\Phi_i^* e$ is likely to be of small magnitude with $\Phi_i^*$ usually a normalized random vector (See Chapter 2). Then, if $|X_i|$ is larger, it is more probable that $|\Phi_i^* R_k|$ is larger. Similar results can hold for other admissible random matrix like normalized Gaussian matrix or sparsity basis $\Psi$ is applied since the coherence parameter $\mu(\Phi)$ or $\mu(\Phi \Psi)$ is also a small number, but it is messier when the norms of its columns are not identical. A detailed discussion is omitted for simplicity.

Therefore, for the OMP algorithm, which is assumed to be reliable to locate all $K$ non-zero elements, each iteration is more likely to recover the non-zero element with larger weight among the unrecovered non-zero elements. This completes the proof. \hfill \Box

On the other hand, as iteration order goes up [32, 33, 57], the OMP algorithm still suffers from significantly increasing computational complexity despite
relatively low complexity of hardware implementation. The reason is that updating $\hat{X}_k$, $R_k$ and $\alpha_k$ needs more operations regarding the matrix operation as iteration order $k$ increases with the size of $I_k$, $\hat{X}_k$ and $R_k$ increasing. Then, the computation complexity of the OMP implementation is considered in term of energy overhead. As done in [32], $\Phi$ and its Gramian matrix $\Phi^*\Phi$ are assumed to be pre-defined and stored. Also, $\Phi^*Y$ obtained in the initialization is stored to facilitate the following iterations and use the Matrix Inversion Lemma [80] to invert the matrix $\Phi^*_I\Phi_I$.

In the $k^{th}$ iteration of OMP algorithm, the dominant operations are the inner products among the computations of $(\Phi^*_I\Phi_I)^{-1}$, $\hat{X}_k$ and $(\Phi^*\Phi_I)\hat{X}_k$. The number and length of the inner products in the corresponding computations is shown in Table 10.

<table>
<thead>
<tr>
<th>Computation</th>
<th>Number of inner products</th>
<th>Length of the vector</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(\Phi^*_I\Phi_I)^{-1}$</td>
<td>$k$</td>
<td>$k - 1$</td>
</tr>
<tr>
<td>$\hat{X}_k$</td>
<td>1</td>
<td>$k$</td>
</tr>
<tr>
<td>$(\Phi^*\Phi_I)\hat{X}_k$</td>
<td>$N - k$</td>
<td>$k$</td>
</tr>
</tbody>
</table>

In term of energy consumption, the energy consumed by the operations of computing the inner products of two vectors is assumed to linearly increase with the length of the vector increasing and then one inner product of two n-length vectors is assumed to require the energy of $nE_m$. With the energy consumption
of other operations like updating the index set $I_k$ and computing the inverse of a number at each iteration neglected for simplicity, the total amount of energy needed to finish the computation of $K$ iterations of OMP algorithm is about,

$$E_{OMP}(K) \approx \left( \frac{2}{3} K^3 + \frac{N + 1}{2} K^2 + \frac{3N - 1}{6} K \right) E_m. \quad (40)$$

From the table and equation above, it can be seen that the total energy overhead of signal recovery using OMP algorithm is increasing significantly with larger $K$ (more non-zero elements need to be recovered) and more energy overhead is expected in the last rounds of iteration in the OMP algorithm. If energy source is insufficient, the last rounds of signal reconstruction are likely to cause the failure of recovering the whole signals. In addition, reducing the energy overhead of signal recovery while being able to obtain the needed information allows renewable energy-powered devices to perform other tasks more frequently and reliably [93, 94, 95].

### 5.2.2 Soft-thresholding OMP

Based on the above observation, the Soft-Thresholding OMP (ST-OMP) is proposed for efficient signal recovery. Note that the proposed ST-OMP is a general approach that can be utilized to address various resource constraints. This chapter will focus on improving energy efficiency due that energy is a major resource constraint in many embedded systems.

Consider a scenario where the signal reconstruction needs to be finished with limited energy supply. A challenging problem is that available energy may be
dynamically changing if the computation is powered by renewable energy sources [96, 97, 98], in particular when available energy is insufficient to support signal reconstruction using OMP. The proposed ST-OMP is very effective to deal with this challenging problem.

At the beginning of each signal reconstruction task, the ST-OMP recovers one non-zero element in the signal $X$ at each iteration using the same procedure in the OMP algorithm. Assume the signal to recover is $K$-sparse and its $(K - L)$ non-zero elements can be recovered given a certain renewable energy level. After that, the energy becomes insufficient to support the remaining iterations. Note that the energy consumption of each iteration increases with the order of iterations because the last iterations involve more computations as discussed in Section 5.2.1. The proposed ST-OMP switches to a low-complexity recovery algorithm, where iteration threshold $L$ for making this switch is determined by the available energy, i.e., it is adjustable in accordance with renewable energy while trying to achieve the highest reconstruction quality as possible. This is expressed formally as

$$L = \arg \min_l (E_{OMP}(K - l) + \Delta E(l) \leq E_{avl}),$$

where $E_{OMP}(k-l)$ is the amount of energy consumed in the first $k$ rounds of iteration, $\Delta E(l)$ is the expected energy consumption of the low-complexity algorithm, and $E_{avl}$ is the available energy at the beginning of each signal reconstruction.
task. Note that $E_{avt}$ is a time-dependent variable as in the case of renewable energy sources.

