西安交通大學
XI'AN JIAOTONG UNIVERSITY

High-Speed Compressed Sensing Reconstruction on FPGA using OMP and AMP (IEEE 2012)

By Lin Bai, Patrick Maechler, Michael Muehlberghuber, and Hubert Kaeslin
Integrated Systems Laboratory, ETH Zurich, Switzerland

Name: PATCHARADANAI SOMBATSATIEN

ID：3124999083

Course: Sparse Signal Processing and its Applications

Contents

西安交通大學
XI'AN JIAOTONG UNIVERSITY

Introduction

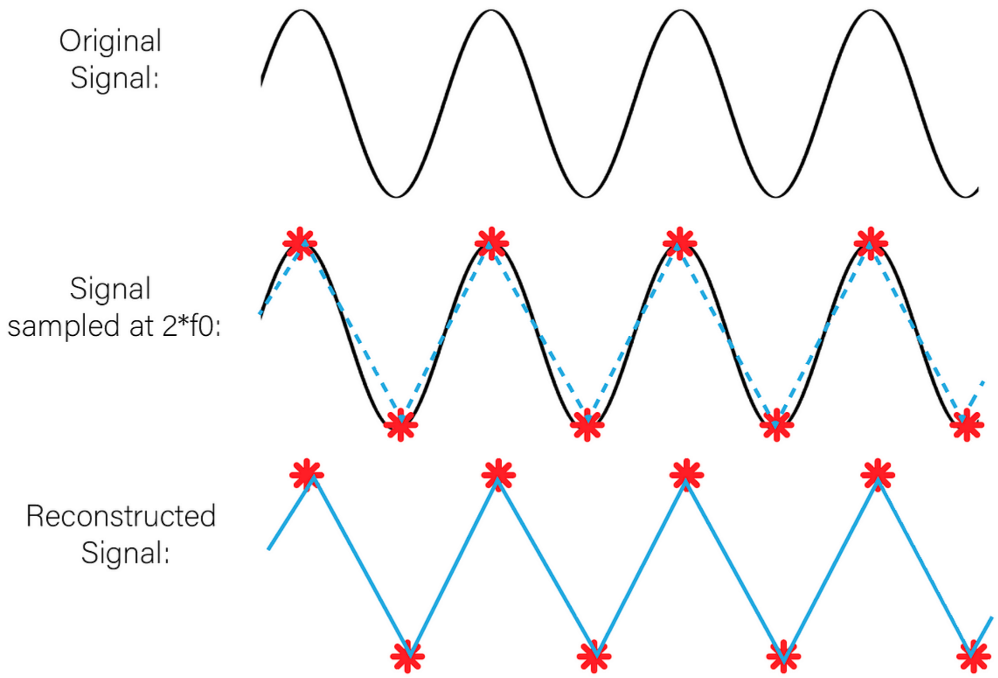
01

1 PART

Nyquist-Shannon sampling theorem

The Nyquist-Shannon sampling theorem is a fundamental principle in signal processing that defines the **minimum sampling rate** required to perfectly reconstruct a continuous-time signal from its discrete samples.

The theory set that the sampling rate must be **at least twice the maximum frequency of the signal** (Nyquist rate).



Nyquist rate: $f_s \geq 2f_c$

1 PART

Compressive Sensing

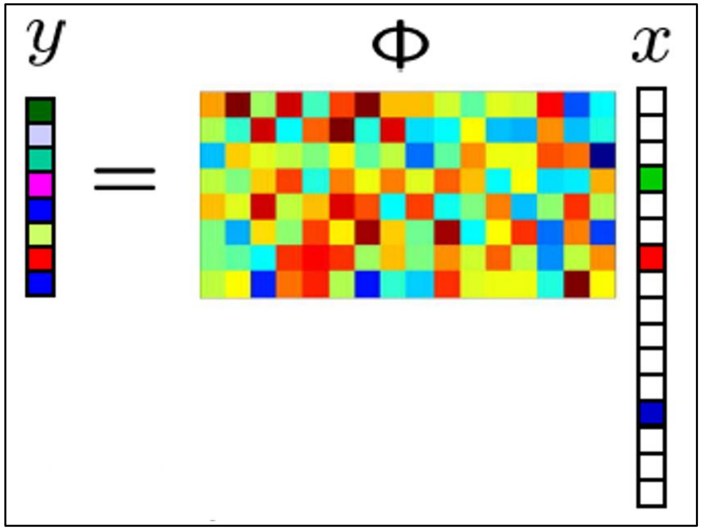
In contrast, Compressive Sensing addresses this limitation by **leveraging signal sparsity**, enabling the **recovery of sparse signals from a small number of linear measurements** which is **underdetermined linear system**. This has made Compressive Sensing attractive for numerous applications such as image acquisition, magnetic resonance imaging (MRI), wireless communication, and radar.

