Accelerating Compressive Sensing Reconstruction OMP Algorithm with CPU, GPU, FPGA and Domain Specific Many-Core

Amey Kulkarni and Tinoosh Mohsenin
Department of Computer Science & Electrical Engineering
University of Maryland, Baltimore County
ameyk1@umbc.edu, tinoosh@umbc.edu

Abstract*—Compressive Sensing (CS) signal reconstruction can be implemented using convex relaxation, non-convex, or local optimization algorithms. Though the reconstruction using convex optimization, such as the Iterative Hard Thresholding algorithm, is more accurate than matching pursuit algorithms, most researchers focus on matching pursuit algorithms because they are less computationally complex. Orthogonal Matching Pursuit (OMP) is a greedy algorithm, which solves the problem by choosing the most significant variable to reduce the least square error. In this paper, we propose an efficient parallel architecture for OMP CS reconstruction. For architecture implementation, we perform measurement and sparsity analysis to reduce the complexity. The proposed architecture is platform independent and is implemented on 7 different platforms including general purpose CPUs, GPUs, a Virtex-7 FPGA and a domain specific many-core. The implementation results indicate that reconstruction time on FPGA is improved by 3× compared to previous FPGA implementation, whereas GPU implementation is 4× faster than the previously proposed GPU-based OMP architecture. The CPU implementation is 6× faster, compared with previous CPU-based implementation. The domain specific many-core acheives 24 times faster reconstruction time when compared to both GPU and CPU implementations.

I. Introduction

Compressive Sensing (CS) has received a significant attention due to the reduction in sampling and measurements, and therefore, resulting in less time and power consumption of signal acquisition. Various applications, such as signal de-noising, satellite remote sensing, image sensing for SAR echo mode and mobile communication, are aiming to use CS. Current research is focused on both sampling techniques and enhancing the reconstruction algorithms in terms of accuracy [1], [2] and complexity reduction on different platforms [3], [4], [5], [6], [7], [8], [9], [10].

Though CS has several advantages, CS reconstruction techniques are very complex and computational intensive. In this paper, we propose a platform independent architecture for CS reconstruction to reduce reconstruction time. The proposed architecture is implemented on various platforms including general purpose CPUs, GPUs, a Virtex-7 FPGA and a domain specific many-core. The structure of this paper is as follows: Section II briefly explains CS reconstruction OMP algorithm; Section III discusses processing platforms used for implementing the proposed architecture. Section IV describes the proposed architecture. Finally, section V, discusses and compares our implementation results on different platforms, while comparing it with previous work.

II. Compressive Sensing and OMP algorithm

The basic theory behind CS lies in solving equation 1. Let ϕ be the measurement matrix of dimension $M \times N$, where M is the number of measurements to be taken and N is the length of the signal, and let x be a k-sparse signal of length N. Multiplying these two vectors yields y of length M, which contains the measurements obtained by the projection of ϕ onto x.

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

OMP takes two inputs: the measured signal (y) and the measurement matrix (ϕ). At each iteration (t), column of ϕ is chosen which is most strongly correlated with y. Least square algorithm is used to obtain a new signal estimate. In the next step, the amount of contribution that column y provides is subtracted to obtain a residue, which is used for the next iteration. Finally, after k iterations, the correct set of columns is determined [11].

Algorithm 1 OMP Reconstruction Algorithm

```
1: Initialize $R_0 = y$, $\phi_0 = \emptyset$, $\Lambda_0 = \emptyset$, $\Phi_0 = \emptyset$ and $t = 0$
2: Find Index $\lambda_t = max_{j=1...n}$ subject to $| < \phi_j R_{t-1} > |$
3: Update $\Lambda_t = \Lambda_{t-1} \bigcup \lambda_t$
4: Update $Q_t = [Q_{t-1} \ \phi_{\Lambda_t}]$
5: Solve the Least Squares Problem
$x_t = \min_x \| y - Q_y \ x \|^2$
6: Calculate new approximation: $\alpha_t = \Phi_t \ x_t$
7: Calculate new residual: $R_t = y - \alpha_t$
8: Increment t, and repeat from step 2 if t<k
After all the iterations, we can find correct sparse signals.
```

