Low Overhead Architectures for OMP Compressive Sensing Reconstruction Algorithm

Amey Kulkarni, *Student Member, IEEE*, and Tinoosh Mohsenin, *Member, IEEE*

***Abstract*— Orthogonal Matching Pursuit (OMP) is an important compressive sensing (CS) recovery and sparsity inducing algorithm, which has potential in various emerging applications ranging from wearable and mobile computing to real-time analytics processing on servers. Thus application aware OMP algorithm implementation is important. In this paper, we propose two different modifications to OMP algorithm named Thresholding technique for OMP (tOMP) and Gradient Descent OMP (GDOMP) to reduce hardware complexity of OMP algorithm. tOMP modifies identification stage of OMP algorithm to reduce reconstruction time and GDOMP modifies residual update phase to reduce chip area. To demonstrate reconstruction efficiency of proposed OMP modifications, we compare signal-to-reconstruction error rate (SRER), signal-to-noise ratio (PSNR), and Structural Similarity index (SSIM) of previously proposed matching pursuit algorithms such as Subspace Pursuit (SP), Look Ahead OMP (LAOMP), and OMP, with tOMP, and GDOMP. We implemented reconfigurable, parallel, and pipelined architectures for three algorithms including OMP, tOMP, and GDOMP which can reconstruct different data vector sizes ranging from 128 to 1024, on 65 nm CMOS technology operating at 1 V supply voltage. The post place and route analysis on area, power, and latency show that, tOMP requires 33% less reconstruction time, and GDOMP consumes 44% less chip area when compared to OMP ASIC implementation. Compared to previously published work, the proposed architectures achieve 2.1 times improvement in Area-Delay product (ADP) and consume 40% less energy.**

***Index Terms*— Approximate Computing, ASIC, Compressive Sensing, energy efficient, low hardware complexity, OMP reconstruction.**

I. Introduction

ORTHOGONAL Matching Pursuit (OMP) algorithm has emerged as an important tool for signal recovery, dictionary learning and sparse data classification. In compressive sensing environment, OMP algorithm has shown tremendous growth as a reconstruction kernel and decryption kernel in crypto-systems [1]–[3]. For sparse data learning and classification, OMP algorithm can be adopted as a multi-least square regression classifier. Various modifications to OMP algorithm are performed to improve classification accuracy on sparse data [4], [5]. Recently, OMP algorithm has been implemented in dictionary learning techniques [6], [7]. Thus application aware OMP hardware implementation is of utmost importance.

Manuscript received July 18, 2016; revised November 7, 2016 and December 20, 2016; accepted December 20, 2016. Date of publication January 20, 2017; date of current version May 25, 2017. This paper was recommended by Associate Editor G. Masera

The authors are with the Department of Computer Science and Electrical Engineering, University of Maryland, Baltimore County, MD 21250 USA (e-mail: ameyk1@umbc.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSI.2017.2648854

Tremendous growth in digital data produced by all kinds of embedded systems including big data and IoT applications, demand reduced data storage and transfer rates with low overhead, and low error rate approximate computing technique. Approximation techniques should withstand the continuously changing data patterns and provide low overhead signal reconstruction. Hence in last couple of years, OMP has gained attention in approximate computing due to its data pattern adaptability and low approximation error rate [6].

CS-based frameworks and dictionary learning based data compression is becoming important in big data processing for various applications such as wearable devices [8], [9], server application [10], [11], and machine learning kernels [12]. In wearable devices and mobile computing, stringent constraints are on power and chip area with reduced latency whereas in server applications, priority is to reduce processing time. The goal of this paper is to propose application aware OMP reconstruction kernels.

In past, many OMP variants have been proposed to improve signal to reconstruction error [13]–[16], and application performance [17], [18]. Additionally, OMP CS reconstruction algorithm is evaluated on different platforms such as CPU, GPU [19], [20], [21], Many-Cores [22], FPGA [23], [24], ASIC [25]. GPU/CPU implementations of OMP algorithm have proven to be costly in terms of power, though they have the ability to reduce reconstruction time, whereas domain specific many-cores are hardware efficient in managing limited signal dimensions only. Therefore to reduce hardware cost, most of the researchers are inclined towards FPGA and ASIC implementations.

The main contributions to the research are:

OMP algorithm modifications to reduce hardware complexity: OMP has three important phases: *Identification*, *Augmentation*, *Residual Update*. The identification phase consists of dot product kernel, which consumes higher number of processing cycles [24], [26], and the residual update phase consists of least square kernel, which has highest computation complexity among all kernels [27]. In this work, we propose two different modifications to OMP algorithm to reduce hardware cost in terms of latency of operations and area requirements. We propose thresholding technique tOMP, to reduce cycle count required for dot product kernel in identification phase. To reduce hardware complexity of least square kernel in augmentation phase, we propose GDOMP algorithm which obtains the weight vector based on gradient descent method.

Algorithmic and hardware complexity evaluations: We compare hardware and time complexity of OMP algorithm

with respect to measurements and sparsity. We evaluate proposed algorithm using PSNR, SRER, and SSIM metrics for different types of images from object identification data set [28] and face detection data set [29].

Reconfigurable hardware implementation and evaluations: We implement reconfigurable OMP, tOMP, and GDOMP architectures that can reconstruct different data vector sizes ranging from 128 to 1024, on ASIC 65 nm, 1V6M, technology to measure hardware overhead in terms of chip area, reconstruction time and energy consumption. We also compare area-delay product (ADP) analysis with previously published papers [24]–[26]. To test OMP architecture, we implement Pseudo-Random Number Generator (PRNG) for Gaussian matrix generation and evaluated error rate between floating point software and fixed point hardware solution using mean square analysis.

The rest of the paper is organized as follows: Section II presents a survey of related work. In Section III, we discuss proposed two algorithms and perform evaluations on MATLAB® with respect to PSNR, SRER, and SSIM. Section IV describes proposed architectures for OMP, tOMP and GDOMP algorithms including sparsity and measurement analysis. Finally, Section V discusses architecture implementation analysis on ASIC 65 nm, 1V6M, CMOS technology.

II. Related Work

As discussed earlier, in last decade many researchers focused on CS reconstruction algorithms. For first few years, the trend in research on CS reconstruction was to reduce latency of operations [22], [30], [31] whereas the focus has been shifted over to low power [27], [32] and reconfigurable [23] architecture for large dimension signals.

The first architecture for CS reconstruction OMP algorithm is presented in Septimus *et al.* [30]. The paper discusses that identification phase is the bottleneck in low latency OMP signal reconstruction since it requires highest number of clock cycles. The least square problem is solved using modified Cholesky method. For implementation, it considers signal size of 128 and assumes sparsity of 5 with 32 (25% of original signal) measurements. The paper compares OMP hardware implementation with MATLAB® fixed-point simulations for execution time analysis. The paper does not show effects of CS on real-time application in terms of accuracy of reconstructed signal. Additionally in case of higher sparsity count, least square kernel will also dominate the latency of operations. To reduce latency of operations, parallel implementation is suggested in Blache *et al.* [31]. The OMP algorithm consists of three interdependent phases i.e. only one phase is executed at a given time while other phases remain idle, therefore execution time can be reduced by exploiting the parallelism in each phase. It uses modified Cholesky algorithm to obtain least square approximation. Each sub-matrices in matrix inversion are computed in parallel, however introducing block Cholesky implementation would have reduced reconfiguration overhead. The authors of this paper, Kulkarni *et al.* [23] introduced block multiplication and inversion to reduce reconstruction time and achieved $2\times$ faster reconstruction time as compared to Septimus *et al.* [33]. The hardware complexity and reconstruction time analysis on FPGA shows that, dot product kernel from identification phase and least square kernel from residual update phase are the architecture bottlenecks. Therefore in this paper we introduce two new modifications tOMP to reduce latency of dot product kernel and GDOMP to reduce complexity of least square kernel.

The low power CS reconstruction trend started in Huang *et al.* [27]. The paper solves least square problem using Matrix Inversion Bypass (MIB) method to reduce computational complexity. To reduce matrix inversion time, MIB takes advantage of previously computed matrix inversion to calculate matrix inverse of ongoing iteration. Furthermore to improve computation time efficiency, Huang *et al.* performs matrix inverse of $(k-1)^{th}$ iteration in parallel with k^{th} iteration. Additionally Huang *et al.* [32] targets battery operated devices, it proposes soft-thresholding technique to reduce energy consumption. In OMP, computation of k^{th} iteration has highest computation complexity and execution time as compared to other iterations, however it recovers very few elements for signal recovery. Huang *et al.* proposes to replace these iterations with low complexity procedure to reduce recovery cost without sacrificing much quality. The threshold at which algorithm switches is adjusted dynamically in accordance to the performance requirements and available energy levels. Hardware is implemented on 65 nm CMOS process with clock frequency of 500 MHz. It achieves significant reduction in computational complexity in particular when sparsity of signal is high because the low complexity procedure recovers more elements in such signals. The implementation of ST-OMP takes 0.16 ms to recover a signal and consumes 0.0205 mJ energy. Burg *et al.* [34] proposes ultra low power application specific instruction set processor (ASIP) for compressive sensing specifically targeted for wireless body sensor networks. ASIP has 3-stage pipeline comprised of fetch, decode and execute stage. The core operates one data word of 16-bit, with 14 unique instructions and sub-V_T latch based memories. The sleep mode allows external clock gating of entire core. The paper shows that there exists a power and execution time trade off. CS processor operates at 0.37 V at 100 KHz with total power 288 nW.