Key Concept of CS:

i. Lower sampling rate:
Nyquist rate: $f_s \geq 2f_c$ but CS: M ≪ N

ii. Efficiency for Sparse Signals:
Utilize signal sparsity to avoid redundant data acquisition.

iii. Reduces Hardware requirements:
enables cost-effective software reconfigurability



1 PART

Motivation of this Paper

The Compressive Sensing theory **enables efficient sparse signal processing** rather than using more **bandwidth** to reduce required measurements. Reducing the number of measurements can reduce the time or cost of signal acquisition. However, CS faces a significant challenge which is **the computational complexity of signal reconstruction** and **the hardware limitation**, which hold back embedded applications. Even though the development of fast recovery algorithms, the computational complexity remains very high.

To bridge this gap, this paper proposes **a programmable application-specific embedded processor** which allows for software implementations of Compressive Sensing algorithms, supporting cost-effective reconfiguration and enabling the implementation of sparse recovery algorithms.

1 PART

Field-Programmable Gate Array (FPGA)

FPGA is reconfigurable integrated circuits that offer **a combination of high-speed parallel processing and hardware flexibility**. Unlike general-purpose processors that execute instructions sequentially, **FPGAs can be programmed to perform multiple operations simultaneously through custom digital circuits**, enabling real-time signal processing with low latency. Their reconfigurable nature allows for hardware optimizations, balancing speed and resource efficiency.

FPGA perform well in embedded systems where low power consumption, high throughput, and adaptability to algorithm updates are critical. By leveraging parallel processing units, pipelining, and optimized memory access, FPGAs can achieved significant performance gains over software implementations.

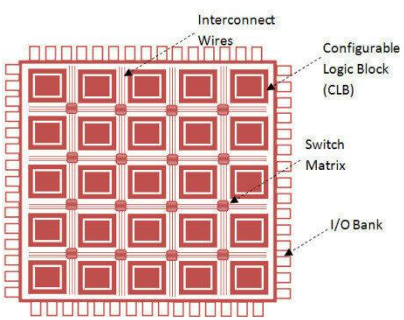
Key Concept of FPGA:

i. Parallel processing

ii. Low power consumption

iii. Updates require re-synthesis, unlike software updates

iv. Require Digital Logic Understanding



1 PART

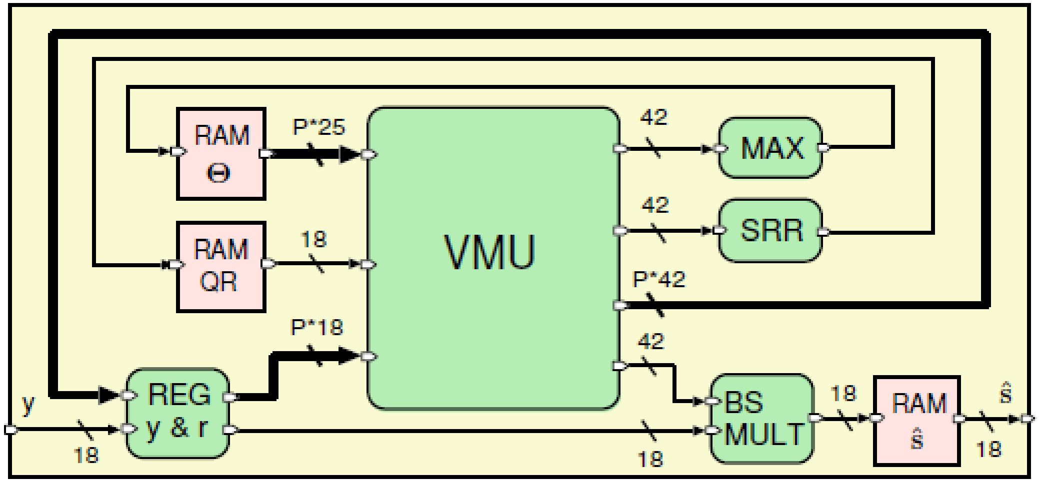
OMP Designed Block Layout

1. **Vector Multiplication Unit (VMU)** which supports multiple operation modes (matrix-vector, subtraction, scalar) for efficient computation. Utilizes 256 parallel multipliers for high-throughput processing.

