Compressive Sensing Algorithm for Signal Reconstruction on FPGA

By

Ryan Frohar
Marko Simic
Ross Matthew
Archit Bhatia

Supervisor: Professor Garcia

A proposal draft submitted for the SYSC 4907 Engineering Project

Department of Systems and Computer Engineering
Faculty of Engineering
Carleton University

October 15th, 2020

Table of Contents

List of Tables

List of Figures

Abstract

Modern technology in the fields of medical devices, video streaming and many other systems require a large amount data to be processed and communicated. These systems require a large bandwidth to process these extensive amounts of data. Compressive Sensing (CS) is a proposed solution to reducing large bandwidth requirements of many data processing systems. CS is a novel architecture used to sample a sparse signal at a sub-Nyquist rate [3]. CS requires complex and math intensive algorithms which can be slow and power consuming. The proposed solution will implement a CS algorithm, specifically an Orthogonal Matching Pursuit (OMP) algorithm, on an FPGA to be able to fully recover a sparse signal.

Chapter 1: Introduction

With many data processing applications emerging and gaining popularity, there is a need to overcome numerous problems that the technology brings. Signal processing applications often sample much more data than needed. For example, a digital camera samples a raw image only to save a JPEG file format. JPEG achieves approximately 10:1 compression of the raw image with little loss in image quality [12]. Such high compression ratios are possible due to the original image containing sparse data.

Signal processing was developed through the Nyquist/Shannon sampling theory which states that the number of samples needed to reconstruct a signal without aliasing is determined by its bandwidth. CS is a novel technology which allows the recovery of sparse signals sampled under the sub-Nyquist rate [3][10]. In many signal processing systems, the received signal contains many redundant values which are sampled and then discarded through compression. Compressive sampling suggests ways to translate sparse signals into already compressed digital form.

The number of samples required can change the functionality of a system, for example an MRI machine could have a greatly reduced scan time required through the implementation of CS [10]. The images taken by an MRI are often sparse and CS could allow the same images to be reconstructed with significantly fewer samples in a shorter scan time [16].

When data is sparse, one can directly acquire a condensed representation of the data with little information loss through linear dimensionality reduction. Linear dimensionality reduction in this context, transforms sparse data from high-dimensional space to low-dimensional space without losing important data [4]. Multiplying the input signal $\mathrm{X}\epsilon\mathbb{R}^{\mathrm{N}}$, which is m-sparse (number of non-zero elements in X is equal to m), by Gaussian-Random matrix $\Phi\epsilon\mathbb{R}^{KxN}$, where $\mathrm{K} \ll \mathrm{N}$, results in vector $\mathrm{Y}\epsilon\mathbb{R}^{\mathrm{K}}$ from equation (1) [4][9]. Since $\mathrm{K} \ll \mathrm{N}$, the resultant vector Y is significantly smaller than the input vector X [8]. The goal is to reduce K to be as low as m which is the relevant data of the signal.

$$\mathrm{Y} = \Phi\mathrm{X} \qquad (1)$$

The solution to condensing a sparse input vector X can be found using the basic properties of linear algebra such as matrix multiplication. Although, CS does not look for a condensed version of a highly sampled input signal. The goal of CS is to reconstruct a signal X given a sub-Nyquist rate sampled vector Y. The solution to this problem will be further looked at in Chapter 2.

The reason that CS has gained so much traction recently involves the condensed signal representation along with the ability to use lower rate Analog to Digital Converters (ADC). The lower sampling rate results in lower power and conservative memory usage [4].

As fourth year undergraduate computer systems engineering students, this project can provide us with the ability to apply our knowledge and skills to a newly emerging field. CS requires the ability to apply a mathematically complex algorithm on an FPGA to develop a complex system. This relates to our program as it is the development of hardware using to software to create a computer system. Using emerging techniques in the FPGA development field will allow to gather the knowledge to create a modern technology.

