2013 International Conference on Computing, Networking and Communications, Multimedia Computing and Communications Symposium

Low-Complexity FPGA Implementation of Compressive Sensing Reconstruction

Jerome L.V.M. Stanislaus and Tinoosh Mohsenin
Dept. of Computer Science & Electrical Engineering
University of Maryland, Baltimore County

***Abstract*—Compressive sensing (CS) is a novel technology which allows sampling of sparse signals under sub-Nyquist rate and reconstructing the image using computational intensive algorithms. Reconstruction algorithms are complex and software implementation of these algorithms is extremely slow and power consuming. In this paper, a low complexity architecture for the reconstruction of compressively sampled signals is proposed. The algorithm used here is Orthogonal Matching Pursuit (OMP) which can be divided into two major processes: optimization problem and least square problem. The most complex part of OMP is to solve the least square problem and a scalable Q-R decomposition (QRD) core is implemented to perform this. A novel thresholding method is used to reduce the processing time for the optimization problem by at least 25%. The proposed architecture reconstructs a 256-length signal with maximum sparsity of 8 and using 64 measurements. Implementation on Xilinx Virtex-5 FPGA runs at two clock rates (85 MHz and 69 MHz), and occupies an area of 15% slices and 80% DSP cores. The total reconstruction for a 128-length signal takes 7.13 μs which is 3.4 times faster than the state-of-art-implementation.**

I. Introduction

In many signal processing systems, the useful information is far less than the sampled data and all the redundant data are eliminated through compression. One such method, compressive sensing (CS), reduces the amount of data collected during the acquisition [1], [2]. In compressive sampling, it is possible that sparse signals [2] and images can be recovered from very few samples (sub-Nyquist rate) compared to the traditional Shannon sampling. Compressive sensing has received a lot of attention in communication, radar and biomedical imaging applications.

In magnetic resonance imaging (MRI), where scan duration is directly proportional to the number of acquired samples, CS has the potential to dramatically decrease scan time [3], [4], [5]. Because information such as boundaries of organs is very sparse in most MR images, compressed sensing makes it possible to reconstruct the same MR image from a very limited set of measurements which significantly reduces the MRI scan duration.

CS has also received a lot of attention in radar imaging. Synthetic Aperture Radar (SAR) is an advanced radar imaging technique that provides long distance high resolution images through finer spatial resolution compared to the normal beam-scanning method. SAR images can be sparse or compressible in some representation such as those from wavelet transform. There have been several studies on CS theory and its applications in radar imaging [6], [7], [8], [9].

CS has two main stages - sampling and reconstruction. While sampling is performed on the transmitter, reconstruction is done on the receiver as shown in Fig. 1. Compact signal representation using CS results in lower rate ADC implementation which reduces the power dissipation and memory requirements as well. However, CS reconstruction requires high computational intensive algorithms.

Consider an m-sparse signal x of dimension $N \times 1$ and Φ is the measurement matrix of dimension $d \times N$, where d is the number of measurements to be taken. Multiplying these two vectors yields y of dimension d:

$$y = \Phi x \tag{1}$$

Now y is processed to reconstruct m values which will be the close estimate for x, denoted as $\hat{x}$. Since the reconstruction is NP-hard, efforts are made to find a close estimate of x. The signal can be sparse in any domain and not necessarily in the sampling domain. Reconstruction requires high computation and the complexity increases with the dimension of the signal. Also, reconstruction is an application of information theory and the complexity increases with the accuracy and the total measurements. There are several reconstruction algorithms proposed in recent years and most of them are computational intensive. Software implementation of these algorithms are time consuming since they often require matrix multiplications for which the processors are poor performers. The above mentioned drawbacks create more interests in hardware implementation for real-time reconstruction for CS signals. The two mostly used reconstruction algorithms are ℓ_1–minimization and Orthogonal Matching Pursuit (OMP). ℓ_1–minimization algorithm is better in terms of accuracy, but its implementation is very complex and time consuming. OMP is less complex [10] and it is a Greedy algorithm that finds the closely correlated values in each iteration. The complexity of the design increases with data length.