2. **Memory Architecture**, Parallel block RAMs store Θ for fast column-wise access. Register banks enable simultaneous access to y and residual r.

3. **Fixed-Point Optimization,** 18-bit I/O, 25-bit for measurement matrix storage, and 42-bit accumulators. This setup balances precision with FPGA resource efficiency.

Reconfigurable architecture supports generic CS problems without relying on structured transforms. FPGA speedup over CPUs by eliminating sequential dependencies, making it versatile for embedded systems.



西安交通大學
XI'AN JIAOTONG UNIVERSITY

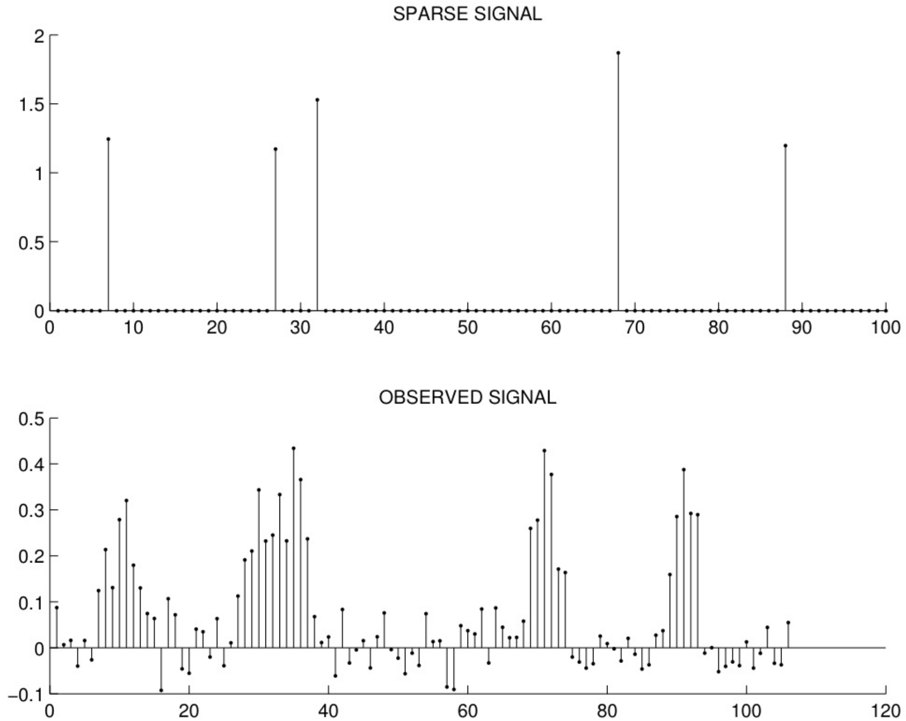
Sparsity in the paper 02

2
PART

Sparsity in the signal

Compressive Sensing required the assumption that signals are compressible signal which is **K-sparse in some basis** (e.g., wavelet, Fourier domain).

For a signal x with sparse representation s (where $x = \Psi s$)



2 PART

l_1 minimization

However, when the sparsity assumption is introduced, the original signal x can be reconstructed through solving the following l_1 minimization. l_1 minimization favors sparse solutions, it is convex problem, and linear programming for linear systems. While l_0 is the true sparsity, l_0 is NP-hard and non convex. In contrast, l_1 **under Restricted Isometry Property (RIP) conditions, l_1 solutions can match l_0 solutions with high probability**.