Recently CS has gained attention as a tool for approximate computing in big data processing. Zhang *et al.* [10] implements compressive sensing based storage reduction for big data analytics, authors convey that for many big data workloads approximate results suffice. Yan *et al.* [35] incorporated CS-based framework into Hadoop and evaluated it on real web-scale production data. It shows reduction in data shuffling I/O up to 99%, and end-to-end job duration by up to 40%. In both [10], [35] research papers, implementations were performed on software platforms thus best performance can be achieved if implemented on hardware platforms such as FPGAs or domain specific many-cores. Xu *et al.* [9] shows efficient long term biomedical application processing using CS with OMP algorithm implementation, whereas Ahmad *et al.* [11] shows usage of CS in big data fusion approach for

Algorithm 1 OMP Reconstruction Algorithm

1:**Initialization**
- $R_0 = S$, $\Lambda_0 = \emptyset$, $_0 = \emptyset$ and $t = 0$

2:**Identification**
- Find Index $\lambda_t = max_{j=1...n}$ subject to $| < \phi_j R_{t-1} > |$

3:**Augmentation**
- Update $\Lambda_t = \Lambda_{t-1} \bigcup \lambda_t$
- Update $Q_t = [Q_{t-1}\ \phi_{\Lambda_t}]$

4:**Residual Update**
- Solve the Least Squares Problem $D_t = \min_D \|S - Q_S D\|^2$
- Calculate new approximation: $\alpha_t = Q_t\ D_t$
- Calculate new residual: $R_t = S\text{-}\alpha_t$

5: Increment t, and repeat from step 2 if $t < k$

After all the iterations, we can find correct sparse signals.

machine to machine communication. In our previous work [3] we developed CS-based fully reconfigurable and scalable approach for big data acceleration on FPGA platform, in which CS sampling is performed on Artix-7 FPGA and CS reconstruction is implemented on Virtex-7 FPGA. To demonstrate the efficiency we explored seizure detection application consisting of 24 sensor channels each transmitting 256 data chunks every second. We also proposed heterogeneous big data acceleration using CS-based framework in [36], which processes 5000 images (2.17 GB) of various sizes ranging from 128×128 to 1024×1024 for object detection. Therefore, CS with OMP reconstruction algorithm is attractive and important in upcoming years to reduce data storage and transfer rates. In this paper, we proposed scalable OMP algorithm with reduced hardware complexity and Energy Delay Product (EDP). The proposed OMP architectures are scalable and can be reconfigured up to 1024 vector sizes.

III. Proposed Reduction Techniques

In this section we discuss OMP algorithm, and proposed two modifications to OMP named thresholding technique for OMP-tOMP, and Gradient Descent OMP-GDOMP algorithms. We perform experiments on two different big data sets to evaluate average PSNR, SRER, and SSIM for tOMP and GDOMP. Furthermore, we compare proposed new modifications with OMP, Look Ahead OMP (LAOMP) [14] and Subspace Pursuit (SP) [37].

Notations used in the paper

- D = Original data, S = Sampled reduced data (measured data), $\hat{D}$ = Reconstructed original data
- n = Length of the original data
- m = Number of measurements
- k = Sparsity
- R = Residual matrix
- ϕ = Measurement Matrix ($size : m \times n$)
- λ = Maximum index after dot product
- t = No. of iterations (usually equal to k

Algorithm 2 Thresholding Technique OMP Reconstruction Algorithm

1:**Initialization**
- $R_0 = S$, $\Lambda_0 = \emptyset$, $Q_0 = \emptyset$, $\phi_0 = \phi$, $p = \emptyset$ and $t = 0$

2:**Identification**
- Sort $\beta = sort_{j=1...n}$ subject to $| < \phi_{t-1} R_{t-1} > |$
- $\lambda_t = \beta_1$

3:**Augmentation**
- Update $\Lambda_t = \Lambda_{t-1} \bigcup \lambda_t$
- Update $Q_t = [Q_{t-1}\ Q_{\Lambda_t}]$

4:**Residual Update**
- Solve the Least Squares Problem $D_t = \min_D \|S - Q_S D\|^2$
- Calculate new approximation: $\alpha_t = Q_t\ D_t$
- Calculate new residual: $R_t = S\text{-}\alpha_t$

5:**Column Reduction**
- $\alpha_t = \beta_{(n-31)...n}$
- ϕ_t=eliminate[ϕ_{α_t}]

6: Increment t, and repeat from step 2 if $t < k$

After all the iterations, we can find correct sparse signals.

A. Compressive Sensing Problem

Let us assume D to be a k-sparse signal of length n. Let ϕ be the measurement matrix projected onto the original signal, D. Measurement matrix (ϕ) must be incoherent with the basis of the sparse signal, D. If D is not sparse in its original bases, it can be transformed to another domain in which the signal is sparse. Then the measurement matrix has to be uncorrelated with the signal in the transformed domain [38]. The size of ϕ is $m \times n$, where $m << n$ and represents the number of measurements. $\hat{D}$ is a m-length vector containing the measurements obtained by the projection of ϕ onto D. Therefore, signal need to be converted to a transformed basis, ψ to induce sparsity and $\hat{D}$ is obtained as:

$$\hat{D} = \phi \psi D = \phi D \tag{1}$$

where ϕ needs to be chosen such that the restricted isometric property (RIP) of order $2k$ is satisfied, where k represents the sparsity of the signal, D.

B. OMP CS Reconstruction Algorithm

OMP is a greedy algorithm, it finds the sparsest solution iteratively by computing support of D and subtracting it from measurement vector S at every iteration. OMP has three different phases, *Identification, Augmentation* and *Residual Update*. In identification phase, index i of highest magnitude of $\phi * R$ is chosen as potential vector to find closest approximation to D. At each iteration, index i is added to the list of estimated support vectors in augmentation phase. The residual update phase generates residual for next iteration. In residual update phase, formed augmented matrix Q is used in least square regression model to find linear relationship between augmented matrix Q and measured vector S. Finally, the amount of contribution that column S provides is subtracted

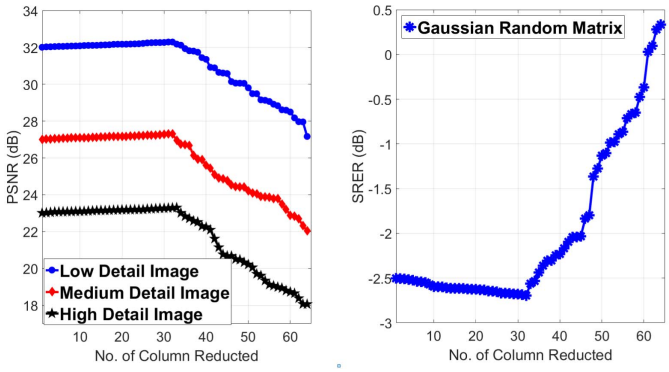

Fig. 1. Performance of Thresholding Technique, with $m = 30\%$ of the signal size and $k = 32$.

to obtain a residue. The OMP algorithm takes k iterations to determine correct set of columns [39].

C. Thresholding Technique for OMP

Identification phase is a severe latency bottleneck on reconstructing signal [16], [23], [33], [40], [41]. Therefore, complexity of the identification phase can be reduced by updating ϕ matrix at each iteration. OMP algorithm iterates k times therefore at each iteration, identification phase calculates dot product of $\phi * R$ and requires $n \times m$ multiplications and $n \times (m-1)$ additions. Jerome *et al.* [40] implements soft threshold technique to update ϕ matrix based on average of resulting dot product vector, whereas Donoho *et al.* [16] introduces hard threshold technique, in which threshold is chosen off-line based on the assumption of Gaussianity. Most of the thresholding techniques proposed previously are based on online thresholding technique which adds new hardware modules, however in this paper we perform offline parameter selection for thresholding which is based on signal size. The soft threshold technique requires additional hardware to calculate average and find minimum averaged column, whereas predetermining threshold affects application flexibility. Also, [40] is the only paper which implements thresholding technique on hardware. In this paper, we implement low cost column reduction phase to the OMP algorithm which updates Ø matrix by eliminating columns that are not contributing in forming residual R at each iteration. Therefore in k^{th} iteration, there will be only $(n-k\times p)\times m$ multiplications. Identification is performed by finding highest index of $\phi * R$, therefore at each step p columns of ϕ with lowest magnitude can be eliminated. In this work, we determine p columns to be eliminated at each iteration. Fig. 1 shows effect of column reduction phase p on both types of input signals with respect to PSNR and SRER of the reconstructed signal, and latency of identification phase of OMP algorithm. To the best of our knowledge, this is the first work which implements hard-thresholding on hardware with satisfactory range of PSNR and SRER.

D. Gradient Descent OMP

Least square kernel is the most computationally intensive kernel among all in OMP algorithm. Least square kernel consists of matrix inversion module which increases the hardware area and power consumption. We implement stochastic gradient descent technique to avoid expensive matrix inversion based least square calculations. Stochastic gradient descent is first order iterative based function minimizing technique in which steps α is taken towards positive gradient of the function. Garg *et al.* [42] implements gradient descent technique with hard thresholding for sparse recovery, whereas in this work we implement stochastic gradient descent technique with mean square error as convergence parameter. In case of large "k" (sparsity) convergence of gradient descent in [42] takes long time as compared to our proposed stochastic gradient descent. To the best of our knowledge none of the previous work on gradient descent based matching pursuit is targeted to OMP algorithm and solving hardware complexity issue and this is the first hardware implementation of stochastic gradient descent based OMP algorithm. Algorithm 3 shows the integration of gradient descent function with OMP algorithm.

Algorithm 3 GD-OMP Reconstruction Algorithm

1:**Initialization**
- $R_0 = S$, $\Lambda_0 = \emptyset$, $Q_0 = \emptyset$ and $t = 0$

2:**Identification**
- Find Index $\lambda_t = max_{j=1\ldots n}$ subject to $| < \phi_j R_{t-1} > |$

3:**Augmentation**
- Update $\Lambda_t = \Lambda_{t-1} \bigcup \lambda_t$
- Update $Q_t = [Q_{t-1}\ \phi_{\Lambda_t}]$

4:**Residual Update**
- Function minimization using Gradient Descent $D_t = Stochastic\_Gradient\_Descent(Q_t, \theta_{Initialize}, S, \alpha)$
- Calculate new approximation: $\alpha_t = Q_t\ D_t$
- Calculate new residual: $R_t = S\text{-}\alpha_t$

5: Increment t, and repeat from step 2 if $t < k$. After all the iterations, we can find correct sparse signals.