This paper uses a high speed architecture for OMP algorithm for 256– length input vector, and Q-R decomposition process (QRD) [11] is used for solving least square problem. The architecture is based on the high speed hardware architecture proposed by J. Stanislaus and T. Mohsenin [12]. A thresholding method is implemented in this paper that eliminates certain columns of Φ matrix to be used for dot product calculation for further iterations. The design has been implemented on Virtex-5 FPGA and results are compared with

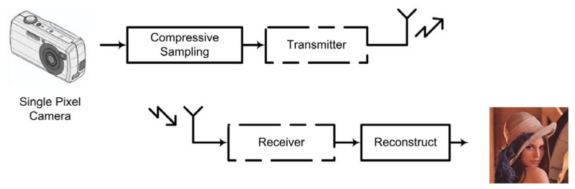


Fig. 1. Basic block diagram for compressive sensing

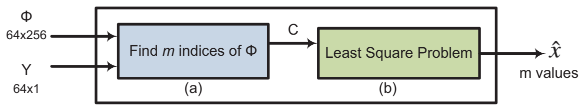


Fig. 2. Basic diagram for OMP reconstruction

.

the previous work.

II. Background

There are only a very few studies available for the implementation of OMP algorithm on the hardware [13][14]. Both studies are based on FPGA implementation, where [13] uses a 128–length vector for a sparsity of 5. The same architecture will take a lot more time to compute 256–length vector with a sparsity of 8 due to the path delay in finding the dot product and also due to the division operations performed in the matrix inverse. GPUs have a bottleneck on memory bandwidth [15]. In the architecture proposed in [12], more than 90% of the computation time is taken by the dot product computation. The high speed architecture explained in [12] will be used as a base for this paper and the new architecture design optimizations are discussed.

III. OMP Algorithm

Consider an m–sparse signal x sampled using a random matrix Φ and y is the sampled data. Our ultimate goal is to find m columns of Φ which contributed to y. To begin with, residual R is initialized to y. For each iteration, a column of Φ is chosen which has the best correlation with R. The residual R is then updated by subtracting the correlation from R for next iteration. This is repeated for m times to find m columns of Φ and the estimated signal $\hat{x}$ is obtained by solving an over-determined least square problem. The procedure is given below:

a. Initialize the residual $R = y$, the index set $\widetilde{\Phi} = \emptyset$ and the iteration counter $t = 1$

b. Find the index λ_t which is most correlated to Φ by solving the optimization problem

$$\lambda_t = arg \max_{j=1..N} | < R_{t-1}, \phi_j > | \tag{2}$$

c. Update the index set Λ_t and column set $\widetilde{\Phi}$

$$\Lambda_t = \Lambda_{t-1} \cup \{\lambda_t\} \tag{3}$$

$$\widetilde{\Phi} = [\widetilde{\Phi} \ \Phi_{\lambda_t}] \tag{4}$$

d. Calculate the new residual according to

$$R_t = R_{t-1} - (\widetilde{\Phi}_t \cdot \widetilde{\Phi}'_t)R_{t-1} \tag{5}$$

e. Increment t and return to step b if t is less than m

f. Solve the least square problem to find $\hat{x}$ for the indices in Λ

$$\hat{x} = arg \min_x ||\widetilde{\Phi}x - y|| \tag{6}$$

IV. Proposed Architecture

Figure 2 shows a high level block diagram for OMP reconstruction. OMP reconstruction consists of two main stages (a) and (b) and takes two inputs Φ and y. Φ is the resultant of measurement matrix (A) multiplied by the linear transformation matrix Ψ. This is done if the signal is not sparse in the sampling domain. Examples include capturing of an image in spatial domain and converted to wavelet domain by the linear wavelet transformation function Ψ. The image signal becomes sparse in the wavelet domain. The architecture works for 256-length signal with a sparsity of 8. Detailed analysis on the work performed by [13] and [12] show that the increased computation time is due to the dot production calculation in 2(a). The architecture proposed here implements a new thresholding method that cuts down the dot product calculation by upto 25% thus improving the reconstruction time of the overall architecture.

A. Improved OMP algorithm

Although the architecture explained in [12] is faster than the state-of-the-art implementation, 90% of the reconstruction time is consumed in the optimization problem for finding m columns of Φ. Studies are made to reduce the processing time of the optimization problem [16], but such methods are very complex to implement in hardware. The algorithm proposed here is scalable and less complex for implementing in hardware.

