Sparse Signal Processing and its Applications Final Project:

High-Speed Compressed Sensing Reconstruction on FPGA (IEEE 2012)

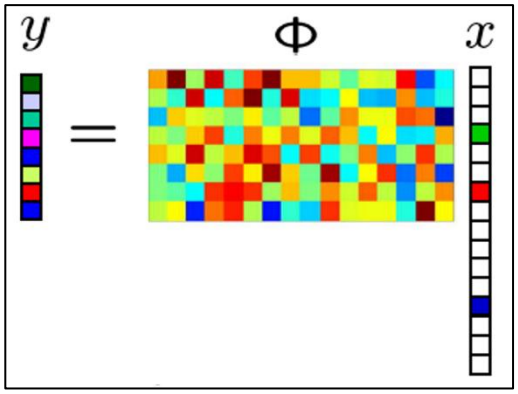
By Lin Bai, Patrick Maechler, Michael Muehlberghuber, and Hubert Kaeslin Integrated Systems Laboratory, ETH Zurich, Switzerland

1. Background & Motivations

1.1 Background:

Nyquist-Shannon sampling theorem: The Nyquist rate 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.

Compressive Sensing (CS): 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 called underdetermined linear system. This has made Compressive Sensing attractive for numerous applications such as image acquisition, magnetic resonance imaging (MRI), wireless communication, and radar.



$Y_{Mx1} = \phi_{MxN} * X_{Nx1}$

Nyquist rate: $f_s \geq 2f_c$ but CS: $M \ll N$

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

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.

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

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

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

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

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

After s is known, x can be recovered by $x = \Psi s$

1.2 Motivation:

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. This design enables cost-effective software reconfigurability, integrates a pseudo-random number generator for CS sampling, and demonstrates flexibility through support for multiple sparse recovery techniques. FPGA is reconfigurable integrated circuits that offer a combination of high-speed parallel processing and hardware flexibility, making them ideal for implementing computationally intensive algorithms like Compressive Sensing (CS) reconstruction. 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.

The project proposes parallel and reconfigurable FPGA implementations of two CS reconstruction algorithms: Orthogonal Matching Pursuit (OMP) and Iterative Hard Thresholding (IHT). These architectures are designed to support generic CS problems with configurable measurement matrices and are capable of handling both highly sparse and moderately sparse signals. The implementation not only demonstrates real-time performance but also show the comparison of algorithmic and hardware trade-offs between OMP and IHT. By targeting real-time application, hardware-efficient recovery, this work advances the deployment of CS in signal processing applications.

2. Recovery Algorithms

2.1 Orthogonal Matching Pursuit (OMP):

OMP iteratively selects the most correlated columns with the residual to identify non-zero coefficients. After each selection, it refines the sparse signal estimate through the 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 estimation.

Input: measurement matrix Θ, measurements y, sparsity K	**Output**: Sparse reconstruction s^K
1 $r^0 = y \; and \; \Gamma^0 = \emptyset$	
2 $\boldsymbol{for\ i\ =\ 1, \dots, K\ do}$	
3 $\lambda^i \leftarrow \ argmax_j \lvert (r^{i-1}, \Theta_j) \rvert$	# Find the best fit column
4 $\Gamma^i \leftarrow \Gamma^{i-1} \; U \; \lambda^i$	# Append the i-th best fit column into Γ^i
5 $s^i \leftarrow \ argmin_s \lvert\lvert (r^{i-1} - \Theta_{\Gamma^i} s) \rvert\rvert_2^2$	# LS optimization
6 $r^i \leftarrow \ \ r^{i-1} - \Theta_{\Gamma^i} s$	# Residual update
7 ***End for***	

2.2 Iterative Hard Thresholding (IHT):

IHT employs iterative hard thresholding (Top-K) to retain only the significant coefficients and discard others. IHT 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
1 $r^0 = y \; and \; s^0 = 0_{Nx1}$	
2 $\boldsymbol{for\ i\ =\ 1, \dots, I_{max}\ do}$	
3 $g^i \leftarrow matmul(\Theta, r^{i-1})$	# Gradient calculation
4 $s^i \leftarrow s^{i-1} + w * g^i$	# Gradient step
5 $s^i \leftarrow \eta_K(s^i)$	# Hard thresholding (keep top-K coefficients)
6 $r^i \leftarrow y - \Theta s^i$	# Residual update
7 ***End for***	

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 LS refinement, IHT leverages thresholding for efficient recovery. Both algorithms prioritize local optimization, but IHT offers greater flexibility in handling varying sparsity levels.

3. Implementation Result and Analysis

3.1 Python Implementation:

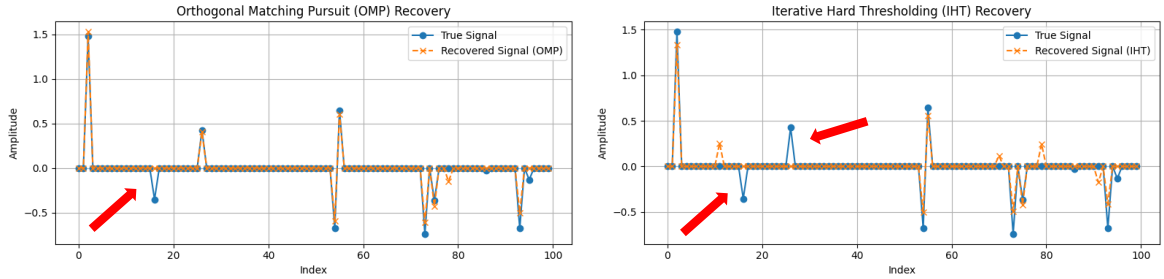
After finished the implement of aforementioned pseudo code of OMP and IHT, I tried to create randomly x original signal by declare **N (Signal length) and k (Sparsity level)**. Then randomly created ***Θ* measurement matrix** by using the dimension of N and **M (Number of measurements)**. Finally, got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

3.2 Result Analysis:

OMP: 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.

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



The computation time from 100 loops repeats show that IHT is faster than OMP around 27%.

```
%%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)
```

4. Project Summary

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

This project demonstrates that while both OMP and IHT algorithms effectively reconstruct sparse signals from compressed measurements, both exhibit distinct trade-offs: 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, whereas IHT offers faster execution through iterative thresholding with moderate accuracy compromises, favoring real-time systems like radar processing. The FPGA 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.

My Implemented Code: *https://github.com/Cheer3142/OMPnIHT_implement.git*