Once $(K - L)$ determined in equation (41) is reached, the value of latest recovered element and $\alpha$ from the $(K - L)^{th}$ iteration are stored,

$$
\begin{cases} 
\hat{X}_{temp} = \hat{X}_{K-L}, \\
\alpha_{temp} = \alpha_{K-L}.
\end{cases}
$$

Signal reconstruction is then switched to a low-complexity algorithm. The $L$ largest elements (absolute values) in vector $\alpha_{temp}$ are chosen and the corresponding coordinates are added into the index set $I_{K-L}$. In the meantime, the corresponding entries in the estimated signal are assigned with the value of the latest recovered $\hat{X}_{temp}$ as Algorithm 3 shows.

<table>
<thead>
<tr>
<th>Initialize: $\text{count} = 1$; $\alpha = \alpha_{temp}$;</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>while</strong> $\text{count} \leq L$ <strong>do</strong></td>
</tr>
<tr>
<td>3.1 $i_k = \text{arg max}_{j \notin I}(</td>
</tr>
<tr>
<td>3.2 $I_{K-L+\text{count}} = I_{K-L+\text{count}-1} \cup i_k$</td>
</tr>
<tr>
<td>3.3 $\hat{X}(K - L + \text{count}) = \hat{X}_{temp}$</td>
</tr>
<tr>
<td>3.4 $\text{count} = \text{count} + 1$</td>
</tr>
</tbody>
</table>

**Algorithm 3:** The low-complexity ST-OMP reconstruction algorithm.

By simply doing so, signal reconstruction avoids the most energy-consuming operation of multiplication between matrix $\Phi^* \in R^{N \times M}$ and vector $R_k \in R^M$ to update $\alpha_k$. Furthermore, there is no need to keep updating $\hat{X}_k \in R^k$, which involves the inversion operations and multiplication between matrix $(\Phi_{I_k}^* \Phi_{I_k})^{-1} \in R^{k \times k}$ and vector $\Phi_{I_k}^* Y \in R^k$.
Furthermore, there exists one special case where ST-OMP is applied to recover signals with the value of non-zero elements known to be equal. Then, the ST-OMP algorithm can take advantage of such signals and directly assign $\hat{X}(K)$ with the known value. Also, once iteration threshold $(K - L)$ determined in equation (41) is reached, vector $\alpha$ from the $(K - L)^{th}$ iteration is stored,

$$\alpha_{\text{temp}} = \alpha_{K-L}. \quad (43)$$

Signal reconstruction is then switched to a low-complexity algorithm. Similarly, coordinates of the $L$ largest elements (absolute values) in vector $\alpha_{\text{temp}}$ are added into the index set and the corresponding entries in the estimated signal are identified as non-zero elements.

### 5.2.3 Convergence study of the Soft-thresholding OMP

In this subsection, the convergence of the ST-OMP algorithm is analyzed studying the requirement on Restricted Isometry Constant (RIC) $\delta$ that suffices for exact support recovery of K-sparse signals. Here, OMP algorithm is assumed to be reliable to locate all $K$ non-zero entries under both the noisy and noiseless measurements where ST-OMP algorithm is applied (see Chapter 2).

Denote $\Lambda$ as index set of the $K$ non-zero entries in the signal $X$. Before the low-complexity algorithm of ST-OMP begins, the index set $I_{K-L}$ consists of $K - L$ indices. As OMP algorithm is assumed reliable to locate all $K$ non-zero entries, it holds that $I_{K-L} \subset \Lambda$ and $I_{K-L}^C = \Lambda - I_{K-L} \in \mathbb{R}^L$ is denoted as the non-zero entries not located yet. Then, the low-complexity algorithm will select the
coordinates of the $L$ largest elements in $|\alpha_{K-L}| = |\Phi^* R_{K-L}|$ to finish the index set estimation. Therefore, in order to accurately locate those remaining non-zero elements ($I_{K-L}$ will not be picked up again as Section 5.2.1 shows), the following inequality needs to hold, $\forall i \in I_L^C$ and $\forall j \notin \Lambda$,

$$|\Phi^*_i R_{K-L}| > |\Phi^*_j R_{K-L}|$$  \hspace{1cm} (44)