B. Algorithm

As seen before, x is an m-sparse signal which is sampled by Φ and y is the sampled data. For finding m columns of Φ, it is clear that we need to calculate the dot product for 2 vectors of length d for $N \times m$ times. The new thresholding method proposed in this paper eliminates certain columns of Φ to be used for dot product calculation for further iterations. The threshold is chosen to be the ratio α of the average of the absolute value of dot products. The columns of Φ whose absolute dot product value is below the threshold are excluded from further iterations. This threshold is updated at the end of each iteration. It should be noted that the mean square error (MSE) of the estimated signal increases with α and attains its minimum when α is 0. The procedure is given below:

a. Initialize the residual $R = y$, the index set $\widetilde{\Phi} = \emptyset$, $\hat{\Phi} = \Phi$, $thresh = 0$ and the iteration counter $t = 1$

b. Find the index λ_t which is most correlated to $\hat{\Phi}$ by solving the optimization problem

$$\lambda_t = arg \max_{j=1...k} | < R_{t-1}, \hat{\Phi}_j > | \tag{7}$$

where k = No. of columns of $\hat{\Phi}$

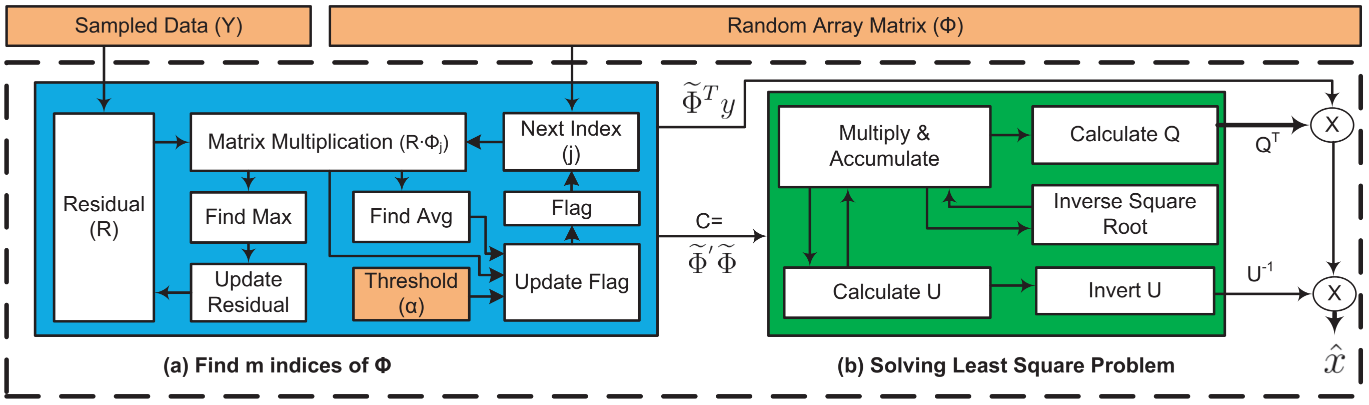


Fig. 3. Detailed architecture diagram of Improved OMP reconstruction algorithm. (a) This block iterates 8 times to solve the optimization problem eliminating columns of Φ those are poorly correlated to Y. Operates at 85 MHz. (b) Finds inverse of a 8x8 matrix after finding 8 columns of Φ. Operates at 69 MHz.

c. Update the index set Λ_t and column set $\widetilde{\Phi}$

$$\Lambda_t = \Lambda_{t-1} \cup \{\lambda_t\} \quad (8)$$

$$\widetilde{\Phi} = [\widetilde{\Phi}\ \hat{\Phi}_{\lambda_t}] \quad (9)$$

d. Calculate the new residual according to

$$R_t = R_{t-1} - (\widetilde{\Phi}_t \cdot \widetilde{\Phi}'_t)R_{t-1} \quad (10)$$

e. Calculate the average of the dot product

$$C_{avg_t} = \frac{1}{k}\sum_{j=1...k} | < R_{t-1}, \hat{\Phi}_j > | \quad (11)$$

f. Remove columns whose dot product is less than the threshold

$$thresh_t = \alpha \times C_{avg_t} \quad (12)$$

$$\hat{\Phi} = \hat{\Phi} \setminus \{\hat{\Phi}_j\} \ \forall\ |R_{t-1} \cdot \hat{\Phi}_j| < thresh_t \quad (13)$$