1.1 Problem Statement

Processing data for signal reconstruction over a communication channel requires a high amount of signal sampling to maintain accuracy. As the number of samples taken increases so does the required bandwidth which places a large load on the receiver. The proposed solution to this problem is implementing an OMP CS algorithm on the receiver. Sampling at a sub-Nyquist rate will decrease the required bandwidth and the load on the receiver. The algorithm will be implemented on an FPGA board and receive a signal from sparse source.

1.2 Organization of Proposal

This proposal will be divided into 3 chapters and a conclusion. Chapter 2 discusses the proposed solution as well as the existing research that exists on the topic. Chapter 3 discusses the project management aspect of the project, including: the software methodology that will be used, a comprehensive project timeline and a team-member role breakdown.

Chapter 2: Proposed Solution

2.1 What Exists?

As the compressive sensing field is emerging in modern technologies, there have been various approaches in reconstruction algorithms, most of which are computationally intensive. Sparse recovery algorithms can be classified into three main categories: Convex Optimization, Matching Pursuit Algorithms, and Bayesian category [19].

Convex Optimization algorithms resolve the sparse signal problem by applying convex relaxation algorithms [19]. These solutions rely on advanced techniques, such as projected gradient methods, interior-point methods, or iterative thresholding [7]. Examples of such involve Basis Pursuit which approaches sparse signal representation by changing the problem to one of minimizing the ℓ 1 -norm of the representation coefficients [20][15]. Matching Pursuit algorithms uses an iterative greedy process to mathematically reconstruct the signal at a sub Nyquist rate. These algorithms tend to be a quick and easy way to find solutions of sparse signals, because most of the optimization techniques take advantage of non-adaptive linear projections to preserve the structure of the signal [6]. Finally, the Bayesian techniques solves the sparse recovery problem by considering a prior knowledge of the sparse signal distribution [7][20]. Bayesian inference offers the potential for more precise and accurate point estimation of the sparse signal or a reduction in the number of CS measurements [17]. In the Bayesian Framework all unknowns are treated as stochastic quantities with assigned probability distributions and can be used to address measurement noise [19]. The three categories each have their own strengths and weaknesses when applying compressive sensing when being applied on an FPGA.

When applying Compressive Sensing on FPGA many of the existing approaches use greedy mathematically iterative algorithms. The reconstruction using convex optimization is more accurate than matching pursuit algorithms, but most researchers focus on matching pursuit algorithms because they are less computationally complex [2]. In comparison with Bayesian Techniques, Matching Pursuit algorithms are faster than other techniques but have a higher probability of error [20]. Bayesian Techniques can be unsuitable for high-dimensional problems due to their intensive computational requirements [7]. Due to these facts, Matching Pursuit algorithms are the most suitable approach for FPGA implementation.

The most common matching pursuit algorithm is the Orthogonal Matching Pursuit, which solves the problem by choosing the most significant variable to reduce the least square error. OMP is used due to its ability to use lower sparsity degrees, and results in less iterations due to the least squares operation [9]. Although it requires less iterations than matching pursuit (MP), it is more computationally intensive. Additionally, OMP has been shown to be quicker than other matching pursuits algorithms such as stagewise OMP and compressive sampling OMP [6]. OMP has been proven to be the quickest and simplest to implement. Many researchers have applied different techniques to the OMP algorithm to satisfy different requirements such as power consumption [1] and optimization [9]. OMP is the algorithm we have chosen to implement due to its relative simplicity and speed. OMP will be further discussed in section 2.2 under the proposed solution.

2.2 Analysis of Solution

CS relies heavily on the sparsity of the signal, which is found abundantly in bio signals, medical images, and radar signals [9]. The intended implementation for this project is to be able to reconstruct an under sampled bio signal using the OMP compressive sensing algorithm deployed on a FPGA.