Assuming the noiseless measurement $Y = \Phi X$, it holds that,

$$Y = \Phi_{I_{K-L}} X_{I_{K-L}} + \Phi_{I_L^C} X_{I_L^C},$$  \hspace{1cm} (45)

then the estimate vector $\hat{X}_{K-L}$ of original signal (see equation (15)), can be as follows,

$$\hat{X}_{K-L} = (\Phi^*_{I_{K-L}} \Phi_{I_{K-L}})^{-1} \Phi^*_{I_{K-L}} Y$$

$$= X_{I_{K-L}} + \Phi_{I_{K-L}}^\dagger \Phi_{I_{K-L}} I_L^C X_{I_L^C},$$  \hspace{1cm} (46)

and then the residual vector $R_{K-L}$ holds that,

$$R_{K-L} = Y - \Phi_{I_{K-L}} \hat{X}_{K-L}$$

$$= \Phi_{I_L^C} X_{I_L^C} - \Phi_{I_{K-L}} \Phi_{I_{K-L}}^\dagger \Phi_{I_L^C} X_{I_L^C}.$$  \hspace{1cm} (47)

As a result, the right term $|\Phi^*_j R_{K-L}|$ in equation (44) follows,

$$|\Phi^*_j R_{K-L}| = |\Phi^*_j \Phi_{I_L^C} X_{I_L^C} - \Phi^*_j \Phi_{I_{K-L}} \Phi_{I_{K-L}}^\dagger \Phi_{I_L^C} X_{I_L^C}|$$

$$\leq \|\Phi^*_j \Phi_{I_L^C} X_{I_L^C}\|_2 + \|\Phi^*_j \Phi_{I_{K-L}} \Phi_{I_{K-L}}^\dagger \Phi_{I_L^C} X_{I_L^C}\|_2$$

$$\leq \delta_{L+1} \|X_{I_L^C}\|_2 + \delta_{K-L+1} \|\Phi_{I_{K-L}}^\dagger \Phi_{I_L^C} X_{I_L^C}\|_2$$

$$\leq \delta_{L+1} \|X_{I_L^C}\|_2 + \frac{\delta_{K-L+1} \delta_K}{1 - \delta_{K-L}} \|X_{I_L^C}\|_2$$  \hspace{1cm} (48)

where the first inequality is based on triangular inequality and the rest are based on equation (15) and the following properties [56, 99]: Let $\Gamma$ and $\Upsilon$ be two index
sets, where $\Gamma, \Upsilon \subset \{1, \ldots, N\}$, $\Gamma \cap \Upsilon = \emptyset$, $b \in R^{\|\Gamma\|_0}$, the followings hold,

$$\|\Phi_\Gamma^* \Phi_\Upsilon b\|_2 \leq \delta \|\Phi_\Gamma^* \Phi_\Upsilon b\|_2$$  \hspace{2cm} (49)

$$\|((\Phi_\Gamma^* \Phi_\Upsilon)^{-1}) b\|_2 \leq \frac{\|b\|_2}{1 - \delta \|\Gamma\|_0}$$  \hspace{2cm} (50)

where $\|\cdot\|_0$ are the $\ell_0$-norm of a vector, i.e. the number of non-zero elements in the vector.

Similarly, the left term $|\Phi_i^* R_{K-L}|$ in equation (44) is,

$$|\Phi_i^* R_{K-L}| = |\Phi_i^* \Phi_i^{\text{IC}} X_i^{\text{IC}} - \Phi_i^* \Phi_i^{\text{IC}} \Phi_i^{\text{IC}} X_i^{\text{IC}}|$$

$$\geq |\Phi_i^* \Phi_i^{\text{IC}} X_i^{\text{IC}}| - \|\Phi_i^* \Phi_i^{\text{IC}} - X_i^{\text{IC}}\|_2$$

$$\geq |\Phi_i^* \Phi_i^{\text{IC}} X_i^{\text{IC}}| - \delta L \|X_i^{\text{IC}}\|_2 - \frac{\delta_{K-L+1} \delta K}{1 - \delta_{K-L}} \|X_i^{\text{IC}}\|_2$$

where the first three inequalities are obtained similar to equation (48), and the last is from the assumption that the $\ell_2$-norm of the columns in matrix $\Phi$ is strictly equal to 1 by adopting the Normalized Bernoulli distribution for simplicity.

Assume $|x_*|$ is the minimum value of all unrecovered non-zero elements in magnitude and it follows that,

$$\|X_i^{\text{IC}}\|_2 \leq \eta |x_*|.$$  \hspace{2cm} (52)

where the $\eta$ is the smallest coefficient for which the above inequality holds. As the monotonicity of RIC [23] stands for any integers $K \leq K'$,

$$\delta_K \leq \delta_{K'}.$$  \hspace{2cm} (53)
it is easy to know \( \max\{\delta_L, \delta_{L+1}, \delta_K, \delta_{K-L+1}\} \leq \delta_K \) always holds (1 \(<\ L \(<\ K \(<\ 1 \) in ST-OMP algorithm). Then, for the two terms in equation (44), it can be reached that \( \forall i \in I^C_L, \forall j \notin \Lambda \)

\[
|\Phi^*_i R_{K-L}| \geq |x_*| - \delta_K \eta |x_*| - \frac{\delta^2_K}{1-\delta_K} \eta |x_*|,
\]

\[
|\Phi^*_j R_{K-L}| \leq \delta_K \eta |x_*| + \frac{\delta^2_K}{1-\delta_K} \eta |x_*|.
\]