g. Increment t and return to step b if t is less than m

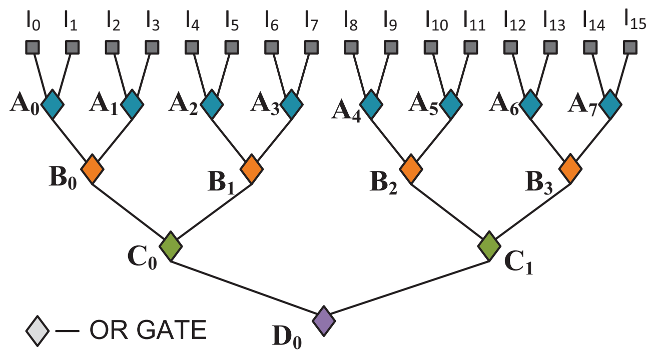
h. Solve the least square problem to find $\hat{x}$ for the indices in Λ

$$\hat{x} = arg \min_x ||\widetilde{\Phi}x - y|| \quad (14)$$

C. Solving Optimization Problem

The hardware is implemented for reconstructing a signal of length $N = 256$, measurements $k = 64$ and sparsity $m = 8$ similar to the work in [12]. The size of data used here is 24-bit which following 10.14Q (10 integers bits and 14 fractional bits) fixed point format. A series of 64 24-bit multipliers are operated in parallel to perform the dot product. This multiplier block is resource shared for sovling least square problem also. The maximum index (λ_t) is updated on each clock cycle as given by (2). At the end of each iteration, the residual is updated by subtracting it with the correlation of the column of Φ as shown in (5). Figure 3(a) shows the block diagram for solving the optimization problem and is repeated to find m indices of Φ as given by (3) and (4).

The architecture for the new algorithm is designed for the same signal specification; that is $N = 256$, measurements $k = 64$ and a sparsity of $m = 8$. Each data uses 24-bit 10.14Q fixed point format. The main change in the improved algorithm



I_0 I_1 I_2 I_3 I_4 I_5 I_6 I_7 I_8 I_9 I_{10} I_{11} I_{12} I_{13} I_{14} I_{15}
A_0 A_1 A_2 A_3 A_4 A_5 A_6 A_7
B_0 B_1 B_2 B_3
C_0 C_1
D_0
— OR GATE

$$O_0 = \overline{D}_0$$
$$O_1 = \overline{C}_0 \cdot C_1$$
$$O_2 = \overline{B}_0 \cdot B_1 + \overline{C}_0 \cdot \overline{B}_2 \cdot B_3$$
$$O_3 = \overline{A}_0 \cdot A_1 + \overline{B}_0 \cdot \overline{A}_2 \cdot A_3 + \overline{C}_0 \cdot \overline{A}_4 \cdot A_5 + \overline{C}_0 \cdot \overline{B}_2 \cdot \overline{A}_6 \cdot A_7$$
$$O_4 = \overline{I}_0 \cdot I_1 + \overline{A}_0 \cdot \overline{I}_2 \cdot I_3 + \overline{B}_0 \cdot \overline{I}_4 \cdot I_5 + \overline{B}_0 \cdot \overline{A}_2 \cdot \overline{I}_6 \cdot I_7 + \overline{C}_0 \cdot \overline{I}_8 \cdot I_9 + \overline{C}_0 \cdot \overline{A}_4 \cdot \overline{I}_{10} \cdot I_{11} + \overline{C}_0 \cdot \overline{B}_2 \cdot \overline{I}_{12} \cdot I_{13} + \overline{C}_0 \cdot \overline{B}_2 \cdot \overline{A}_6 \cdot \overline{I}_{14} \cdot I_{15}$$
$$Z = [O_0 O_1 O_2 O_3 O_4]$$

Fig. 4. 16-bit leading zero calculator

is the calculation of average and eliminating the columns of Φ whose dot product is below the threshold. To calculate the average of the dot products on each iteration, the absoulte value of the dot product of $\hat{\Phi}_i$ and R is added to an accumulator. Once the final sum is found, the average is calculated from the sum. Since fixed point division operation in hardware has larger latency, the average is approximated. The approximate average for iteration t is calculated as follows:

$$C_{sum_t} = \sum_{j=1}^{k} \hat{\Phi}_j \cdot R \quad (15)$$

$$C_{sum_t} = C_{sum_t} + (N-k) \times (\hat{\Phi}_{\lambda_t} \cdot R_{t-1}) \quad (16)$$

$$C_{avg_t} = \frac{C_{sum_t}}{N} \quad (17)$$

The advantage of this approximation is that it avoids the fixed point division by k as given in (11) since division by 256 is carried out just by shifting C_{sum_t} to the right through 8 bits. It is also obvious that the approximate average will be greater than the actual average. This means that MSE for this