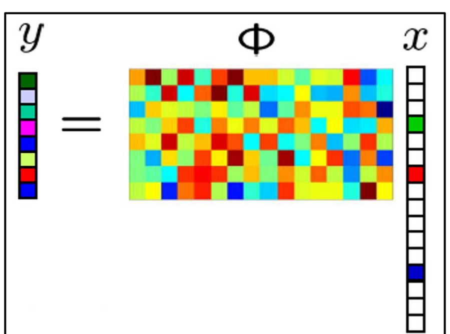
$$Y_{Mx1} = \phi_{MxN} * X_{Nx1}; \quad M \ll N$$

$$\arg\min ||x||_1 \;\; s.t. \;\; y = \phi x \quad ; \phi = \text{Measurement/Sensing matrix (phi)}$$

For a signal x with sparse representation s (where s = $\Psi^{-1} x$),

where: Ψ: The basis of sparse representation (psi)

s: The sparse coefficients of x in Ψ basis



$$\Theta = \Phi\Psi \in R^{MxN} \quad ; \Theta \text{ is combined measurement matrix (theta)}$$

$$\text{From } y = \phi x = \phi \Psi s = \Theta s$$

$$\text{So} \rightarrow s = \arg\min ||s||_1 \;\; s.t. \;\; y = \Theta s$$

After s is known, x can be recovered by x = Ψs

Implementation 03

Orthogonal Matching Pursuit

Iteratively selects the sensing matrix columns most correlated with the residual signal to identify non-zero coefficients. After each selection, it refines the sparse signal estimate through least squares (LS) optimization within the subspace formed from current and previously selected columns. This greedy approach makes locally optimal choices at each step. This ensures orthogonality of the estimate to the residual.

Input: measurement matrix Θ, measurements y, sparsity K	**Output**: Sparse reconstruction s^K
$r^0 = y \; and \; \Gamma^0 = \emptyset$	
for i = 1, …, K do	
$\lambda^i \leftarrow \; argmax_j \lvert (r^{i-1}, \Theta_j) \rvert$	# Find the best fit column
$\Gamma^i \leftarrow \Gamma^{i-1} \; U \; \lambda^i$	# Append the i-th best fit column into Γ^i
$s^i \leftarrow \; argmin_s \lvert\lvert (r^{i-1} - \Theta_{\Gamma^i} s) \rvert\rvert_2^2$	# LS optimization
$r^i \leftarrow \; r^{i-1} - \Theta_{\Gamma^i} s$	# Residual update
End for	

3 PART

Iterative Hard Thresholding

Employs iterative hard thresholding (Top-K) to retain significant coefficients and discard others. Its convergence and accuracy depend on parameters such as non-adaptive step size and threshold. Faster execution but compromised accuracy from fixed thresholding.

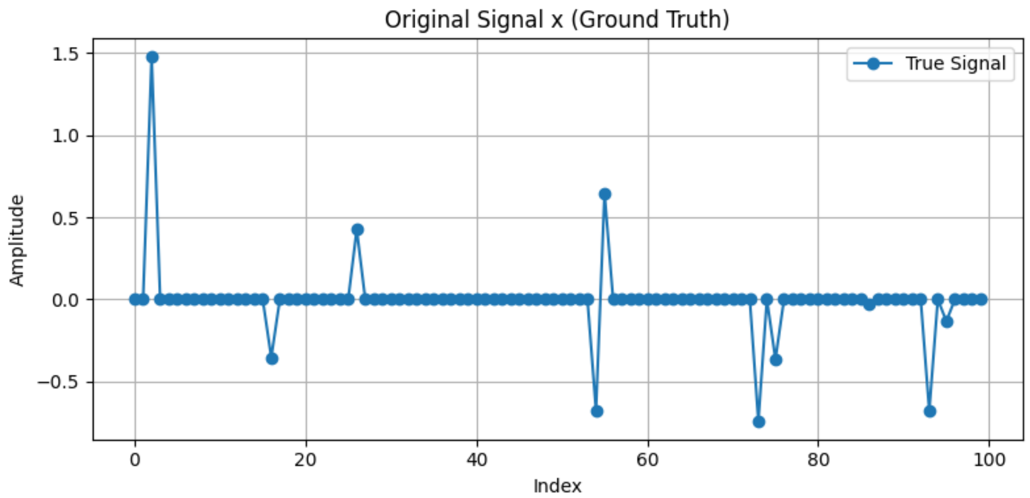
Input: measurement matrix Θ, measurements y, sparsity K, Iteration I_{max} , step size w	**Output**: Sparse reconstruction s^K
$r^0 = y\ and\ s^0 = 0_{Nx1}$	
$\boldsymbol{for\ i\ =\ 1, \ldots, I_{max}\ do}$	
$g^i \leftarrow matmul(\Theta, r^{i-1})$	# Gradient calculation
$s^i \leftarrow s^{i-1} + w * g^i$	# Gradient step
$s^i \leftarrow \eta_K(s^i)$	# Hard thresholding (keep top-K coefficients)
$r^i \leftarrow y - \Theta s^i$	# Residual update
End for	