Stochastic Gradient Descent Function

1:**Initialization**
- Inputs: Q_t, $\theta_{Initialize}$, S, α, $i = 0$
- Output: θ

2: **Gradient Update**
- $\theta_i = \theta_i - \alpha \sum_{j=1}^{j=m} [\theta * (Q^j) - S^j] * Q_i^j$;

3: **Check Convergence**

Cost Computation using Mean Square Error analysis

4: If Cost $> \gamma$, Increment i, and repeat from step 2

E. Comparison of Algorithms

We perform MATLAB® simulations to compare performance of tOMP, and GDOMP with OMP, Subspcae Pursuit (SP), and Look Ahead OMP (LAOMP) CS reconstruction algorithms in terms of PSNR and SRER.

1) Performance Metric:

- Signal to Reconstruction Error ratio is a mean square error between original signal D and reconstructed

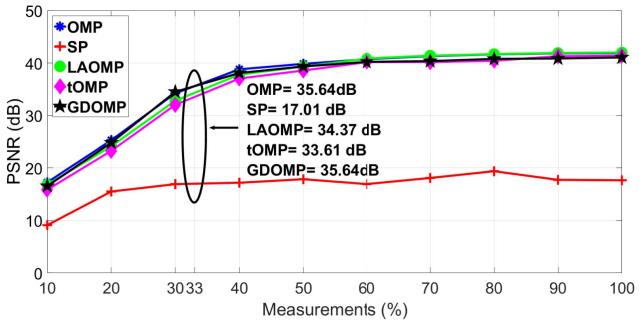

Fig. 2. Comparison of tOMP and GDOMP algorithms with three different reconstruction algorithms including OMP, Subspace Pursuit (SP), and Look Ahead OMP (LAOMP) with respect to PSNR of reconstructed image with $k = 8$.

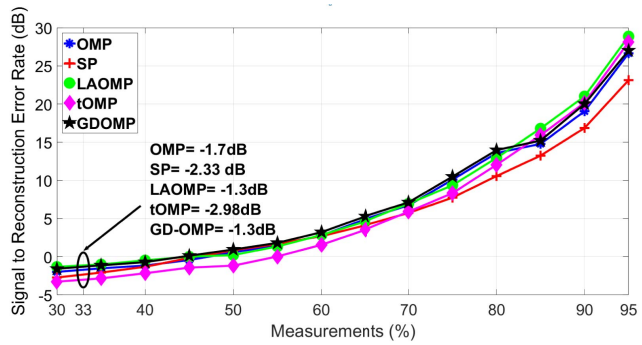

Fig. 3. Comparison of tOMP and GDOMP algorithms with three different reconstruction algorithms including OMP, Subspace Pursuit (SP), and Look Ahead OMP (LAOMP) with respect to SRER of reconstructed Gaussian input signal length 256 with $k = 8$.

signal $\tilde{D}$.

$$SRER = 10\ log\ \left(\frac{\mathbb{E}\|D\|_2}{\mathbb{E}\|D - \tilde{D}\|_2}\right) \tag{2}$$

- PSNR is measured via mean square error (MSE) between original image D and reconstructed image $\tilde{D}$.

$$MSE = \frac{1}{mn} \sum_{i=0}^{i=m-1} \sum_{j=0}^{j=n-1} D(i,j) - \tilde{D}(i,j)\Big]^2 \tag{3}$$

$$PSNR = 20\ log_{10}\ \left(\frac{MAX_{pixel}}{\sqrt{MSE}}\right) \tag{4}$$

- Structural Similarity Index measure (SSIM) is perception based model, in this quality of images is calculated based on structural degradation on various windows of an image [43] as shown in Fig. 4.

2) Experimental Set-Up: For Gaussian signal reconstruction experiments are performed with $n = 512$, $k = 8$, and $T = 1000$ signal sizes.

(i) Randomly generate sensing matrix $(m \times n)$ using Gaussian source [14].
(ii) Randomly generate $k - sparse$ vector using zero-one sparse matrices or Gaussian sparse matrices.
(iii) For each vector, compute measurement vector $S = \phi D$ and apply to CS reconstruction algorithms independently.

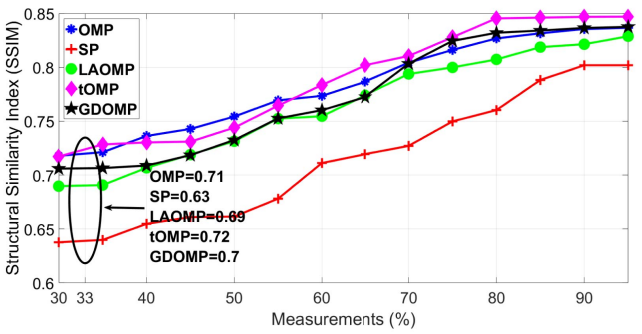

Fig. 4. Comparison of tOMP and GDOMP algorithms with three different reconstruction algorithms including OMP, Subspace Pursuit (SP), and Look Ahead OMP (LAOMP) with respect to Structural Similarity Index (SSIM) of reconstructed image with $k = 8$.

(iv) Repeat step (i to iii) for T times ($T = 1000$). Then evaluate algorithm performance using SRER.
(v) Repeat all steps for a new measurement.

For image reconstruction experiments are performed on different level of information content of $n \times n$[1] where $n = 128$, $k = 8$, $m = 42$ and $T = 1000$.

(i) Randomly generate sensing matrix using Gaussian source.
(ii) Measurement vector is computed using wavelet transform and Gaussian signal vector.
(iii) Repeat experiment (step i to ii) for T times ($T = 1000$). Then evaluate algorithm performance using average PSNR.

Fig. 3 shows SRER performance of OMP, tOMP, and GDOMP algorithm. We chose measurements to be from 30% to 95 % of the original signal i.e., m= 100–200. Error rate of tOMP and GDOMP algorithm closely follows error rate of OMP, SP, and LAOMP algorithms for Gaussian input signal length of 256 and sparsity $k = 8$. For measurements between 30% to 50%, error rate of tOMP and GDOMP is in the range of –1.0 dB to 0 dB. Similarly, Fig. 2 shows PSNR analysis of image reconstruction following steps as discussed in experiment set-up. Additionally, Fig. 5 shows fixed point analysis results of OMP, tOMP, and GDOMP algorithm. PSNR of image reconstruction for lower detail image is in the range of 30 dB–42 dB, medium level of detail is 27 dB–34 dB, and higher level of detail is 21 dB–32 dB.

F. Compressive Sensing for Big Data Applications

To demonstrate the efficiency of CS based data reduction for big data applications, we perform experiments on two different applications including object identification and face detection. We implement cascade classifier in openCV [45], [46] consisting of several simple classifier stages, applied to region of interest until the candidate image is passed or rejected [47]. For object identification application we use CALTECH-256 object identification data set [28] consisting of 256 different categories with 30,607 images, and for face detection application

[1]For convenience to explain, we selected row and column size to be same. In real-time streaming data can be of different column and row sizes

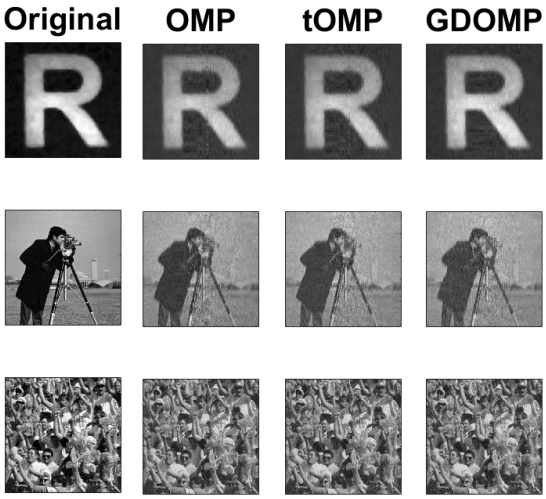

Fig. 5. Comparison of OMP, tOMP, and GD-OMP reconstructed images with fixed point arithmetic and sparsity $k = 8$. Upper image with a low level of detail, middle image with a medium level of detail, and bottom image with a relatively large amount of detail (Image Courtesy: [44]).

we use FDDB [29] data set consisting of 2,845 images with a total of 5,171 faces. The feasibility and efficiency of compressive sensing with OMP reconstruction algorithm for big data processing is evaluated using a MapReduce (with single node setup) implementation using low power NVIDIA Jetson TK1 platform. Table I shows that with 33% measurements and $k = \frac{n}{8}$, using CS OMP for object identification application reduces data up to 50% with losing only 1.7% accuracy. For face detection application we performed experiments on 1,000 images, achieving data reduction up to 48% with average PSNR of 34 dB and losing 0.4% detection accuracy, where accuracy is calculated as shown in equation 5, where t_p–number of true positives, t_n–number of true negatives, f_p–number of false positives, f_n–number of false negatives.

$$Detection\ Accuracy = \frac{t_p + t_n}{t_p + t_n + f_p + f_n}. \quad (5)$$

IV. Hardware Implementation

Increasing attention to approximate computing demands low hardware overhead reconstruction kernel. However, reconstruction of the compressed signal on hardware is challenging since it has interdependent phases and consists of complex matrix inversion, dot product operations, and sort kernels. We implement reconfigurable OMP CS reconstruction algorithm and its modifications called tOMP, and GDOMP as discussed in Section III. The hardware implementation is reconfigurable for different sizes of input signal ranging from $n = 128 \ldots 1024$, sparsity $k = 8 \ldots 64$, with measurements $m = 25\% \ldots 35\%$ of the original signal. In this section, first we discuss effect of number of measurements and sparsity count on important aspects of design i.e. reconstruction time, hardware complexity, and reconstruction quality in terms of PSNR.