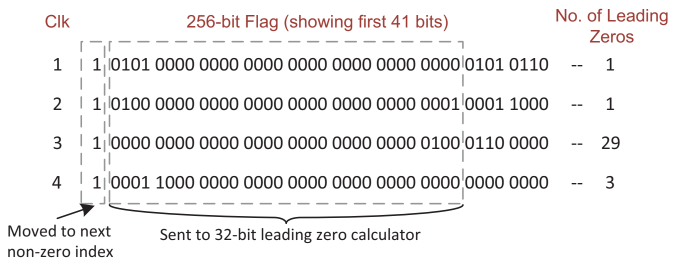


Fig. 5. Finding the next non-zero index of 256-bit flag for 4 clock cycles. First 41 bits are shown here and the remaining bits are assumed 0s

architecture will be smaller for a fixed threshold (α).

The improved architecture is built on top of the original architecture explained in section III. A 256-bit flag is used to keep track of the active columns of Φ for any iteration. The i^{th} bit of $flag$ corresponds to the i^{th} column of Φ. In the original hardware the dot product is calculated every clock cycle for i^{th} column of Φ and i is then incremented by 1. But in the improved architecture, i is incremented to the next non-zero index of $flag$, skipping all other columns of Φ in between. The next non-zero index of $flag$ is found by sending the first 32 bits of $flag$ to a 32-bit leading zeros calculator. The output (Z) of 32-bit leading zeros calculator is 6-bit which gives the number of leading zeros. Figure 4 shows the logic for finding the number of leading zeros for a 16-bit input (I).

The $flag$ is then shifted to its left by Z times to point it to the next non-zero index. Leading zero calculator for higher bit size (> 32) increases the latency and thus decreasing the operating frequency of the hardware. Figure 5 illustrates an example for 4 clock cycles. All the bits of $flag$ are set to 1 for the first iteration and are updated before the next iteration. The remaining hardware blocks are kept the same as shown in Fig. 3. The detailed pseudo-code for the above described algorithm is given below:

```
$flag \leftarrow AllOnes$
for $Iter = 1$ to $m$ do
    $Index \leftarrow 1; C_{sum} \leftarrow 0$
    while $Index < N$ do
        $DP_{Index} \leftarrow R \cdot \Phi_{Index}$
        $C_{sum} \leftarrow C_{sum} + DP_{Index}$
        $LeadZeros \leftarrow LeadingZeros32(flag)$
        $Index \leftarrow Index + LeadZeros$
        $flag \leftarrow flag << LeadZeros$ {<< - Left Shift}
    end while
    $C_{thresh} \leftarrow \alpha \times C_{sum}/N$
    for $i = 1$ to $N$ do
        $flag[i] \leftarrow (DP_i < C_{thresh})$ ? 0:1
    end for
end for
```

LeadingZeros32(X) gives the number of leading zeros for the first 32 bits of X

D. Solving Least Square Problem

After finding m columns of Φ, the next stage is to solve the least square problem given by (6). This often requires finding

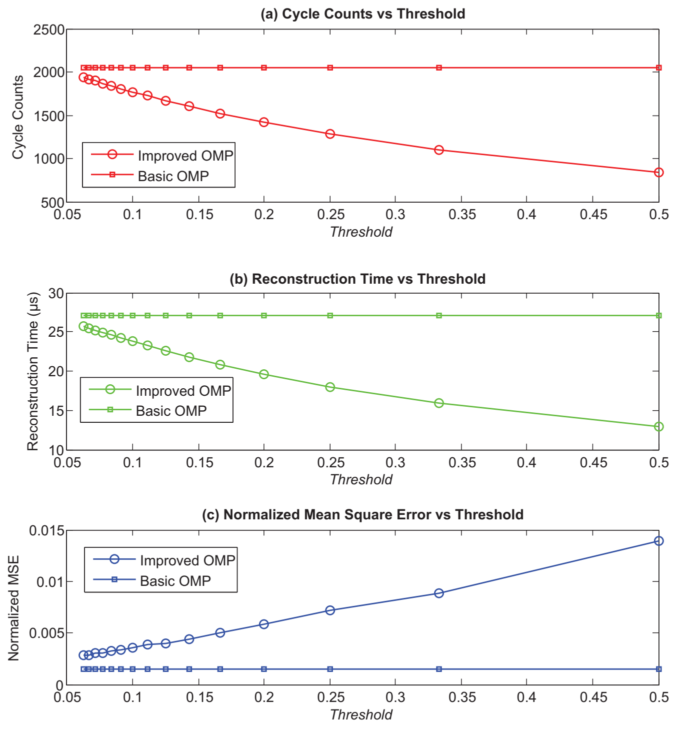


Fig. 6. Plots showing the variation of different parameters with the threshold (α) for the improved architecture. (a) Reduction in the cycle counts for increasing threshold. (b) Reconstruction time as a function of α. (c) Variation in Normalized MSE as a function of threshold.