The OMP algorithm is a computationally intensive reconstruction algorithm that aims to reconstruct a sub-Nyquist rate sampled sparse signal [1]. This is done by leveraging the properties of linear algebra and statistical optimization [9]. Suppose X is a one-sparse (one non-zero element) vector of size N where the second element of X contains the value of two and all the other elements of X are zeros as seen in **Figure 2.1**.

$$X = [\,0\,,2\,,0\,,0\,,\dots\dots..,0\,]^T$$

Figure 2.1: One-Sparse vector X

The product $\Phi \cdot \mathrm{X}$ will be the second column of Φ multiplied by two. Therefore, from the properties of linear algebra, the vector Y will be $X_2\Phi_2$ which equals $2 \cdot \Phi_2$. Since Φ_2 is a column from the Gaussian-Random matrix, Y is simply a scaled version of that column. The goal of compressive sensing is to obtain the vector X with sole inputs Φ and Y [4]. Since vector X is sparse, we know that vector Y is a sparse linear combination of the columns of Φ. In our example, Y is mathematically the second column of Φ multiplied by two, therefore Y will have the highest statistical correlation to the second column of Φ when compared to any other column in the matrix. The column in Φ with the highest correlation with vector Y yields which index of vector X is non-zero. Clearly, this example is simplified since, CS aims to reconstruct signals that are m-sparse where m >> 1. The idea of correlation is used heavily in the OMP algorithm [4].

The reconstruction algorithm requires two inputs, the matrix Φ and the measurement vector Y [4]. These two matrices will be inputted into the OMP algorithm and an approximation of the original signal X will be output [4]. This is because the optimal reconstruction is a non-deterministic polynomial time problem and involves iterating over all possible solutions for vector X [9]. Therefore, this algorithm aims to approximate the original signal X.

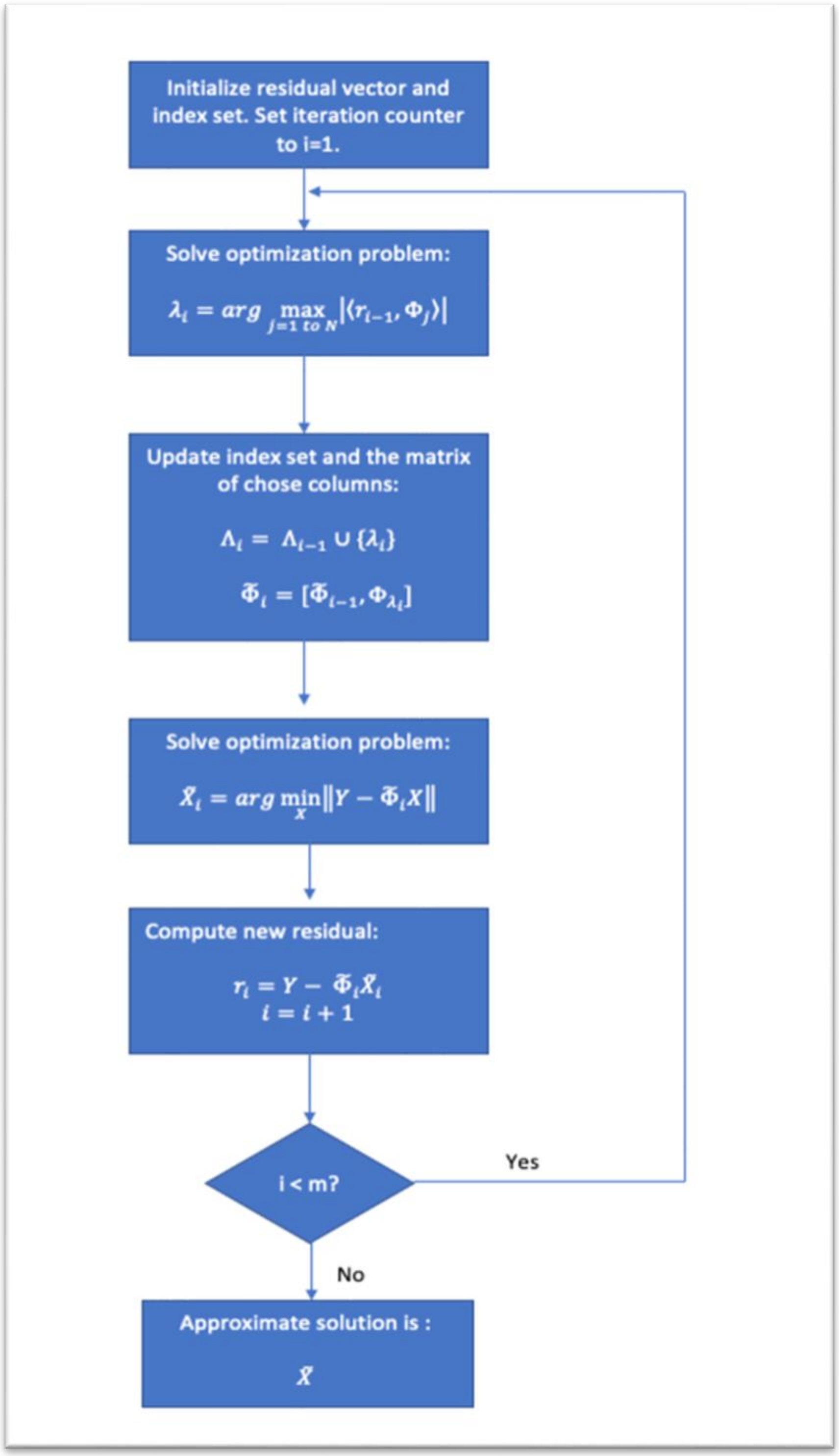
If vector X is an m-sparse signal, we need to find the m columns of the matrix Φ which correlate the most to measurement vector Y [4][8]. This set of columns is referred to as the index set. The OMP algorithm is iterative therefore every time a column from Φ is chosen, its specific contribution is subtracted from vector Y and iterated over again. Once the index set has been determined, a matrix comprising of these specific columns, indicated as $\widetilde{\Phi}$, can be found [9]. The values of the approximate signal X are found by solving an over-determined least squares system of matrices. The OMP algorithm is a detailed and iterative seven step process outlined below [9]:

1. Initialize the residual r_0, and the index set $\Lambda_0 = \{\emptyset\}$. Set Iteration counter i=1
2. Find the index λ_i, solving the optimization problem:
$$\lambda_i = arg \max_{j=1\ to\ N} \left|\langle r_{i-1}, \Phi_j \rangle\right|$$
3. Update the index set and the matrix of chosen columns:
$$\Lambda_i = \Lambda_{i-1} \cup \{\lambda_i\}$$
$$\widetilde{\Phi}_i = [\widetilde{\Phi}_{i-1}, \Phi_{\lambda_i}]$$
4. Solve a least square problem for new approximation $\tilde{X}$ of signal X:
$$\tilde{X}_i = arg \min_X \left\|Y - \widetilde{\Phi}_i X\right\|$$
5. Compute the new residual:
$$r_i = Y - \widetilde{\Phi}_i \tilde{X}_i$$
6. Increment counter "i" and return to second step if i < m.
7. Gather final approximation $\tilde{X}$

A flowchart of the OMP algorithm is shown in **Figure 2.2**. The algorithm is essentially split up into three main tasks. Determining the index set, least square solving for computing the signal approximation and residual update.

The OMP algorithm involves complex mathematical computations and outputs the reconstructed signal in real time. Therefore, the FPGA board being used must have high processing abilities. Some of the metrics which are important for an FPGA to be considered high performing in this context are: fast memory, large memory space, fast clock speed and amount of DSP slices.

While the clock speed of a FPGA is often not the limiting factor in signal processing it is still important for completing mathematical equations, which require multiple operations to be performed [5]. As well, dedicated DSP slices on an FPGA allows for very efficient multiplication and multiply-accumulate (MAC) operations [5]. The time it takes to complete the matrix multiplications which are performed in the OMP algorithm will be increased by these factors which is important for outputting real-time measurements.