We focus on architectural modifications to original OMP as compared to previous work in [24], [25]. The architectural modifications include efficient off-chip memory transfer management using double buffering approach, using LU decomposition to take advantage of symmetry in the matrix inversion, and effective scheduling of temporary data to reduce memory size on-chip.

TABLE I
Case Study: Big Data Applications

Number of Images	Original Data Size (MB)	Data Reduction (%)	Average PSNR (dB)	Detection Accuracy (%)	CS Detection Accuracy (%)
Object Detection on CALTECH-256 [28]					
10,000	263	46.6	32.1	89	88.7
50,000	1,074	46.7	33.9	90.3	88.6
Face Detection on FDDB [29]					
500	217	48.5	34.6	92.1	90.8
1,000	434	48.7	32.1	92.8	91.2

A. Overview of Hardware Complexity of OMP CS Reconstruction Algorithm

In CS matching pursuit algorithms, there are different strategies for choosing number of measurements m for exact recovery of a signal. Increase in number of measurements will guarantee exact recovery of signal but it will increase size of measured signal (S) and measurement matrix (ϕ), thus increasing hardware complexity in terms of operator count, and memory requirements. Fig. 6 shows that increment in number of measurement will have linear growth in hardware complexity and reconstruction time. In case of large number of measurements, reconstruction time is dominated by dot product kernel whereas, higher sparsity count suggests that least square kernel is a bottleneck. Increase in sparsity count not only adds extra number of iterations, but also increases complexity of least square kernel, thus increasing reconstruction time exponentially and linear increase in hardware complexity. We perform experiments to select number of measurements and sparsity as shown in Fig. 6, where hardware complexity is calculated based on number of required multipliers, adders, combinatorial logic, and memory resources. Reconstruction time is reported based on MATLAB® simulation analysis. It is observed that the hardware complexity and reconstruction time is increased as number of measurements and sparsity count is increased however, PSNR of reconstructed image remain constant after certain number of measurements and sparsity count.

Hardware complexity of OMP algorithm is mainly dominated by dot product kernel ($\phi * R$) in identification phase, and least square kernel in residual update phase. Identification and residual update phase, each consumes up to 40%–50% of the total cycles, Septimus *et al.* [33] report 68% and 8%, Blache *et al.* [31] report 41% and 47%, and Kulkarni *et al.* [23], [48] report 44% and 48% cycles for identification and residual update phase respectively where both [31], [33] implements Cholesky decomposition, and Kulkarni *et al.* [23], [48] implements LU decomposition for matrix inversion in residual update phase. tOMP reduces computation complexity of identification phase and GDOMP

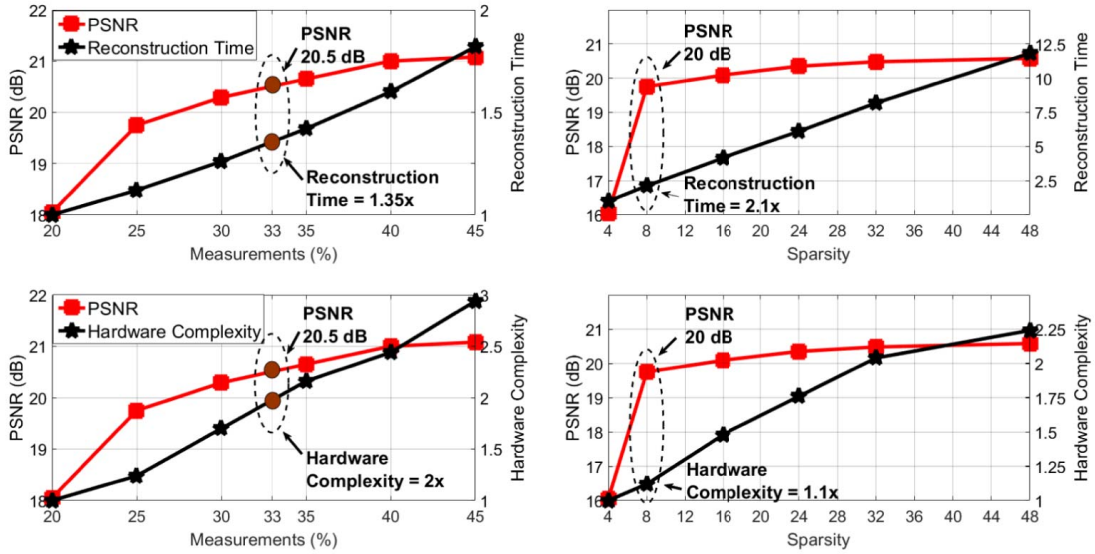

Fig. 6. Trade-off between computational complexity and signal reconstruction quality using measurement and sparsity analysis.

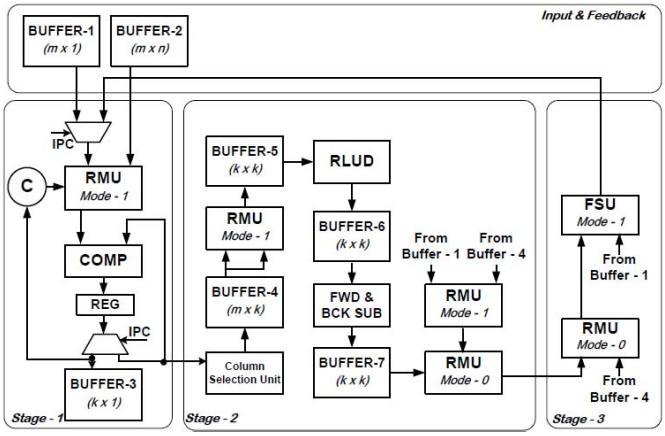

Fig. 7. Block diagram for implementation of OMP Algorithm, where RMU—Reconfigurable Multiplier Unit, COMP—Comparator, FSU—Fixed Point Subtraction Unit, RLUD—Reconfigurable LU Decomposition module.

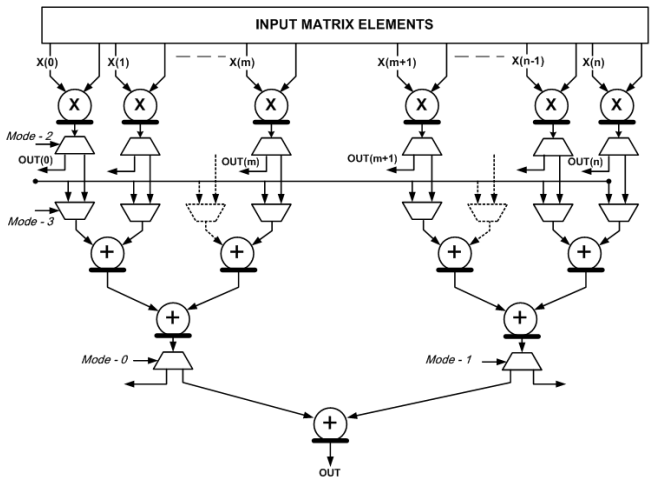

Fig. 8. Reconfigurable fixed-point multiplier unit (RMU).

reduces computational complexity of residual update phase. In tOMP at each iteration lowest magnitude columns C_L which are not contributing to signal reconstruction are reduced, where $C_L = \frac{n}{p}$ is chosen based on application. As discussed in Section III-C for demonstration, we configured $p = k$ for image processing application. Thus for the final iteration, dot product kernel requires to calculate only $k \times m$ multiplications reducing computational complexity up to 75%. In GDOMP algorithm, least square kernel is implemented by using gradient descent technique. To compute gradient descent of a function, it requires $m \times k$ matrix multiplications, m number adders and k number subtractions thus reducing computational complexity. We propose reconfigurable fixed-point multiplier to reuse the operators in order to reduce chip area.

TABLE II
DESCRIPTION OF MODES IN RECONFIGURABLE MULTIPLIER UNIT (RMU)

Modes of Operation	Description
Mode-0	m-stage matrix-vector multiplication
Mode-1	k-stage matrix-vector multiplication
Mode-2	Multiplication of m elements
Mode-3	Summation of m elements

B. Hardware Implementation of Baseline OMP Algorithm

OMP contains iterative interdependent kernels which repeat till the sparsity count making parallel implementation challenging. Therefore in proposed architectures, each phase is implemented in parallel and pipelined to reduce reconstruction time. The interdependent structure allows to reuse kernels and buffers. We propose reconfigurable k-stage and m-stage fixed-point matrix multiplications, the pipeline structure of reconfigurable multiplier allows us double buffering approach.

Algorithm 4 Basic LU Decomposition

Input $X \leftarrow \phi_{squared}$ with elements $x_{i,j}$
Output L, U triangular matrices with elements $l_{i,j}$, $u_{i,j}$
1 **while** X is not empty **do**
2 $p' \leftarrow$ number of rows of X
3 **for** $1 < i < p'$ **do**
4 $l_{i1} \leftarrow \frac{x_{i1}}{x_{11}}$
5 **for** $1 < j < p'$ **do**
6 $u_{1j} \leftarrow x_{1j}$
7 **for** $1 < i, j < p'$ **do**
8 $x_{ij} \leftarrow x_{ij} - (l_{i1} \times u_{1j})$
9 X $\leftarrow$ inner sub-matrix of X of size $p' - 1 \times p' - 1$ by removing first row and column

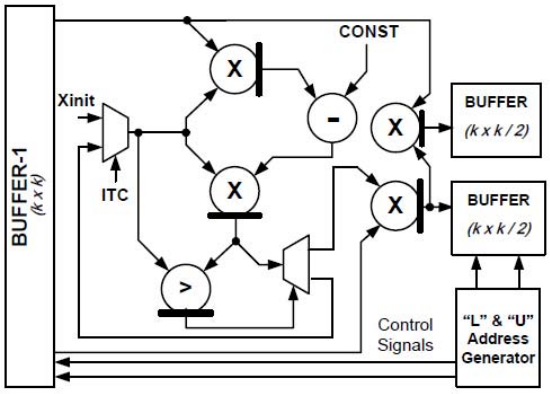

Fig. 9. LU decomposition architecture consisting of Newton-Raphson division method architecture.