Including the case of the noisy measurement \( Y = \Phi X + e \), the above inequalities follows that,

\[
|\Phi^*_i R_{K-L}| \geq |x_*| - \delta_K \eta |x_*| - \frac{\delta^2_K}{1-\delta_K} \eta |x_*| - |\Phi^*_i (I - \Phi_{I_{K-L}} \Phi_{I_{K-L}}^\dagger) e|,
\]

\[
|\Phi^*_j R_{K-L}| \leq \delta_K \eta |x_*| + \frac{\delta^2_K}{1-\delta_K} \eta |x_*| + |\Phi^*_j (I - \Phi_{I_{K-L}} \Phi_{I_{K-L}}^\dagger) e|.
\]

where the third term in the two inequalities is based on the error \( e \) from the vector \( Y \) and \( \tilde{X}_{K-L} \). Use the fact [100] that \( \|(I - \Phi_{I_{K-L}} \Phi_{I_{K-L}}^\dagger) e\|_2 \leq \|e\|_2 \) and assume that

\[
\|e\|_2 \leq \frac{\epsilon \delta_K}{(1 - \delta_K) \sqrt{1 + \delta_1}} \|X^C_{I_L}\|_2,
\]

where \( \epsilon \) is the smallest coefficient for the inequality to hold. Then, the inequalities in equation (55) can be further transformed as,

\[
|\Phi^*_i R_{K-L}| \geq |x_*| - \delta_K \eta |x_*| - \frac{\delta^2_K}{1-\delta_K} \eta |x_*| - \frac{\delta \delta_K \eta}{1-\delta_K} |x_*|,
\]

\[
|\Phi^*_j R_{K-L}| \leq \delta_K \eta |x_*| + \frac{\delta^2_K}{1-\delta_K} \eta |x_*| + \frac{\delta \delta_K \eta}{1-\delta_K} |x_*|.
\]

where the last term is based on equation (52) and the following property [56, 99]:

Let \( \Gamma \subset \{1, ..., N\} \) be an index set, \( b \in \mathbb{R}^{||\Gamma||_0} \), it holds that,

\[
\|\Phi^*_i b\|_2 \leq \sqrt{1 + \delta_{||\Gamma||_0}} \|b\|_2
\]
Combine the equation (54) and (57), inequality (44) can stand if the following holds,

\[ \delta_K < \frac{1}{1+2(1+\epsilon)\eta}. \]  

(59)

where the coefficients \( \eta \) and \( \epsilon \) are from equation (52) and (56), and \( \epsilon = 0 \) is for noiseless measurement. In other words, the ST-OMP can locate all the non-zero elements if the above inequality holds.

If the absolute value of each entry is closer (\( \eta \) is smaller and closer to \( \sqrt{L} \)) and iteration threshold \( L \) is smaller, the upper limit of \( \delta_K \) is higher making the exact support recovery more possible. Compared to the noiseless situation, the upper limit is further lowered with the introduction of coefficient \( \epsilon \), and lower measurement noise level will lead to higher upper limit on the \( \delta_K \) that is more likely to result in exact support recovery and vice versa.

Again, consider the special case (see Section 5.2.2) that all non-zero elements are equal with the value known (denoted as \( x^* \)), and the residual in the noisy measurement is as follows,

\[ R_{K-L} = \Phi^{I_{L}}_{I_{L}} X_{I_{L}} + e. \]

(60)

Then, assuming \( \|e\|_2 \leq \frac{\delta_L \sqrt{L}}{\sqrt{1+\sigma_1}} |x_*| \), for vector \( |\Phi_j^* R_{K-L}| \subset |\Phi_j^* R_{K-L}| \) with \( \forall j \notin \Lambda \),

\[
|\Phi_j^* R_{K-L}| \leq \|\Phi_j^* \Phi_{I_{L}}^c X_{I_{L}}^c\|_2 + |\Phi_j^* e| \\
\leq \delta_{L+1}\|X\|_2 + \epsilon \delta_L \sqrt{L} |x_*| \\
\leq \delta_{L+1} \sqrt{L} |x_*| + \epsilon \delta_L \sqrt{L} |x_*|. \]

(61)
Also, for vector $|\Phi_i^* R_{K-L}| \subset |\Phi^* R_{K-L}|$ with $\forall i \in \Lambda - I_{K-L}$:

$$|\Phi_i^* R_{K-L}| = |\Phi_i^* \Phi_{I_{L}}^* X_{I_{L}}^*| - |\Phi_i^* e|$$

$$\geq |\Phi_i^* \Phi_i x_i| - \|\Phi_i^* \Phi_{I_{L} - i}^* X_{I_{L} - i}\|_2 - |\Phi_i^* e|$$  \hspace{1cm} (62)

$$\geq |x_*| - \delta_L \sqrt{L} |x_*| - \epsilon \delta_L \sqrt{L} |x_*|.$$

Similarly, it can be known that the proposed ST-OMP algorithm can reliably reconstruct the $K$-sparse flat signals, i.e. the equation (44) stands, if it holds that,

$$\delta_{L+1} < \frac{1}{2(1+\epsilon)\sqrt{L}},$$  \hspace{1cm} (63)

where $\epsilon = 0$ is for noiseless measurement.