Initialize residual vector and index set. Set iteration counter to i=1.
Solve optimization problem:
$\lambda_i = arg \max_{j=1 \, to \, N} |\langle r_{i-1}, \Phi_j \rangle|$
Update index set and the matrix of chose columns:
$\Lambda_i = \Lambda_{i-1} \cup \{\lambda_i\}$
$\tilde{\Phi}_i = [\tilde{\Phi}_{i-1}, \Phi_{\lambda_i}]$
Solve optimization problem:
$\tilde{X}_i = arg \min_x \|Y - \tilde{\Phi}_i X\|$
Compute new residual:
$r_i = Y - \tilde{\Phi}_i \tilde{X}_i$
$i = i + 1$
i < m?
Yes
No
Approximate solution is :
$\tilde{X}$

Figure 2.2: Flow chart of OMP Algorithm

Memory speed and size serves an important role in digital signal processing because there are many data points being taken and saved into a limited memory space which must be easily and quickly accessible. Memory reading and writing speeds will often present a bottleneck in a process as they are not as fast as most FPGA's clock rates.

The FPGA which was chosen that provided all of these important elements is the Xilinx Spartan-7 xc7s50 FPGA. On the selected development board, Spartan-7 operates at a 450MHz internal clock speed, has 128Mb Quad-SPI flash memory, 256MB DDR3L RAM, and has 120DSP slices.

In order to implement our OMP algorithm on to the FPGA, we have decided to use High Level Synthesis (HLS) as opposed to Hand Coded Hardware Description Language (HDL). Since the OMP algorithm is a highly complex mathematical algorithm, the design can be easily expressed in higher–level languages. Using HLS it makes it easier to code complex algorithms and is better suited for design reuse. High Level Synthesis eases the design validation and results in significantly shorter synthesize times. [18][13]. HDL makes it difficult to implement complex algorithms as it has many drawbacks in the language. HDL syntax is very ridged and error prone leading to a much harder implementation of computationally complex algorithms. Implementing the algorithm in HLS also results in a much smaller design time than HDL [13]. HLS achieves a worse performance, but it is close to their HDL counterparts when implementing complex algorithms [13]. This trade-off is wanted in our project as despite suffering a lower performance, HLS provides significant faster design and synthesize times. Furthermore, HLS uses a significantly more resources on the chip than HDL but with modern FPGAs there is a large resource availability to compensate the use of HLS [11]. HLS provides a higher-level abstraction to design a complex algorithm reducing design and synthesis times significantly. This will improve production and significantly improve design flexibility of our project in order to meet the tight deadlines mentioned in chapter 3.

Chapter 3: Project Management.

3.1 Timeline

The timeline of the project will be shown using a Gantt Chart in **Figure 3.1**.