Furthermore temporary matrix elements are stored in shared buffers to reduce size of on-chip memory. In this section we discuss architecture for OMP algorithm as shown in Fig. 7 implementation. In thresholding technique, column reduction phase is implemented consisting of sorting dot product vector β. To reduce hardware complexity, stochastic gradient descent function is used in GDOMP instead of Moore-Penrose pseudo inverse. In Sections IV-C and IV-D, we discuss new modified kernels and overall architecture.

1) Implementation of Identification Phase: At the start of execution, measured vector (S) and measurement matrix (ϕ) are stored in buffer 1 and 2 respectively. At first, iteration residual vector (R) is initialized to measured vector (S), therefore buffer-1 is used to calculate dot product in identification phase. To calculate dot product, reconfigurable parallel and pipeline tree structure is used as shown in Fig. 8. Parallel structure reduces execution time required for dot product whereas pipeline structure allows us double buffering approach to hide memory accesses. In double buffering, at each transfer one block of m memory elements is accessed, one memory block is used for computation while other memory block is being transferred. The reconfigurable fixed-point multiplication unit (RMU) has two modes of operation, in mode-0 it finds m-stage matrix multiplication whereas in mode-1, k-stage matrix multiplication is performed. In identification phase, RMU is operated in mode-0, which performs m multiplications and $m-1$ additions. After dot product calculations are performed,

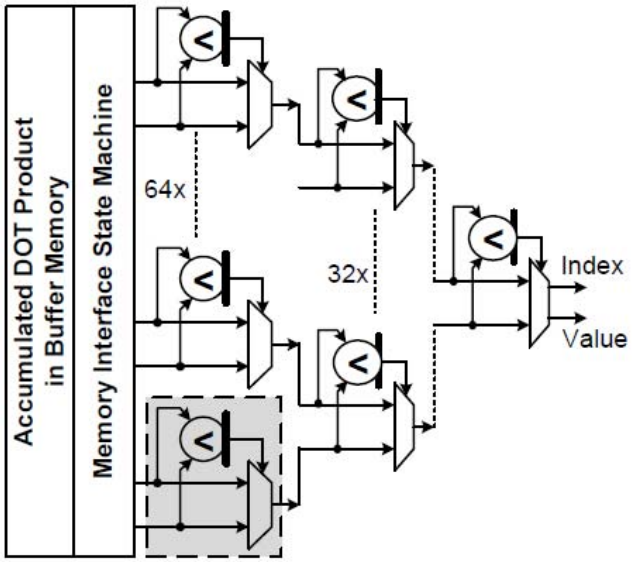

Fig. 10. Merge sort tree architecture with select-value kernels.

simple comparator unit (COMP) is implemented to find maximum value from dot product kernel. At each iteration, COMP compares new dot product kernel value with previous value till n iterations. Index of Highest value is transferred to buffer-3 after n iterations, and issued to column selection unit.

2) Implementation of Augmentation and Residual Update Phase: The column selection unit transfers m elements of ϕ matrix from buffer-2 to buffer-4. Augmented matrix Q from buffer-4 is used for solving least square. The least square kernel is performed in four steps. In step-1 RMU is operated in mode-0 to perform squaring operation, it needs m multiplications and $m-1$ additions, at each iteration row of m elements is added till $m \times k$ matrix. Therefore, it requires to store $k \times k$ elements at the k^{th} iteration in buffer-5. Step-2 inverts the stored square matrix using re-configurable LU decomposition kernel (RLUD).

Algorithm 4 shows the steps to calculate LU decomposition of non-singular $Q_{squared}$ matrix. At each iteration, same steps are applied to sub-blocks of matrix. LU decomposition architecture as shown in Fig. 9 allows considerable amount of parallelism available within each step. To calculate LUD of $k \times k$ matrix, k divisions are required. We perform division operation using Newton-Raphson method shown in equation 6. Newton-Raphson method finds the solution iteratively, to reduce number of iterations we need to calculate initial approximation as shown in equation 7. Fig. 9 shows the Newton-Raphson division method architecture, the steps are interdependent thus cannot be implemented in parallel. It requires 3 multiplication and a comparison to calculate the approximate solution.

$$X_{i+1} = X_i(2 - RX_i) \tag{6}$$

$$X_0 = \frac{(48 - 32R)}{17} \tag{7}$$

While computing inverse of matrix $Q_{squared}$, multiplication of measured vector (S) consisting of m elements in buffer-1 and augmented matrix (Q) in buffer-4 is performed in order to save reconstruction time. Finally to obtain estimation of original signal, in stage-3 RMU is operated in mode-1. At each stage,

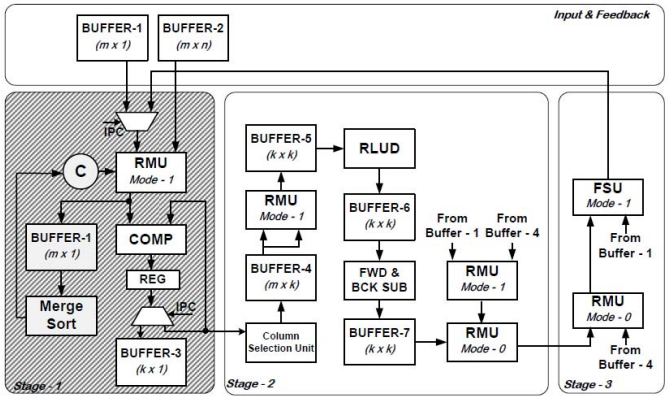

Fig. 11. Block diagram of the Thresholding technique for OMP Algorithm implementation, modifications are performed in Identification stage (stage-1), where RMU—Reconfigurable Multiplier Unit, COMP—Comparator, FSU—Fixed Point Subtraction Unit, RLUD—Reconfigurable LU Decomposition module.

except k^{th} iteration, residual R is calculated by performing subtraction between measured signal S and estimated signal α.

C. Hardware Implementation of Thresholding Technique—tOMP Algorithm

Implementation of OMP algorithm advocates that dot product kernel is bottleneck. Therefore we propose thresholding technique that discards lowest magnitude "$\frac{n}{p}$" columns at each iteration. Fig. 11 shows architecture block diagram for tOMP hardware implementation. To find lowest magnitude of dot product kernel, we implement merge sort algorithm as shown in stage-1. Augmentation and residual update phase are not dependent on column reduction phase thus merge sort is implemented in parallel with them to avoid penalty on reconstruction time. As shown in Fig. 10 we use select-value block to find descending elements at a time, typically smallest element among two is selected whereas other element is used for next cycle. The select-value block is used in merge sort tree architecture where, at each iteration new elements are accessed from buffer-8 using memory wrapper unit. The merge sort is implemented while considering hardware area overhead and required time is kept within execution time bound of augmentation and residual update phase. We implement merge sort for sorting $2 \times \frac{n}{k}$ elements each iteration such that we can obtain index of lowest magnitude column at the start of identification phase. Finally, index of smallest value elements from dot product kernel are transferred to counter (C), the counter will avoid calls for lowest magnitude columns thereby reducing size of dot product calculations for each iteration. In thresholding technique, column reduction phase is added to improve reconstruction time. Identification, augmentation and residual update phases are implemented as same as OMP hardware implementation, discussed in Section IV-B.

D. Hardware Implementation of Gradient Descent—GDOMP Algorithm

The gradient descent kernel is used to reduce overall hardware complexity of the OMP architecture by replacing

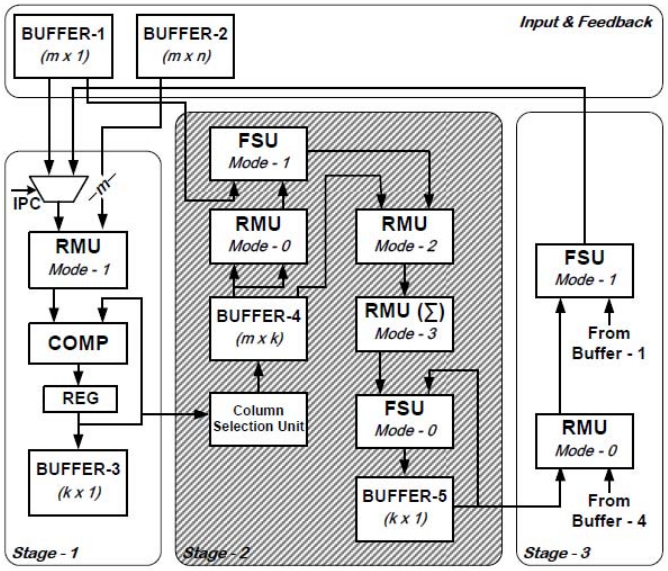

Fig. 12. Block diagram for implementation of the Gradient Descent OMP Algorithm.

Moore-Penrose pseudo inverse with gradient descent function. The modification mainly changes residual update phase, while identification and augmentation phases remain same. Fig. 12 shows architecture for GDOMP algorithm, where stage-2 is changed. The GDOMP function is described in Algorithm 3, it consists of m-stage and k-stage matrix-vector multiplication, m-stage vector subtraction, and m-stage summation unit. We use re-configurable fixed-point multiplier unit (Fig. 8), which operates in four different modes as described in Table II. Reusing multiplication operator reduces chip area, however gradient descent kernel iteratively calculates estimated signal, and repeated till mean square error reduces to satisfactory range, this impact overall reconstruction time. Therefore to reduce iterations, we input range of estimation vector θ. In stage-2, column of measurement matrix (ϕ) is chosen based on index of highest element from dot product kernel. The columns of measurement matrix are augmented at each iteration, the augmented matrix (Q) is stored in buffer-4. RMU is operated in mode-0 and subtraction unit is operated in mode-1 in order to calculate temporary vector, which is multiplied with augmented matrix to obtain new θ_i vector. At each iteration, θ_i is calculated by subtracting new vector from the previous. Finally, new weight vector values are stored on buffer-5. Stage-3 calculates the residue vector R and the loop is repeated till the sparsity count k.