5.2.4 Recovery performance of the Soft-thresholding OMP

However, it is easy to anticipate that the low-complexity algorithm of ST-OMP may introduce some error like estimation error of the non-zero elements correctly located or even identifying some zero elements as non-zero.

Denote $I(L) \in R^L$ as the indices obtained in the low-complexity ST-OMP algorithm. Again, the OMP algorithm is assumed to be reliable to locate all $K$ non-zero entries where ST-OMP algorithm is applied. Then it can be known that $I_{K-L} \in \Lambda$ holds but $I(L) \in \Lambda$ may not, and $\Lambda - I(L)$ is denoted as the coordinate set of non-zero not located. Then, the signal recovery error of the ST-OMP algorithm exists in:

- Estimation error of the non-zero elements correctly located: $\|X_{I_{K-L}} - \hat{X}_{K-L}\|^2 + \|X_{\Lambda \cap I(L)} - \hat{X}_{temp}\|^2.$
It is easy to know that the first term $\|X_{IK-L} - \hat{X}_{K-L}\|_2^2$ is converging to zero with $L$ decreasing as more iterations lie in the OMP algorithm and vice versa. The second term is diminished as both $X_{\Lambda \cap I(L)}$ and $\hat{X}_{\text{temp}}$ are usually of small weight with proper choice of $L$ considering Proposition 1.

- Recovery error of the non-zero elements incorrectly located: $\|X_{\Lambda - I(L)}\|_2^2 + \text{length}[\Lambda - I(L)]\hat{X}_{\text{temp}}^2$

Similarly, $\|X_{\Lambda - I(L)}\|_2^2$ and $\hat{X}_{\text{temp}}$ are likely to be of small value if $L$ is properly chosen considering Proposition 1. Also, it can be expected that smaller $L$ may lead to more or even all non-zero elements being located and vice versa.

Therefore, with proper choice of $L$, the recovery quality degradation is weakened to some extent considering that the non-zero elements of more weight are usually recovered with higher accuracy in the first $(K - L)$ iterations. Then, it is natural to think that application of ST-OMP to nearly-flat signals may undermine the merits of the proposed ST-OMP algorithm since the low-complexity algorithm is usually supposed to process less significant signals. However, based on equation (52), the coefficient $\eta$ is smaller and closer to $\sqrt{L}$ if the signals to recover are more flat. Then, it is more probable that equation (59) can hold and thus the signals can be reliably recovered.
Furthermore, with the OMP algorithm able to locate all the $K$ flat non-zero elements, it is interesting to observe that the proposed ST-OMP algorithm can achieve the signal recovery with high precision if taking advantage of the prior information about the known value (assumed to be $\hat{X}_{\text{temp}}$ here).

- To recover signals with non-zero equal-valued elements, estimated signal $\hat{X}_k$ can be directly assigned with the known value (See 5.2.2). So, there is no estimation error if all the non-zero elements can be correctly located and it holds that,

\[
\|X_{I_K - L} - \hat{X}_{\text{temp}}\|^2_2 + \|X_{\Lambda \cap I(L)} - \hat{X}_{\text{temp}}\|^2_2 = 0. \tag{64}
\]

- The signal recovery error mainly lies in if any or how many non-zero elements cannot be found by ST-OMP: $2 \cdot \text{length}[\Lambda - I(L)]\hat{X}_{\text{temp}}^2$. Similarly, it can be expected that setting a proper upper limit for $L$ can limit $\text{length}(\Lambda - I(L))$.

### 5.2.5 Implementation Architecture of the Soft-thresholding OMP

To apply the proposed ST-OMP, the system needs to know the amount of available energy $E_{\text{avl}}$ at the beginning of each signal reconstruction task, and energy consumption related to the signal reconstruction. Existing work [101, 102] have shown that $E_{\text{avl}}$ can be estimated by some prediction algorithms with sufficient accuracy. These algorithms operate at a much lower rate (e.g., once per hour) and thus the energy overhead can be ignored.
On the other hand, $E_{OMP}(K - l)$ and $\triangle E(l)$ in equation (41) will depend upon a specific hardware implementation. Based on equation (40), $E_{OMP}(K - l)$ can be estimated through simulations or pre-operation hardware measures while $\triangle E(l)$ can be ignored compared with $E_{OMP}(K - l)$. Then, the maximum iteration number $(K - L)_{max}$ of OMP algorithm mode can be stored in a look-up table (LUT) for runtime comparison with $E_{avl}$.