3.1.1 Gantt Chart

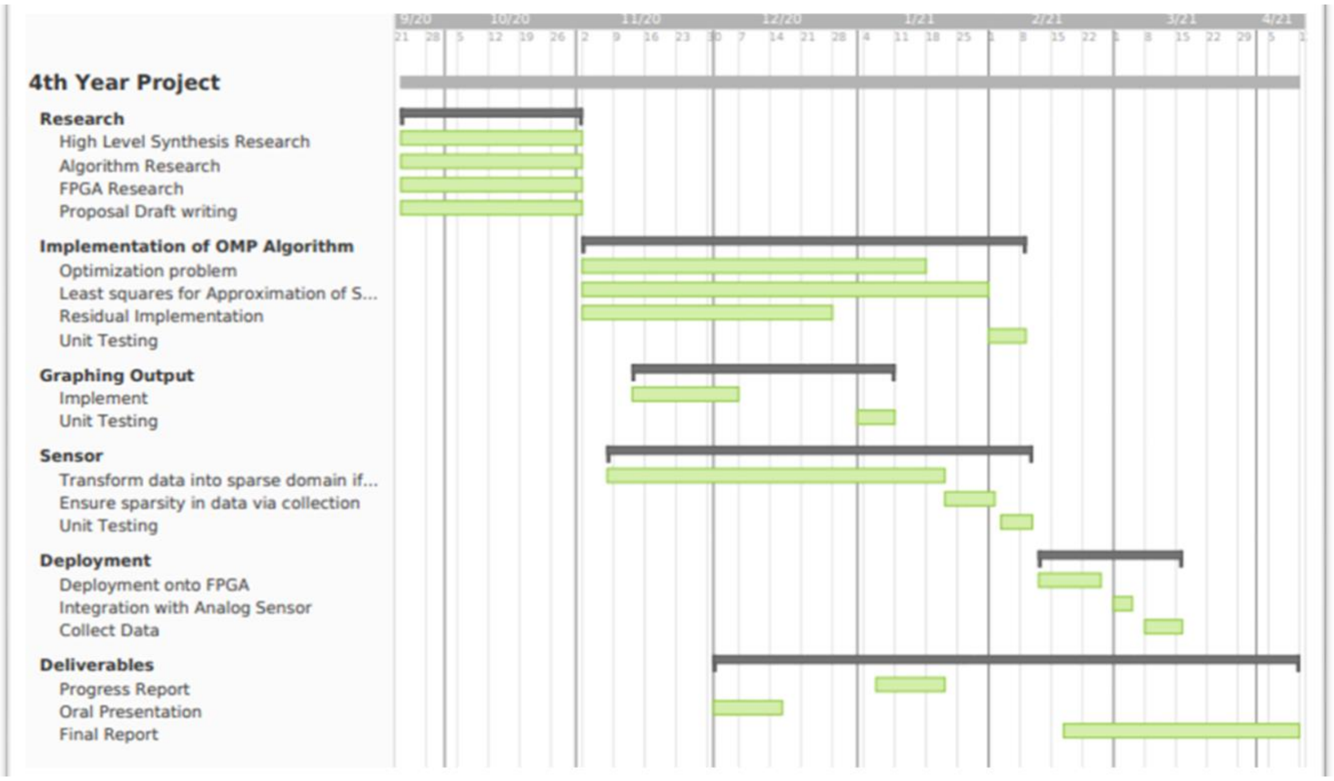


Figure 3.1: Gantt Chart of Project

3.2 Methodology

The methodology our group proposes to use for this project is the Agile Scrum process. Our group selected this method over other methods due to its ability to be adaptive, flexible and people oriented.

The nature of this project will require for bi-weekly meetings with the supervisor to discuss changes and overcome roadblocks within the project. Agile allows for developers to obtain feedback while in the developing stages [14]. The project will have changes made to it according to what works and what changes the supervisor will recommend.

3.3 Budget

The budget is yet to be finalized. Details will be available in the final draft of the proposal.

3.4 Role Breakdown

Group Member	Roles
Ryan Frohar	Proposal: • Timeline • Structure of report • Proposed Solution Research: • High Level Synthesis Vs Hand Coded HDL Implementations: • Optimization Problem • Sensors
Ross Matthew	Proposal: • Introduction • Proposed Solution Research: • FPGA Selection Implementations: • Residual Implementation • Sensors
Marko Simic	Proposal: • Abstract • Proposed Solution Research: • High Level Synthesis Vs Hand Coded HDL Implementations: • Least squares problem
Archit Bhatia	Proposal: • Methodology • Role break down • Conclusion Research: • FPGA Selection Implementations: • Unit Testing • Graphing outputs

Table 3.1: Break down of Roles

Conclusion

During communication between modern technologies, there are large data collections which lead to some unwanted data. Having large amounts of data leads to poor performance in the device, the best way to filter down these large sets of data are CS. In the report we presented the algorithms used in CS to filter out important data, solution we have proposed, and how we will manage this project. From the findings presented in this report we have concluded that by using OMP CS, data collection will be limited to important data hence increasing performance of the device while lowering power consumption.