V. ASIC Implementation Results

OMP, tOMP, and GDOMP architectures are implemented using Verilog HDL to describe architecture and hardware, synthesized using Cadence® RTL compiler, and placed and routed using Cadence® SoC encounter. We use a standard-cell-based automatic place and route flow to implement all architectures. All three architectures are implemented with signal size $n = 128 \ldots 1024$, sparsity $k = 8 \ldots 64$, and measurements $m = 25\% \ldots 35\%$ to compare with previous

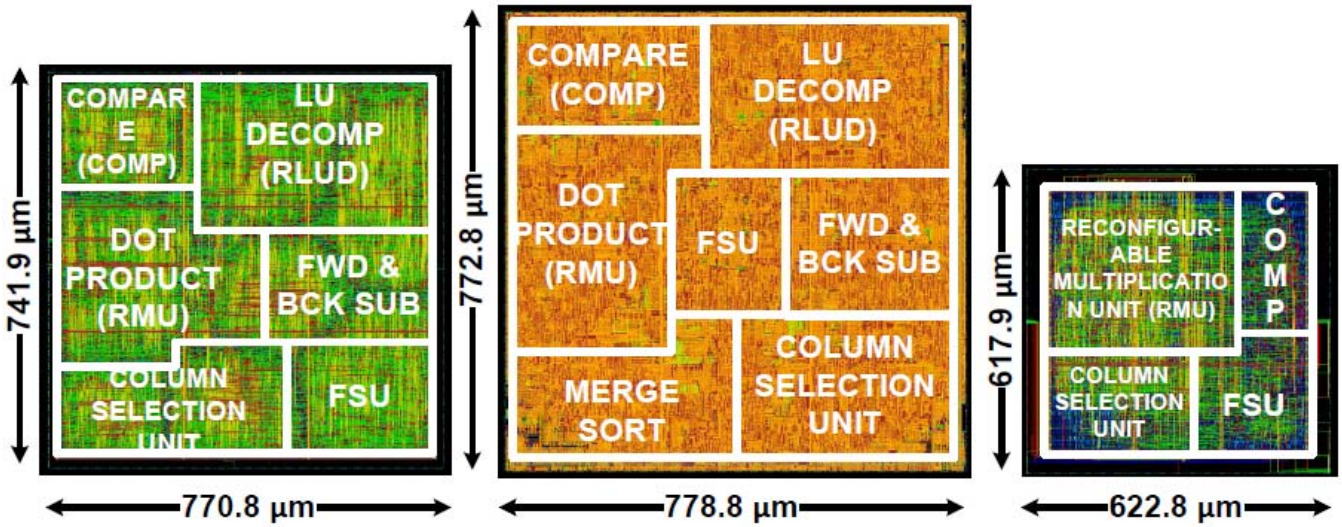

Fig. 13. Post-layout view of hardware implementation of (A) OMP, (B) tOMP, (C) GDOMP architectures in 65 nm, 1V6M TSMC CMOS technology.

TABLE III

EXPERIMENT RESULTS BASED ON MATLAB® IMPLEMENTATION FOR SELECTION OF NUMBER OF FIXED-POINT DATAPATH WORD WIDTH VS. ACCURACY OF DATA RECONSTRUCTION IN TERMS OF PSNR, SRER, SSIM FOR OMP RECONSTRUCTION ALGORITHM, WHERE NR NO RECONSTRUCTION, WE DO NOT CALCULATE ACCURACY STATISTICS WHEN OMP IS UNABLE TO RECONSTRUCT THE DATA

Total bits	8-bit			14-bit			16-bit		
Fractional bits	PSNR (dB)	SRER (dB)	SSIM	PSNR (dB)	SRER (dB)	SSIM	PSNR (dB)	SRER (dB)	SSIM
2	27.1	-1.7	0.6	32.1	0.91	0.67	32.8	0.96	0.69
6	NR	NR	NR	35.6	-1.7	0.71	35.48	-1.67	0.7
8	NR	NR	NR	NR	NR	NR	35.9	-1.7	0.73

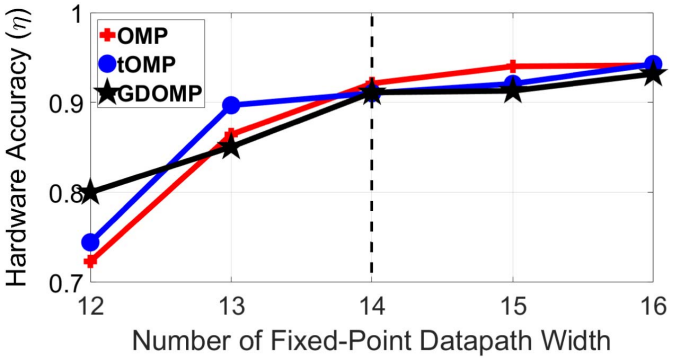

Fig. 14. Hardware Accuracy of OMP, tOMP, and GDOMP algorithm with respect to number of fixed-point bits in datapath word, where η is calculated as shown in (8).

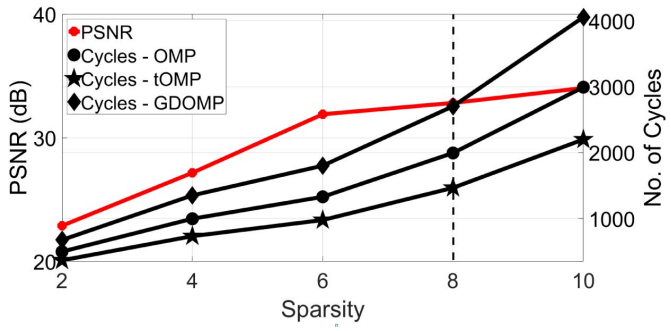

Fig. 15. Analysis of required reconstruction cycles and reconstruction PSNR with respect to sparsity count for OMP, tOMP and GDOMP algorithm on cameraman's image, where $n = 256, m = 64$ as shown in Fig. 5.

work. Fig. 13 shows postroute GDS II layout implementations for the OMP, tOMP, GDOMP architectures.

A. Hardware Accuracy Analysis

OMP, tOMP and GDOMP architectures are implemented using fixed-point arithmetic. We model a custom fixed-point OMP reconstruction algorithm to find optimum datapath word length. To quantify performance gap between floating point MATLAB® implementations and proposed OMP architectures, we calculate average accuracy (η) using equation 8, where α^{sw} and α^{hw} be the estimated signal obtained from MATLAB® software solution and hardware implementations respectively. Fig. 14 shows accuracy of reconstructed signal is dependent on number of fixed-point bits used in datapath. Accuracy of the output is verified at each hardware step along the data path post truncation/saturation. We use Qm.n quantization format for datapath where, "m" is integer bits and "n" is fraction bits. Table III shows, trade-off between number of fixed point data path word and PSNR, SRER, and SSIM for OMP algorithm implementation. The results shown in this paper are based on Q8.6 fixed point precision (i.e. total of 14 bits width with 6 fractional bits). The output of reconfigurable multiplier unit (RMU) is truncated to Q16.12 bits with accuracy of 2.8×10^{-5} with respect to floating point. The output of

TABLE IV

POST-LAYOUT RESULTS ON 65 nm, 1V6M TSMC CMOS TECHNOLOGY FOR OMP, tOMP, AND GDOMP ARCHITECTURES AND ITS COMPARISON WITH PREVIOUSLY PUBLISHED WORK, WHERE n—SIGNAL SIZE, k—SPARSITY, m—NUMBER OF MEASUREMENTS, POWER ANALYSIS IS PERFORMED AT 100 MHz CLOCK FREQUENCY, ENERGY CONSUMPTION IS CALCULATED AS IN (9), AND FIXED-POINT BIT FORMAT IS SHOWN AS Qm.n WHERE "m" NUMBER OF INTEGER BITS, AND "n" FRACTIONAL BITS

Architecture	n,k,m	Format Qm.n	Frequency (MHz)	Reconstruction Time (μs)	Area (μm^2)	Power (mW)	Energy (μJ)	ADP ($\mu m^2 - s$)
Jerome et.al. [24]	256,8,64	10.14	165	13.69	690,000	-	-	9.44
Srikanthan et.al. [25]	256,8,64	32-point	189.7	12.27	638,728	109.8	2.5	7.82
OMP **(This Work)**	256,8,64	8.6	196	10.17	571,928	98.7	1.96	5.79
tOMP **(This Work)**	256,8,64	8.6	198.12	7.4	601,880	103.11	1.51	4.4
GDOMP **(This Work)**	256,8,64	8.6	272.5	11.68	384,886	69.21	1.87	4.5

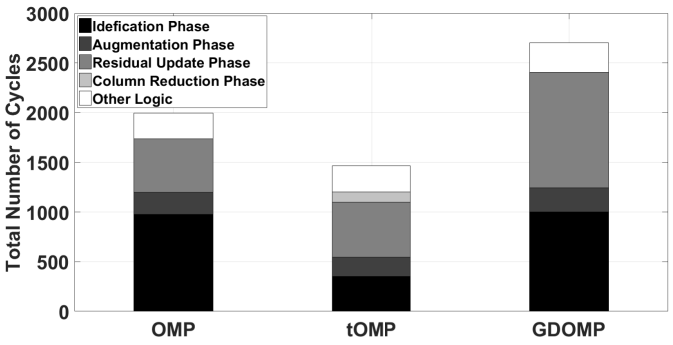

Fig. 16. Total number of cycles required for each kernel in OMP, tOMP, and GDOMP algorithms, where $Reconstruction\ Time = Total\ Number\ Cycles \times (1/Max\ Frequency)$.