3 PART

Implemented Signal

After finished the implement of aforementioned pseudo code of OMP and IHT, I tried to create randomly x original signal by declare **N (Signal length) and k (Sparsity level)**. Then randomly created **Θ measurement matrix** by using the dimension of N and M (Number of measurements). Finally, got **y observed signal** from matrix multiplication between original signal and measurement matrix, and have gathered all parameters.

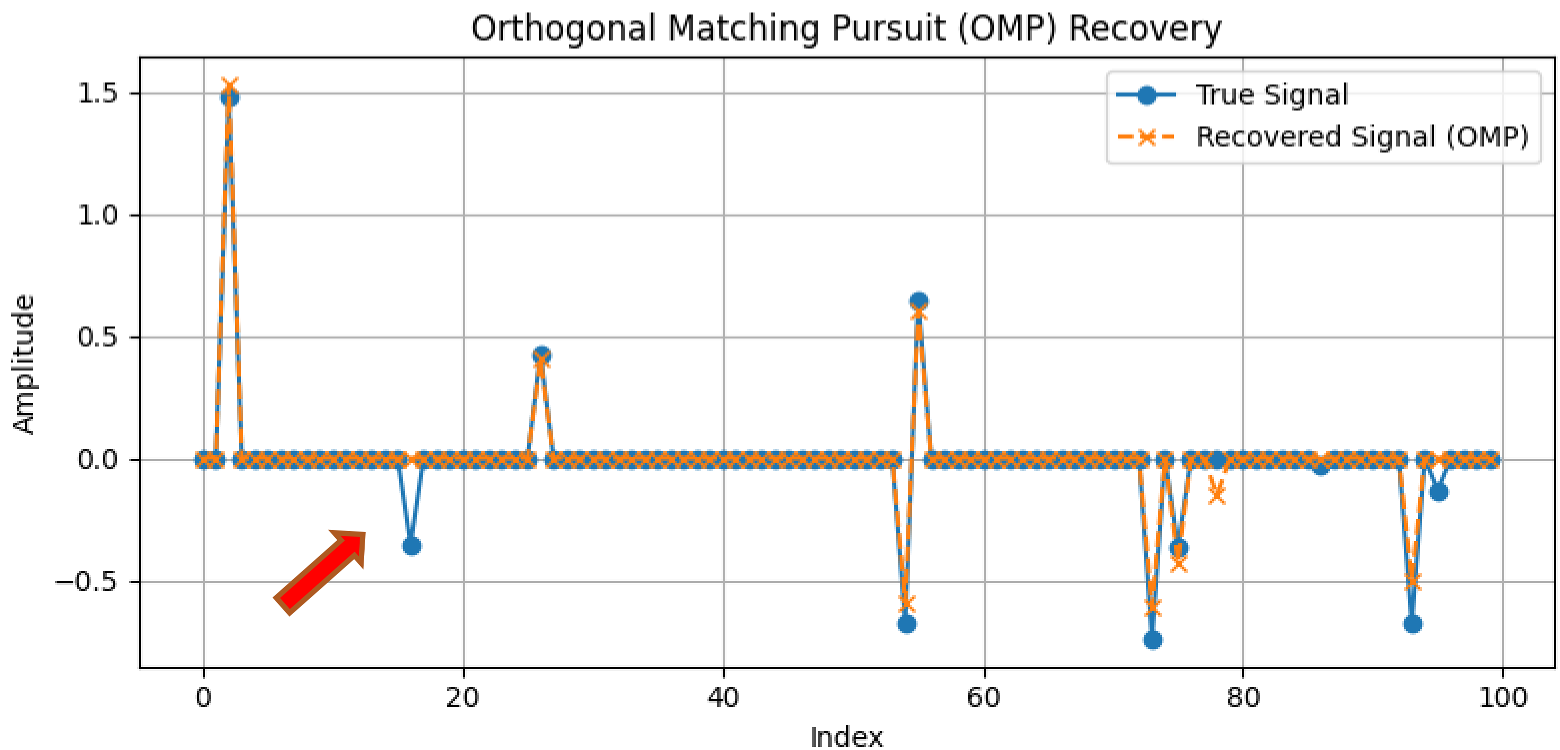
```
[46] # --- Example setup ---
     np.random.seed(0)
     N = 100          # Signal length
     M = 40           # Number of measurements
     k = 10           # Sparsity level
     iterations = 100

     # Random k-sparse signal
     x_true = np.zeros(N)
     nonzero_indices = np.random.choice(N, k, replace=False)
     x_true[nonzero_indices] = np.random.randn(k)

     # Measurement matrix and observed signal
     A = np.random.randn(M, N)
     y = A @ x_true
```