In addition, a upper limit $L \leq L_{MAX}$ needs to be enforced for signal reconstruction as more errors may be introduced with larger threshold $L$ as mentioned in Section 5.2.4. In practice, signal reconstruction will fail if $L$ obtained from equation (41) is larger than $L_{MAX}$ due to insufficient renewable energy. This sets up a lower bound on the recovery quality, which can be determined by the specific performance requirement of an application. For example, the proposed ST-OMP algorithm can be used to recover a $N$-length flat signal with the value of non-zero elements assumed unknown, and the simulation results under different configurations (number of measurements $M$, iteration threshold $L$, signal sparsity $K$ and measurement noise $e$) can be easily obtained and helpful to set up the upper limit $L_{MAX}$.

Also, it may exist that the sparsity $K$ or its estimation $\hat{K}$ are unknown or the signals are not exactly sparse (e.g. measurements contaminated by noise). As mentioned in Chapter 2, a halting criterion regarding the dynamic residual $R_k$ can be enforced to control the number of iterations. Then, if the updated index length $k$ reaches $(K - L)_{max}$ and further convergence of error is still required,
\(\alpha_{(K-L)_{\text{max}}} \) can be used to identify multiple possible non-zero elements avoiding the most intensive calculation of further updating \(\alpha_k\) (see Section 5.2.1) rather than failing to reconstruct the signal due to insufficient energy resource.

However, as it is possible to hold or drop the task earlier with signal sparsity \(K\) or its estimation \(\hat{K}\) known, the renewable energy-powered system can avoid unnecessary waste of energy trying but failing to recover the signal. Besides, iterative computation regarding the halting criterion is not needed. So, acquiring the signal sparsity \(K\) or its estimation \(\hat{K}\) at relatively lower energy consumption is encouraged in practical application. For example, an effective method is proposed in [60] that can estimate the spectral support priori to signal sampling and recovery in a fully spectrum-blind system.

As a summary, the implementation architecture for the proposed ST-OMP is shown in Figure 29.

In addition, to further reduce the computation overhead, the OMP mode of the proposed ST-OMP algorithm can be easily replaced by other OMP-based algorithms (i.e., MOMP algorithm [100]) that can reconstruct the signal with high accuracy.

5.3 Evaluation

5.3.1 Numerical Results

This section first evaluates the effect of soft thresholding on the performance of flat signal reconstruction with the value of non-zero element known. The
Figure 29: Implementation architecture of the proposed ST-OMP algorithm.

Experiments were repeated $10^5$ times to obtain the statistical results of recovery accuracy (i.e., identifying the non-zero elements). In Figure 30, sparse signals with different numbers of non-zero elements (60, 80, 100 and 120) randomly assigned to be 1 were recovered using the proposed ST-OMP ($\Phi \in \mathbb{R}^{1024 \times 2048}$) by deliberately changing the value of $L$ from 1 to 59.

As shown in Figure 30, the reconstruction accuracy in the four cases shows almost no degradation when $L$ is smaller than 20. Recall section 5.2.4 that a proper choice of $L$ can ensure the reliability of recovering the flat signals and it is easy to know that $L = 20$ here corresponds to the maximum value of $L$. 
Figure 30: Recovery accuracy for flat signals with different sparsities and threshold $L$.

that guarantees the signal reconstruction reliability. As $L$ increases above 25, recovery accuracy gradually decreases and maximum performance degradation is about 12% when $L$ is increased to 59 with $K = 60$, which means all indices are obtained in one iteration. Given the number of measurements $M$ and signal length $N$, it can be seen that high quality of such flat signal reconstruction can be maintained with proper choice of $L_{MAX}$ as the upper limit of $L$.

Then, for the unequally-valued sparse signals, a gray image of fish in the white background with size $500 \times 1000$ pixels was recovered under different values of $L$ to visualize performance degradation of ST-OMP algorithm. Each column of the image was treated as one signal and the white background was set to zero along with smaller coefficients. As shown in Figure 31, the percentage indicates
Figure 31: Grey image reconstruction via the proposed ST-OMP.

how many non-zero elements were recovered using low-complexity procedure in the second phase of ST-OMP. Even at a level of 90%, the quality of recovered image is still acceptable, i.e., the object can be sufficiently identified. In addition, the MOMP algorithm is included in the ST-OMP algorithm with 2 indices added into the index set in each iteration at the beginning of ST-OMP algorithm. The signal recovery degradation can be observed but it is still of high quality.

Based on equation (40) and previous chapters on OMP algorithm implementation, normalized results of estimated energy consumption for the grey image reconstruction are shown in Table 11. These results further show that ST-OMP
can significantly reduce computational complexity and energy consumption while maintaining high reconstruction quality. This is very important for realtime energy-constrained embedded system.