the inverse of an 8x8 matrix C where $C = \widetilde{\Phi}^T\widetilde{\Phi}$. The main purpose of this is to solve for $\hat{x}$ from:

$$(\widetilde{\Phi}^T\widetilde{\Phi})\hat{x} = \widetilde{\Phi}^T y \tag{18}$$

The rest of the hardware is derived from [12] for its speed and efficiency for larger vector length. It includes the QRD method and also the fast inverse square root algorithm.

V. FPGA Implementation and Analysis

The proposed compressive sensing reconstruction architecture is synthesized and placed and routed on a Xilinx Virtex-5 XC5VL110T FPGA device. The implementation for the improved OMP runs at two different clocks, 85 MHz and 69 MHz, for blocks (a) and (b), respectively, as shown in Fig. 2.

A. Hardware Implementation of Improved OMP

For a 256–length vector with sparsity $m = 8$ and threshold (α) = 0, it requires 2100 clock cycles to find 8 columns of Φ and 160 cycles for QRD block. Hence the total reconstruction time is 27.14 μs. This proves that the real bottleneck of the algorithm is finding the dot product of Φ and residual R that takes $256 \times 8 = 2048$ cycles.

For the improved OMP, the threshold (α) is varied from 0.05 to 0.5 and the performance of the hardware is shown in Fig. 6. For each α value, the experiment is repeated for 1000 different sets of 256-bit input signal (x) of sparsity 8 and the average is taken. As expected, the cycle counts for improved OMP

	Septimus[13]	**This work**	
		Basic OMP	Imp OMP
Operating Freq(MHz)	39	85,69	85,69
Slice LUTs	NA	11%	15%
DSP Core	NA	80%	80%
Reconstruction Time*(μs)	24	10	7.13^{+}
Normalized MSE	0.0015	0.0015	0.0071^{+}
Speed Up	-	2.4	3.4^{+}

TABLE I
IMPLEMENTATION SUMMARY FOR THE PROPOSED OMP RECONSTRUCTION HARDWARE ON VIRTEX-5 FPGA. *RECONSTRUCTION TIME FOR 128-LENGTH VECTOR OF SPARISITY 5. $^{+}$ RESULTS FOR A THRESHOLD $\alpha = 0.25$

decreases with increasing α as shown in Fig. 6(a). It is also evident from Fig. 6(b) that the reconstruction time decreases with α due to the fact that block (a) of Fig. 2 contributes to 90% of the reconstruction time. Figure 6(c) shows the trade-off in Normalized MSE for improved reconstruction time. The improved OMP hardware for 128–length vector of sparsity 5 on Xilinx FPGA Virtex-5 takes approximately 444 cycles for finding 5 columns of Φ with threshold (α) = 0.25. This gives a reconstruction time of 7.13 μs which is 3.4 times faster than [13]. The approximate reconstruction time for a 256–length vector of sparsity 8 with $\alpha = 0.25$ is 18 μs. This is because it takes approximately 1280 cycles to find 8 columns of Φ as opposed to 2048 cycles for hardware implemented for basic OMP [12]. Table I summarizes the hardware implementation results for the proposed method compared to the state-of-the-art results [12], [13]. A reconstructed 256×256 image using the proposed method with $\alpha = 0.25$ is shown in Fig. 7. The image is divided into 256 sub-images of dimension 16×16 and the reconstruction is performed sequentially as described in [17]. The reconstructed image is enhanced using a 6×6 median filter.

VI. CONCLUSION

This paper presents an architectural design and FPGA implementation of low-complexity compressive sensing reconstruction hardware. The proposed architecture supports vectors of length 256. A thresholding method is applied for the improved version to eliminate certain columns of ϕ matrix for reducing the dot product computation latency. The reconstruction time on Xilinx FPGA Virtex-5 for a 128-length signal of sparsity 5 is 7.13 μs for proposed improved OMP with threshold, $\alpha = 0.25$. This is 3.4 times faster than the state-of-the-art implementation. The same architecture with threshold, $\alpha = 0.25$ for a 256-length signal of sparsity 8 takes on an average of 17 μs for reconstruction as compared to 27.12 μs reconstruction time of unimproved architecture as shown in Figure 6(b).