References

[1] Amey Kulkarni and Tinoosh Mohsenin, *Low Overhead Architectures for OMP Compressive Sensing Reconstruction Algorithm*, IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 64, no. 6, pp. 1468–1480, 2017.

[2] Amey Kulkarni and Tinoosh Mohsenin, *Accelerating Compressive Sensing Reconstruction OMP Algorithm with CPU, GPU, FPGA and Domain Specific Many-Core*. 2015.

[3] Amey Kulkarni, Houman Homayoun, Tinoosh Mohsenin. *A Parallel and Reconfigurable Architecture for Efficient OMP Compressive Sensing*. May 2014.

[4] Avi Septimus and Raphael Steinberg, Compressive sampling hardware reconstruction, *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, 2010.

[5] Ciara Moore, Neil Hanley, John McAllister, Máire O'Neill, Elizabeth O'Sullivan, and Xiaolin Cao, *Targeting FPGA DSP Slices for a Large Integer Multiplier for Integer Based FHE.* Financial Cryptography and Data Security, 2013.

[6] Dinesh Veeramachaneni, *Implementation of Compressive Sensing Algorithms on Arm Cortex Processor and FPGAs*. June 2015.

[7] Elaine C. Marques, *A Review of Sparse Recovery Algorithms*. October 12th, 2018.

[8] Gunes Karabulut and Abbas Yongacoglu, *Sparse channel estimation using orthogonal matching pursuit algorithm*. IEEE 60th Vehicular Technology Conference, 2004.

[9] Hassan Rabah, Abbes Amira, Basant K. Mohanty, Somaya Almaadeed, and Pramod K. Meher, *FPGA Implementation of Orthogonal Matching Pursuit for Compressive Sensing Reconstruction.* IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 23, no. 10, pp. 2209–2220, 2015.

[10] Jerome L.V.M. Stanislaus and Tinoosh Mohsenin. *Low-Complexity FPGA Implementation of Compressive Sensing Reconstruction*. 2013.

[11] Marc-André Tétrault, *Two FPGA Case Studies Comparing High Level Synthesis and Manual HDL for HEP applications*. 21st IEEE NPSS Real Time Conference, Williamsburg, June 2018.

[12] Mazen M. Abuzaher, Jamil S. Al-Azzeh. *JPEG Based Compression Algorithm*. April 2017.

[13] Michael D. Zwagerman, *High Level Synthesis, a Use Case Comparison with Hardware Description Language*. April 2015.

[14] Mohamed Awad, *A Comparison between Agile and Traditional Software Development Methodologies.* The University of Western Australia, 2005.

[15] Patrick S. Huggins and Steven W. Zucker, *Greedy Basis Pursuit*. June 2006.

[16] Rick Chartrand, *Fast Algorithms for Nonconvex Compressive Sensing: MRI Reconstruction from Very Few Data.* Los Alamos National Laboratory, 2009.

[17] Shihao Ji, Ya Xue, and Lawrence Carin, *Bayesian Compressive Sensing*. IEEE transactions on signal processing, vol. 56, no. 6, June 2008.

[18] Stephane Gauthier and Zubair Wadood, High-Level Synthesis: Can it outperform hand-coded HDL?. June 25th 2020. https://www.silexica.com/wp-content/uploads/High-level-Synthesis-Can-it-outperform-hand-coded-HDL.pdf

[19] S. Derin Babacan, Rafael Molina and Aggelos K. Katsaggelos, *Bayesian Compressive Sensing Using Laplace Priors*. IEEE TRANSACTIONS ON IMAGE PROCESSING, VOL. 19, NO. 1, JANUARY 2010.

[20] Youness Arjoune and Naima Kaabouch. *Compressive Sensing: Performance Comparison Of Sparse Recovery Algorithms.* Unknown.