Table 11: Hardware measurements of the proposed ST-OMP.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Percentage of energy consumption (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>OMP</td>
<td>100</td>
</tr>
<tr>
<td>ST-OMP-10%</td>
<td>78.58</td>
</tr>
<tr>
<td>ST-OMP-10%/MOMP</td>
<td>39.29</td>
</tr>
<tr>
<td>ST-OMP-50%</td>
<td>20.77</td>
</tr>
<tr>
<td>ST-OMP-50%/MOMP</td>
<td>10.39</td>
</tr>
<tr>
<td>ST-OMP-90%</td>
<td>0.74</td>
</tr>
<tr>
<td>ST-OMP-90%/MOMP</td>
<td>0.37</td>
</tr>
</tbody>
</table>

Then, the proposed technique is evaluated in a self-sustained video monitoring system. Consider the system to be powered by solar energy and process compressively-sensed video (24 images/second with image size of 500 × 1000 pixels) in realtime. The commonly used solar energy model [103] is adopted to describe daily solar energy.

Since solar energy pattern is repetitive yet non-deterministic, the reconstruction algorithm needs to be adjusted dynamically to achieve the highest possible reconstruction quality under different energy condition. Figure 32 shows the
available solar power (as in the dashed columns, which also reflect the runtime adjustment of ST-OMP based on the available power) and the corresponding performance measured by Signal-to-Error Ratio (SER), where errors are defined as the difference between the recovered and original images. If energy is low, ST-OMP with a larger threshold $L$ is selected to reduce the computational complexity, thereby reducing energy consumption in signal reconstruction. As expected, this will introduce some performance degradation. If setting the maximum allowable SER degradation to be 6dB, the corresponding $L_{MAX}$ can be determined. The proposed ST-OMP only fails 8.3% of the time (i.e., threshold $L > L_{MAX}$ in 2 out of 24 time slots). In comparison, original OMP will fail 87.5% of the time due to its large energy consumption under insufficient solar energy supply.
Therefore, through soft-thresholding the proposed technique is able to achieve better tradeoffs between recovery quality and energy consumption.

5.3.2 Case Study: DFT-Domain Signal Reconstruction

In this case study, the signals $X$ of length $N = 1024$ samples containing $S = 20$ complex-valued sinusoids are generated, in which the frequencies are randomly selected at each trial, which are not necessary to be at the integral frequency $(2\pi \cdot n/N)$. Measured signal $Y$ of length $M = 300$ is obtained using the normalized Bernoulli matrix $\Phi(\in \mathbb{R}^{300 \times 1024})$.

Then, the proposed ST-OMP algorithm is adopted to estimate the DFT-frequencies closest to the occupied frequencies of the signals $X$ in the DFT basis. As the estimation of DFT basis support $\hat{K}$ may not be accurate and usually is larger than the number of occupied frequencies $S$ due to spectral leakage or the measurement noises, inaccurate estimated DFT basis sparsity $\hat{K}$ also needs to be taken into consideration. Also, larger $\hat{K}$ can be interpreted as more strict halting criterion for the residual vector. After $10^5$ independent trials with different $L = 1, \ldots, 19$, and $\hat{K} = S + n$, where $n = 0, 5, 10$ and $15$, the success rate of DFT-frequencies closest to the occupied frequencies being located by ST-OMP algorithm is obtained and the results are shown in Figure 33.

From Figure 33, it can be seen that ST-OMP algorithm with $\hat{K} = S$ achieves high probability of targeting those DFT frequencies though under high $L$ (E.G.
<table>
<thead>
<tr>
<th>List length $L$</th>
<th>Probability $\hat{K} = S$</th>
<th>Probability $\hat{K} = S + 5$</th>
<th>Probability $\hat{K} = S + 10$</th>
<th>Probability $\hat{K} = S + 15$</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>1.00</td>
<td>0.95</td>
<td>0.90</td>
<td>0.85</td>
</tr>
<tr>
<td>10</td>
<td>0.95</td>
<td>0.90</td>
<td>0.85</td>
<td>0.80</td>
</tr>
<tr>
<td>15</td>
<td>0.90</td>
<td>0.85</td>
<td>0.80</td>
<td>0.75</td>
</tr>
</tbody>
</table>

Figure 33: Success rate of the detected DFT-frequencies by ST-OMP algorithm being closest to the occupied frequencies under different estimated sparsity $\hat{K}$ and list length $L$.

The probability is 0.9 when $L$ is 10). The performance is quite acceptable considering that there are not any restrictions on distance between two occupied frequencies in simulation that may lead to the failure of locating the two close DFT frequencies. Recall Proposition 1 that OMP algorithm is likely to first recover the non-zero elements with larger weight, which, in this case, are usually of the DFT-frequencies closest to the occupied frequencies, thus the recovery rate under high $L$ is still high. Furthermore, larger estimated $\hat{K}$ can significantly increase the rate of identifying the occupied frequencies and thus increase the reliability of sparse-channel detection, for example, the recovery rate at $\hat{K} = S + 15$
is almost 1 for $L$ up to 19. In the same time, larger upper limit $L_{\text{MAX}}$ of ST-OMP algorithm is more affordable under larger value of estimated $\hat{K}$ and thus the corresponding overhead of signal recovery can be maintained at certain level or even reduced.