3 PART

OMP Reconstruction

Correctly identifies most of non-zero components and recover amplitudes closely match ground truth. False negative at indices near ~20 due to the correlation noise of signal.

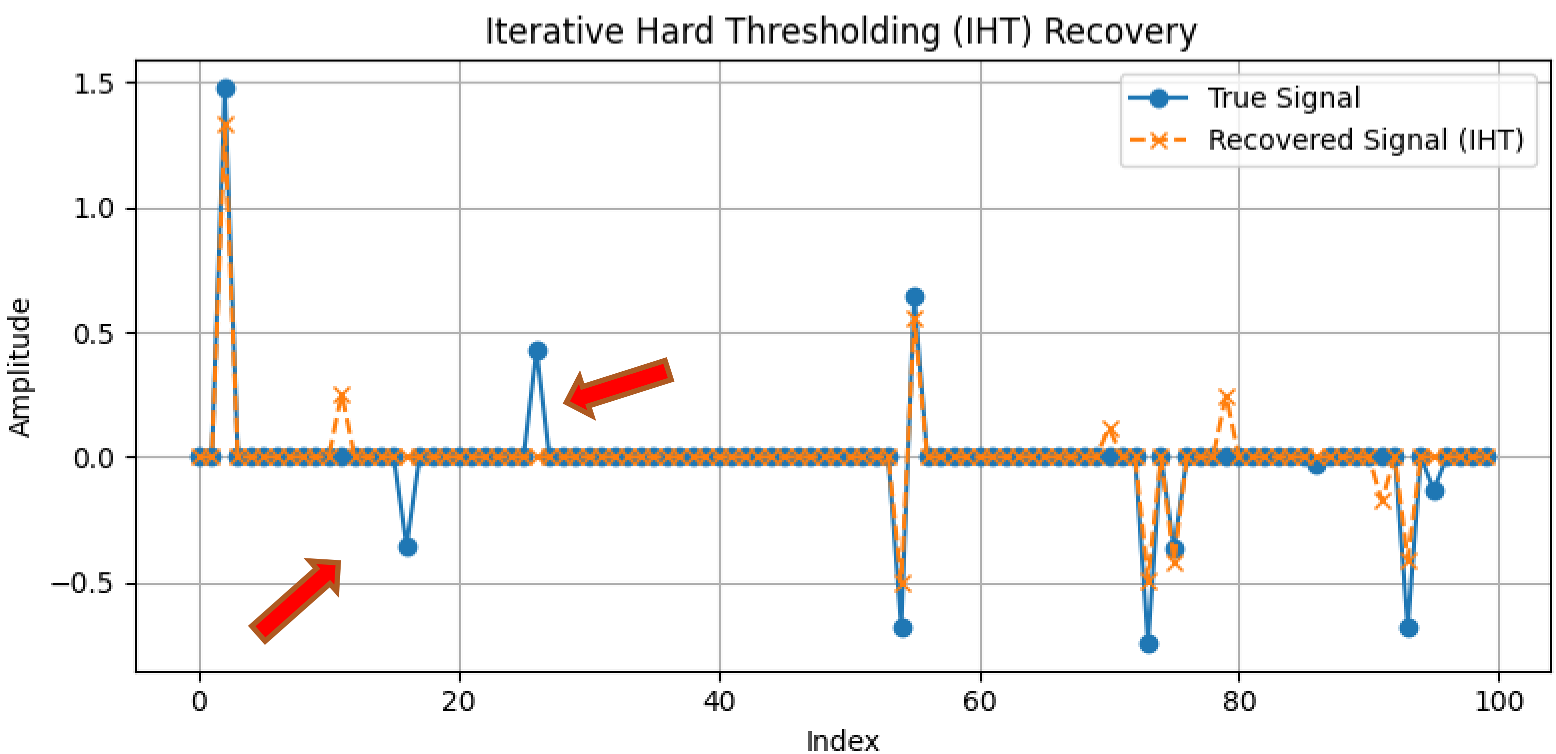


OMP Mean Squared Error (MSE): 0.002310

3 PART

IHT Reconstruction

Identifies major spikes but with amplitude underestimation (near zero spike is significantly lower than ground truth). Thresholding causes "leakage" around true spikes and some false negatives, and quantization artifacts which come from step size and hard thresholding visible in residual.



IHT Mean Squared Error (MSE): 0.006909

04 Summary

4 PART

The Comparison of CS Method

Unlike OMP, IHT does not require LS in each iteration, making it adaptable to diverse signal conditions. While OMP relies on iterative column selection and Least Square refinement, IHT leverages thresholding for efficient recovery. Both algorithms prioritize local optimization, but IHT offers greater flexibility in handling varying sparsity levels.

```
%%timeit
x_omp = omp(y, A, k)

2.48 ms ± 109 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)

[52] %%timeit
x_iht = n_iht(y, A, k, iterations)

1.96 ms ± 408 µs per loop (mean ± std. dev. of 7 runs, 1000 loops each)
```

Orthogonal Matching Pursuit (OMP)	Iterative Hard Thresholding (IHT)
Superior accuracy due to Least Square optimization	Compromised accuracy from fixed thresholding
Computationally expensive	Faster execution
Artifacts from greedy column selection (Sensitive to measurement matrix coherence)	Quantization artifacts which come from step size and hard thresholding

Summary of My Implemented Result

This project demonstrates that while both OMP and IHT algorithms effectively reconstruct sparse signals from compressed measurements.