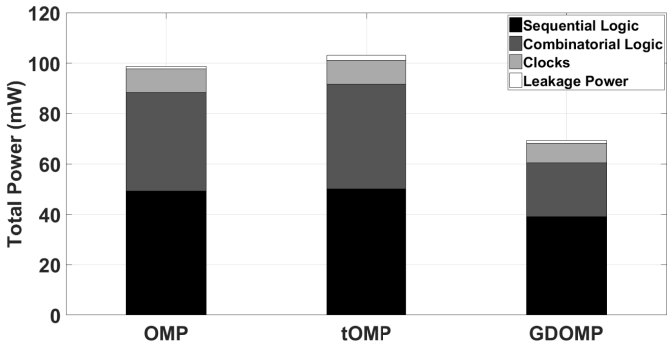

Fig. 17. Post-layout power breakdown for OMP, tOMP, and GDOMP algorithms, power analysis is performed at 100MHz clock frequency.

compared and subtraction (FSU) unit is preserved as similar to its input sizes as Q8.6 and Q16.12 bits respectively. In OMP and tOMP architecture, reconfigurable LU decomposition (RLU) module output is truncated to Q16.12 with accuracy of 1.9×10^{-5}. Fixed-point implementation reduces area and the amount of switching activity on wires and logic gates, consequently reducing the power dissipation. In our previous work [23], we showed the trade-off between no. of fixed point data path word width and area of an architecture, and power dissipation

$$Hardware\ Accuracy\ (\eta) = \frac{1}{n}\sum_{j=1}^{j=n}(\alpha^{sw} - \alpha^{hw}). \quad (8)$$

B. Execution Time Analysis

Figs. 15 and 16 shows execution time analysis for reconstruction of signal $n = 256$ with 25% measurements and sparsity $k = 8$. All three algorithms have similar PSNR values and at sparsity $k = 8$, PSNR starts remaining constant as shown in Fig. 15. The latency overhead is calculated by using a counter, which runs at clock frequency. The counter is enabled when the identification kernel starts executing, it counts the number of clock cycles until the output from the residual update phase after k iterations is available. Kernel specific cycle requirements are presented in Fig. 16, OMP requires 1994 cycles for reconstruction, whereas tOMP requires 27% less and GDOMP takes 35% extra reconstruction cycles. Identification kernel in tOMP reduces up to 54% reconstruction cycles as that of OMP identification kernel. Whereas, residual update kernel in GDOMP takes $2\times$ extra cycles. Though GDOMP increases cycle count, in following section we discuss area reduction and ADP efficiency of GDOMP as compared to OMP algorithm.

C. Area and Power Analysis

The chip area required for OMP, tOMP, and GDOMP architectures is as shown in Fig. 13. OMP takes $0.57mm^2$ chip area with 486K logic gates, tOMP requires $0.60mm^2$ chip area with 512K logic gates, and GDOMP requires 37%, and 33% less chip area as compared to tOMP, and OMP architectures respectively with 320K logic gates.

Fig. 17 shows power breakdown of all three architecture at 100 MHz clock frequency. The switching power is combination of sequential and combinatorial logic activity power and clock power, whereas leakage power is due to CMOS logic gates. In OMP 49% of total power is consumed by sequential logic activity, and in tOMP and GDOMP architectures, 48% and 56% of total power is consumed by sequential logic activity respectively. OMP consumes 98.7 mw, tOMP consumes 103 mw, and GDOMP consumes 69.2 mw power. However the energy consumption for tOMP is 23% less, and for GDOMP its 5% less as compared to OMP architecture, where energy consumption is calculated as shown in equation 9

$$Energy = Power_{at100MHz} \times Reconstruction\ Time_{at100MHz}. \quad (9)$$

For tOMP algorithm, reconstruction time is reduced by 1.4 × and has 22% less energy consumption, however chip area is increased by 5.2% with respect to OMP architecture. Although reconstruction time of GDOMP algorithm is increased by 1.1 ×, it shows tremendous reduction in chip area of 1.5 × and energy consumption by 5%. Therefore tradeoff exists between reconstruction time and chip area when we compare OMP, tOMP, and GDOMP architectures.

D. Comparison with Previous Work

The OMP, tOMP, and GDOMP architectures postlayout results are compared with recently implemented OMP architectures with respect to reconstruction time, ADP, power and energy consumption in Table IV. One of the critical issues with previous implementations is that, it does not provide hardware accuracy analysis, application analysis, and energy analysis. Additionally, previously proposed OMP architectures are not reconfigurable i.e. they do not allow different range of input signal size or number of measurements. Table IV shows comparison between previously published work [24], [25]. Stanislaus *et al.* [24] implemented OMP architecture on 65 nm CMOS technology consisting of QRD process for matrix inversion. Compared to OMP architecture in Stanislaus *et al.* [24], our proposed OMP architecture requires 265 less reconstruction cycles and proposed OMP, tOMP, and GDOMP architectures have 1.6×, 2.1×, and 2× lesser ADP respectively. Meher *et al.* [25] implemented OMP architecture in 65 nm CMOS technology, with reconfigurable 32-point and 5-point inner product unit by reusing arithmetic components. Compared to our OMP architecture implementations in 65 nm technology, Meher *et al.* [25] OMP architecture requires 334 extra cycles and takes 1.2× extra reconstruction time. Compared to OMP, tOMP, and GDOMP architectures, it has 1.3×, 1.8×, and 1.7× higher ADP and consumes 1.27×, 1.65×, and 1.33× more energy respectively.

VI. Conclusion

In this paper we propose two different algorithmic modifications to OMP CS reconstruction algorithm called Thresholding technique-tOMP and Gradient Descent-GDOMP for hardware and time complexity reduction. To evaluate algorithmic efficiency, we perform SRER analysis on tOMP and GDOMP, compared to Subspace pursuit (SP) and OMP algorithm, tOMP has 0.2 dB–1.2 dB and GDOMP has 0.1 dB and 0.12 dB less SRER respectively, when experimented for Gaussian signal input. Additionally we evaluated accuracy of reconstruction using OMP algorithm for big data processing on face detection and object identification benchmarks. The OMP algorithm and its modifications are implemented on 65 nm, 1V6M CMOS technology. We show analysis of OMP, tOMP, and GDOMP algorithm with respect to reconstruction time, energy and chip area. Post place and route results show that, compared to OMP algorithm, tOMP requires 33% times less reconstruction time, and GDOMP consumes 44% times less chip area. Compared to previously published papers, proposed modifications have up to 2.1 times less area-delay product and consumes 40% times less energy.

References

[1] T. Bianchi, V. Bioglio, and E. Magli, "Analysis of one-time random projections for privacy preserving compressed sensing," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 2, pp. 313–327, Feb. 2016.

[2] A. Mirhoseini, A. R. Sadeghi, and F. Koushanfar, "Cryptoml: Secure outsourcing of big data machine learning applications," in *Proc. IEEE Int. Symp. Hardw. Oriented Secur. Trust (HOST)*, May 2016, pp. 149–154.

[3] A. Kulkarni, A. Jafari, C. Shea, and T. Mohsenin, "CS-based secured big data processing on FPGA," in *Proc. IEEE 24th Annu. Int. Symp. Field-Program. Custom Comput. Mach. (FCCM)*, May 2016, p. 201.

[4] A. Majumdar and R. K. Ward, "Robust classifiers for data reduced via random projections," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 40, no. 5, pp. 1359–1371, Oct. 2010.

[5] A. Majumdar and R. K. Ward, "Fast group sparse classification," *Can. J. Electr. Comput. Eng.*, vol. 34, no. 4, pp. 136–144, 2009.

[6] B. Rouhani, E. Songhori, A. Mirhoseini, and F. Koushanfar, "SSketch: An automated framework for streaming sketch-based analysis of big data on FPGA," in *Proc. IEEE 23rd Annu. Int. Symp. Field-Program. Custom Comput. Mach. (FCCM)*, May 2015, pp. 187–194.

[7] A. Mirhoseini, B. D. Rouhani, E. M. Songhori, and F. Koushanfar, "Perform-ML: Performance optimized machine learning by platform and content aware customization," in *Proc. 53rd Annu. Design Autom. Conf. (DAC)*, New York, NY, USA, Jun. 2016, pp. 20:1–20:6.

[8] A. Jafari, A. Page, C. Sagedy, E. Smith, and T. Mohsenin, "A low power seizure detection processor based on direct use of compressively-sensed data and employing a deterministic random matrix," in *Proc. IEEE Biomed. Circuits Syst. (BioCAS) Conf.*, Oct. 2015, pp. 1–4.

[9] K. Xu, Y. Li, and F. Ren, "An energy-efficient compressive sensing framework incorporating online dictionary learning for long-term wireless health monitoring," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Mar. 2016, pp. 804–808.

[10] J. Zhang, Y. Yan, L. J. Chen, M. Wang, T. Moscibroda, and Z. Zhang, "Impression store: Compressive sensing-based storage for big data analytics," in *Proc. 6th USENIX Conf. Hot Topics Cloud Comput. (HotCloud)*, Berkeley, CA, USA, 2014, p. 1.

[11] A. Ahmad, A. Paul, M. Rathore, and H. Chang, "An efficient multidimensional big data fusion approach in machine-to-machine communication," *ACM Trans. Embeded Comput. Syst.*, vol. 15, no. 2, pp. 39:1–39:25, Jun. 2016.

[12] A. Mirhoseini, E. M. Songhori, B. D. Rouhani, and F. Koushanfar, "Flexible transformations for learning big data," in *Proc. ACM SIGMETRICS Int. Conf. Meas. Modeling Comput. Syst.*, New York, NY, USA, 2015, pp. 453–454.