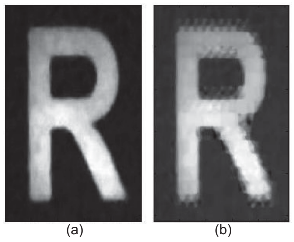


Fig. 7. (a) Original Image (b) Reconstructed Image (enhanced*) using the proposed method, PSNR = 30.25 dB. The 256×256 image is divided into sub-images of dimension 16×16 and iterated for 256 times. * denotes the reconstructed image is enhanced using a 6×6 median filter.

REFERENCES

[1] E. Candès, "Compressive sampling," in *Proceedings of the International Congress of the Mathematicians*, 2006, pp. 1433–1452.
[2] E. Candès and M. Wakin, "An introduction to compressive sampling," *Signal Processing Magazine, IEEE*, vol. 25, no. 2, pp. 21–30, Mar 2010.
[3] S. Ma, W. Yin, Y. Zhang, and A. Chakraborty, "An efficient algorithm for compressed mr imaging using total variation and wavelets," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, Jun. 2008, pp. 1–8.
[4] D. Kim, J. Trzasko, M. Smelyanskiy, C. Haider, A. Manduca, and P. Dubey, "High-performance 3d compressive sensing mri reconstruction," in *Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE*, Sep. 2010, pp. 3321–3324.
[5] M. Lustig, D. Donoho, J. Santos, and J. Pauly, "Compressed Sensing MRI," *Signal Processing Magazine, IEEE*, vol. 25, no. 2, pp. 72–82, Mar. 2008.
[6] R. Baraniuk and P. Steeghs, "Compressive radar imaging," in *Radar Conference, 2007 IEEE*, april 2007, pp. 128 –133.
[7] M. Herman and T. Strohmer, "Compressed sensing radar," in *Radar Conference, 2008. RADAR '08. IEEE*, may 2008, pp. 1 –6.
[8] ——, "High-resolution radar via compressed sensing," *Signal Processing, IEEE Transactions on*, vol. 57, no. 6, pp. 2275 –2284, june 2009.
[9] Y. Yu, A. Petropulu, and H. Poor, "Compressive sensing for mimo radar," in *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*, april 2009, pp. 3017 –3020.
[10] J. Tropp and A. Gilbert, "Signal recovery from random measurements via orthogonal matching pursuit," *IEEE Trans. on Information Theory*, vol. 53, no. 12, pp. 4655–4666, 2007.
[11] M. Karkooti, J. Cavallaro, and C. Dick, "FPGA Implementation of Matrix Inversion Using QRD-RLS Algorithm," *Signals, Systems and Computers, 2005. Conference Record of the Thirty-Ninth Asilomar Conference on*, pp. 1625–1629, 2006.
[12] J. Stanislaus and T. Mohsenin, "High performance compressive sensing reconstruction hardware with qrd process," in *Circuits and Systems (ISCAS), 2012 IEEE International Symposium on*, may 2012, pp. 29 –32.
[13] A. Septimus and R. Steinberg, "Compressive sampling hardware reconstruction," in *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*, 2010, pp. 3316–3319.
[14] D. Yang, H. Li, G. Peterson, and A. Fathy, "Compressed sensing based UWB receiver: Hardware compressing and FPGA reconstruction," *Information Sciences and Systems, 2009. CISS 2009. 43rd Annual Conference on*, pp. 198–201, 2009.
[15] M. Andrecut, "Fast GPU implementation of sparse signal recovery from random projections," 2008. [Online]. Available: http://www.arxiv.org/PS_cache/arxiv/pdf/0809/0809.1833v1.pdf
[16] N. B. Karahanoglu and H. Erdogan, "Compressed sensing signal recovery via a* orthogonal matching pursuit," in *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, may 2011, pp. 3732 –3735.
[17] P. Sermwuthisarn, S. Auethavekiat, and V. Patanavijit, "A fast image recovery using compressive sensing technique with block based orthogonal matching pursuit," in *Intelligent Signal Processing and Communication Systems, 2009. ISPACS 2009. International Symposium on*, jan. 2009, pp. 212 –215.