- ❖ OMP provides superior reconstruction accuracy through its least-squares optimization but at higher computational cost, making it suitable for precision-critical applications like medical imaging

- ❖ IHT offers faster execution through iterative thresholding with moderate accuracy compromises, favoring real-time systems like radar processing.

The implementations highlight how OMP's resource-intensive approach yields exact spike recovery while IHT achieves lower accuracy with 27% faster performance, emphasizing that algorithm choice should balance precision needs against latency constraints in compressive sensing deployments. These results validate Compressive Sensing practical potential when paired with hardware-aware algorithm optimization.

Paper Result

TABLE I
FPGA IMPLEMENTATION RESULTS (XILINX VIRTEX-6)

	AMP	OMP
Frequency [MHz]	165	100
Proc. time [μs]	$15.81 \cdot I_{max}$	$10.97 \cdot K + 0.59 + \sum_{l=1}^{K} 0.34 \cdot l$
Slices	12113 (32%)	32010 (84%)
Block RAMs	256 (61%)	258 (62%)
DSP slices	258 (33%)	261 (33%)

Speed & Efficiency: 4000–5000× faster than CPU (Matlab).
Performance Trade-offs:

- OMP:
  - Superior accuracy (23.5 dB SNR for images).
  - Quadratic complexity in sparsity (*K*), optimal for *K ≤ 36*.
  - Higher resource usage: 84% slices, 62% BRAMs.
- AMP:
  - Faster for less sparse signals (21.4 dB SNR).
  - Fixed iteration count, independent of *K*.
  - Lower resources: 32% slices, 61% BRAMs.

4 PART

Reference

[1] M. Safarpour, I. Hautala, and O. Silvén, "An Embedded Programmable Processor for Compressive Sensing Applications," in Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), New Orleans, LA, USA, Mar. 2017, pp. 1013–1017.
DOI: 10.1109/ICASSP.2017.7952310

[2] L. Bai, P. Maechler, M. Muehlberghuber, and H. Kaeslin, "High-Speed Compressed Sensing Reconstruction on FPGA Using OMP and AMP," in IEEE Transactions on Signal Processing, vol. 62, no. 19, pp. 5076-5089, Oct. 2014.
DOI: 10.1109/TSP.2014.2345341.

Thank you