[13] D. Needell and R. Vershynin, "Signal recovery from incomplete and inaccurate measurements via regularized orthogonal matching pursuit," *IEEE J. Sel. Topics Signal Process.*, vol. 4, no. 2, pp. 310–316, Apr. 2010.

[14] S. Chatterjee, D. Sundman, and M. Skoglund, "Look ahead orthogonal matching pursuit," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2011, pp. 4024–4027.

[15] P. Swamy, S. Ambat, S. Chatterjee, and K. Hari, "Reduced look ahead orthogonal matching pursuit," in *Proc. 20th Nat. Conf. Commun. (NCC)*, Feb. 2014, pp. 1–6.

[16] D. L. Donoho, Y. Tsaig, I. Drori, and J.-L. Starck, "Sparse solution of underdetermined systems of linear equations by stagewise orthogonal matching pursuit," *IEEE Trans. Inf. Theory*, vol. 58, no. 2, pp. 1094–1121, Feb. 2012.

[17] A. Korde, D. Bradley, and T. Mohsenin, "Detection performance of radar compressive sensing in noisy environments," *Proc. SPIE*, vol. 8717, May 2013.

[18] N. Vaswani and W. Lu, "Modified-CS: Modifying compressive sensing for problems with partially known support," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2009, pp. 488–492.

[19] M. Andrecut. (Sep. 2008). "Fast GPU implementation of sparse signal recovery from random projections." [Online]. Available: https://arxiv.org/abs/0809.1833

[20] Y. Fang, L. Chen, J. Wu, and B. Huang, "GPU implementation of orthogonal matching pursuit for compressive sensing," in *Proc. IEEE 17th Int. Conf. Parallel Distrib. Syst. (ICPADS)*, Dec. 2011, pp. 1044–1047.

[21] M. Andrecut, "Fast GPU implementation of sparse signal recovery from random projections," *Eng. Lett.*, pp. 151–158, 2009.

[22] A. Kulkarni, J. L. Stanislaus, and T. Mohsenin, "Parallel heterogeneous architectures for efficient OMP compressive sensing reconstruction," *Proc. SPIE*, vol. 9109, p. 91090G, May 2014.

[23] A. M. Kulkarni, H. Homayoun, and T. Mohsenin, "A parallel and reconfigurable architecture for efficient OMP compressive sensing reconstruction," in *Proc. 24th Great Lakes Symp. VLSI (GLSVLSI)*, New York, NY, USA, 2014, pp. 299–304.
[24] J. Stanislaus and T. Mohsenin, "High performance compressive sensing reconstruction hardware with qrd process," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2012, pp. 29–32.
[25] P. Meher, B. Mohanty, and T. Srikanthan, "Area-delay efficient architecture for MP algorithm using reconfigurable inner-product circuits," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Jun. 2014, pp. 2628–2631.
[26] H. Rabah, A. Amira, B. Mohanty, S. Almaadeed, and P. Meher, "FPGA implementation of orthogonal matching pursuit for compressive sensing reconstruction," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, to be published.
[27] G. Huang and L. Wang, "High-speed signal reconstruction with orthogonal matching pursuit via matrix inversion bypass," in *Proc. IEEE Workshop Signal Process. Syst. (SiPS)*, Oct. 2012, pp. 191–196.
[28] G. Griffin, A. Holub, and P. Perona, "Caltech-256 object category dataset," California Inst. Technol., Pasadena, CA, USA, Tech. Rep., 2007. [Online]. Available: http://authors.library.caltech.edu/7694
[29] V. Jain and E. Learned-Miller, "FDDB: A benchmark for face detection in unconstrained settings," Univ. Massachusetts Amherst, Amherst, MA, USA, Tech. Rep. UM-CS-2010-009, 2010.
[30] A. Septimus and R. Steinberg, "Compressive sampling hardware reconstruction," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Jun. 2010, pp. 3316–3319.
[31] P. Blache, H. Rabah, and A. Amira, "High level prototyping and FPGA implementation of the orthogonal matching pursuit algorithm," in *Proc. 11th Int. Conf. Inf. Sci. Signal Process. Appl. (ISSPA)*, Jul. 2012, pp. 1336–1340.
[32] G. Huang and L. Wang, "Soft-thresholding orthogonal matching pursuit for efficient signal reconstruction," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2013, pp. 2543–2547.
[33] A. Septimus and R. Steinberg, "Compressive sampling hardware reconstruction," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Jun. 2010, pp. 3316–3319.
[34] J. Constantin *et al.*, "Tamarisc-CS: An ultra-low-power application-specific processor for compressed sensing," in *Proc. IEEE/IFIP 20th Int. Conf. VLSI Syst. Chip (VLSI-SoC)*, Oct. 2012, pp. 159–164.
[35] Y. Yan *et al.*, "Distributed outlier detection using compressive sensing," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, New York, NY, USA, 2015, pp. 3–16.
[36] A. Kulkarni, C. Shea, T. Abtahi, and T. Mohsenin, "Low overhead CS-based heterogeneous framework for big data acceleration," *Trans. Embedded Comput. Syst*.
[37] W. Dai and O. Milenkovic, "Subspace pursuit for compressive sensing signal reconstruction," *IEEE Trans. Inf. Theory*, vol. 55, no. 5, pp. 2230–2249, May 2009.
[38] E. J. Candès and M. Wakin, "An introduction to compressive sampling," *IEEE Signal Process. Mag.*, vol. 25, no. 2, pp. 21–30, Mar. 2010.
[39] J. A. Tropp and A. C. Gilbert, "Signal recovery from random measurements via orthogonal matching pursuit," *IEEE Trans. Inf. Theory*, vol. 53, no. 12, pp. 4655–4666, Dec. 2007.
[40] J. L. V. M. Stanislaus and T. Mohsenin, "Low-complexity FPGA implementation of compressive sensing reconstruction," in *Proc. Int. Conf. Comput., Netw. Commun. (ICNC)*, Jan. 2013, pp. 671–675.
[41] L. Bai, P. Maechler, M. Muehlberghuber, and H. Kaeslin, "High-speed compressed sensing reconstruction on FPGA using OMP and AMP," in *Proc. 19th IEEE Int. Conf. Electron., Circuits Syst. (ICECS)*, 2012, pp. 53–56.
[42] R. Garg and R. Khandekar, "Gradient descent with sparsification: An iterative algorithm for sparse recovery with restricted isometry property," in *Proc. 26th Annu. Int. Conf. Mach. Learn. (ICML)*, New York, NY, USA, 2009, pp. 337–344.
[43] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.
[44] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 3rd ed. Upper Saddle River, NJ, USA: Prentice-Hall, 2006.
[45] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., (CVPR)*, vol. 1. Dec. 2001, pp. I-511–I-518.
[46] R. Girshick, "Fast R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1440–1448.
[47] *Haar Feature-Based Cascade Classifier for Object Detection*, accessed on Feb. 2, 2016. [Online]. Available: http://docs.opencv.org/
[48] A. Kulkarni and T. Mohsenin, "Accelerating compressive sensing reconstruction OMP algorithm with CPU, GPU, FPGA and domain specific many-core," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2015, pp. 970–973.
[49] A. Kulkarni, A. Jafari, C. Sagedy, and T. Mohsenin, "Sketching-based high-performance biomedical big data processing accelerator," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2016, pp. 1138–1141.
[50] A. Mirhoseini, B. D. Rouhani, E. M. Songhori, and F. Koushanfar, "Perform-ML: Performance optimized machine learning by platform and content aware customization," in *Proc. 53rd Annu. Design Autom. Conf. (DAC)*, New York, NY, USA, Jun. 2016, pp. 20:1–20:6.

Amey Kulkarni received the M.Tech degree in VLSI design from Vellore Institute of Technology (VIT), Vellore, India, in 2010. He is currently working toward the Ph.D. degree in computer engineering at the University of Maryland, Baltimore County, MD, USA. His main research focus is on designing real-time and low power hardware architecture that are resource efficient, cognitive and trustworthy. He is Key Designer of the 192 core many-core chip called Power Efficient Nano Cluster (PENC). He is actively researching strategies to efficiently perform big data acceleration using hardware platforms particularly FPGA and multiprocessor system-on-chip (MPSoC). During his academic career, he has published over 10 papers in peer reviewed conferences and journals. He also served as reviewer for IEEE International Symposium on Circuits and Systems (ISCAS) and Springer's *Arabian Journal for Science and Engineering*. He worked for two years as a VLSI engineer at Silicon Interfaces and 3-D microsystems, India.

Tinoosh Mohsenin received the M.S. degree from Rice University, Houston, TX, USA, in 2004, and the Ph.D. degree from the University of California, Davis, CA, USA, in 2010, both in electrical and computer engineering. She is an Assistant Professor in the Department of Computer Science and Electrical Engineering at University of Maryland Baltimore County, MD, USA, where she directs the Energy Efficient High Performance Computing (EEHPC) Lab. Her research focus is on the development of highly accurate high performance processors for machine learning, knowledge extraction and data sparsification and recovery that consume as little energy as possible. Prof. Mohsenin has over 60 peer-reviewed journal and conference publications. She currently leads a number of research projects including the design of next generation wearable biomedical processors, hardware accelerators for deep learning and convolutional neural networks, real time brain signal artifact removal and processing for brain computing interface and assistive devices, which are all funded by National Science Foundation (NSF), Army Research Lab (ARL), Boeing and Xilinx. She has served as Associate Editor for the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—PART I: REGULAR PAPERS (TCAS-I) and currently serves as an Associate Editor for the IEEE TRANSACTIONS ON BIOMEDICAL CIRCUITS AND SYSTEMS (TBioCAS). She has served as technical program committee member of the International Solid-State Circuits Student Research (ISSCC-SRP), IEEE Biomedical Circuits and Systems (BioCAS), IEEE Circuits and Systems (ISCAS) and International Symposium on Quality Electronic Design (ISQED). She also serves as secretary of IEEE P1890 WG on Error Correction Coding for Non-Volatile Memories.