![Figure 34](image)

Figure 34: Success rate of the detected DFT-frequencies by ST-OMP algorithm being in the sensitive area under different estimated sparsity $\hat{K}$ and list length $L$.

However, it is natural to concern us that larger estimated $\hat{K}$ may identify unused frequencies. Define a concept of sensitive area from whose frequencies to the occupied frequencies is less than $2\pi \cdot 5/N$. The rate that the estimated DFT-frequencies of ST-OMP are in the sensitive area are obtained after $10^5$ independent trials and shown in Figure 34. It can be seen that both larger $\hat{K}$ and $L$ contribute to the decrease of the rate but even at the maximum degradation
\((\hat{K} = S + 15 \text{ and } L = 19)\) the rate is still about 0.9, which is acceptable to CR in the sparse-channel detection as sufficient number of vacant frequencies can still be detected.

![Figure 35: Accuracy of recovering the closest DFT-frequencies and targeting the sensitive area under different variance.](image)

For comparison, the measured noiseless signal \(Y\) is changed to the noisy measurement input \(Y' = \Phi X + e\), where \(e\) is i.i.d. Gaussian noise of variance \(\sigma\) added to each measurement. Fix \(L = 10\) and \(\hat{K} = S + 10\), for each value of \(\sigma\), 10^5 independent trials are performed and the above two rates are obtained as shown in Figure 35. It can be seen that the rate of targeting the sensitive area is easier to be subject to the noise but its minimum rate is still above 0.96.
To demonstrate the signal recovery effects, fix $\hat{K} = S + 10$, $L = 10$ and $\sigma = 1.5$, the DFT of original signal $X$ and the estimated signal in DFT basis are shown in Figure 36. It can be seen that DFT of original signal is barely compressible with too many non-zero elements, but the proposed ST-OMP algorithm manages to detect the DFT-frequencies closest to the frequencies of signals. In addition, the extra detected DFT-frequencies due that $\hat{K} > S$ are also close to the occupied frequencies and sufficient number of vacant frequencies can still be detected.
5.4 Summary

The original iterative OMP algorithm involves large computational complexity in the last rounds of iteration while only less significant signal elements are recovered. Based on this observation, ST-OMP technique was proposed to improve the tradeoffs among computational complexity, performance, and cost in signal reconstruction. Recovering less significant elements with low-complexity computations can significantly reduce energy consumption without affecting recovery quality. This is a much-needed feature in many self-powered embedded systems. Further work is being directed toward developing a dedicated ASIC for the proposed ST-OMP with energy management technique introduced to efficiently distribute the limited energy to achieve higher performance.
Chapter 6

Conclusions and Future Work

6.1 Conclusions

In this dissertation, an algorithmic transformation technique, Matrix Inversion Bypass (MIB), was proposed to improve the Orthogonal Matching Pursuit (OMP) based signal reconstruction in compressive sensing applications. By decoupling two timing-critical operations in the signal recovery iteration, the speed of signal reconstruction can be greatly improved. The architectural design of the proposed MIB transform was also presented in this dissertation. This design targets application-specific hardware platforms such as dedicated hardware accelerators for embedded signal processing applications. The implementation of the proposed MIB transform was optimized to reduce hardware overheads and improve energy efficiency in a wireless video monitoring system. In addition, an FPGA-based architecture is realized for high-speed signal recovery by efficiently
utilizing the hardware resources through system-level optimization. Without introducing large hardware overheads, the proposed implementation improves the speed of signal recovery by up to $1.4 \times$ while maintaining the same level of algorithmic performance. A higher speed-up ratio can be achieved in the less sparse signals.

The original iterative OMP algorithm involves large computational complexity in the last rounds of iteration while only less significant signal elements are recovered. Based on this observation, a *Soft-thresholding Orthogonal Matching Pursuit* (ST-OMP) technique was proposed to improve the tradeoffs among computational complexity, performance, and cost in signal reconstruction. Recovering less significant elements with low-complexity computations can significantly reduce energy consumption without affecting recovery quality. This is a much-needed feature in many self-powered embedded systems.

### 6.2 Future Work

Future work is directed towards the followings:

- Identify other suitable applications for the proposed techniques.

- Further improve the hardware utilization efficiency for more efficient compressive sensing signal reconstruction.
• Apply the proposed technique and architecture to complex valued systems in which complex sparsity basis is needed, such as channel estimation and radar imaging.

• Develop a dedicated ASIC for the proposed ST-OMP with energy management technique introduced to efficiently distribute the limited energy to achieve higher performance.
Bibliography


Medicine, vol. 58, no. 6, pp. 1182 - 1195, Dec. 2007.