The variables used in the algorithm are defined below:

- N× N = Images Size (e.g $128 \times 128 ... 768 \times 768$)
- M = Measurements (e.g 42...252)
- k = Sparsity (e.g 32)
- R = Residual Matrix ($size : M \times 1$)
- ϕ = Measurement Matrix ($size : M \times N$)
- λ = Maximum Index after Dot Product
- t = No. of iterations (k)

III. Processing Platform Architecture

A. Off-the-shelf Processors

1) BlueGrid Cluster: BlueGrid is a cluster located at UMBC's HPC center which hosts 13 IBM BladeCenter HS22 servers comprising 104 cores and a few hundred gigabytes of random access memory (RAM) with Intel Xeon processors running Red Hat Linux (RHL), a 64-bit operating system. We use the OpenMP tool set to perform experiments on *BlueGrid*.

2) GPUs and CPUs: We use two different GPU families (Tesla M2070 and GeForce 640) and CPU platforms to show that the proposed architecture has the least reconstruction time compared to previous published work. To perform our experiments we used CUDA and OpenCL. The results show that although OpenCL and CUDA have similar platform, memory, and programming models, CUDA implementation is faster than OpenCL. Thus we use CUDA 6.0 for experiments. Proposed architecture is also implemented in parallel on the Intel Xeon using OpenCL and in serial on Intel i7 using Matlab.

3) FPGA: The proposed re-configurable and parallel architecture is implemented on the Virtex-7 XC7VX485T. The architecture is implemented by using fixed point arithmetic. The architecture is fully placed and routed on the FPGA. The detailed evaluation of the performance is evaluated in section V.

B. Domain Specific Many-Core

The domain specific many-core architecture consists of in-order processors that has a 6 stage pipeline, a RISC-like DSP instruction set, and a Hardvard memory model. It consists of 64 low-power small cores [12], [13]. Each core operates on a 16-bit data-path with a minimal instruction and data memory suitable for task level parallelism. Moreover, the cores have a limited low complexity instruction set to reduce area and power. The cores communicate through a simple, scalable hierarchical network that reduces the number of hops in communication. Each core and router was synthesized, placed and routed in a 65 nm CMOS process. Our many-core integrated development environment (IDE) and simulator, designed using Eclipse and Xtext Plug-in, provides the number of cycles, instructions and data memory used per core. For each kernel, different levels of parallelism are investigated to analyze the run time and energy consumption.

IV. Proposed Work

Figure 1 shows that the OMP algorithm can be partitioned in three main kernels, such as Dot product, sort and least square. These three kernels are interdependent, hence efficient parallel implementation of an algorithm is complex. We reduce hardware complexity of the algorithm based on our analysis on sparsity, number of measurements, and fixed point hardware.

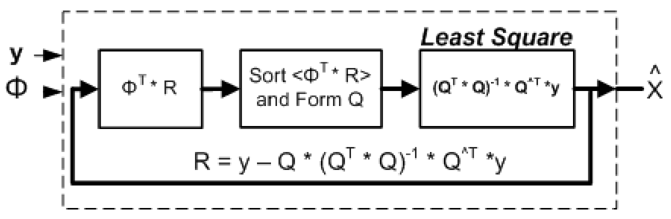

Fig. 1. Basic Block Diagram for OMP Reconstruction Algorithm

TABLE I
PSNR and Sparsity Analysis numbers for variety of Image Sizes with OMP Reconstruction Algorithm

Image Sizes	Sparsity k	PSNR (dB)		
		Low Detail Image	Medium Detail Image	High Detail Image
256×256	8	27.22	25.36	23.43
256×256	**32**	**34.70**	**25.02**	**23.07**
256×256	48	34.89	24.20	22.02
384×384	8	21.24	20.86	19.07
384×384	**32**	**22.88**	**19.97**	**18.34**
384×384	48	22.91	18.67	17.02
512×512	8	21.19	16.20	14.45
512×512	**32**	**25.65**	**15.52**	**14.05**
512×512	48	25.71	14.89	13.57

A. Sparsity Analysis

From Figure 1, it's observed that least square is the most complex kernel in OMP, which consists of Q matrix multiplication, transpose, and inversion operations. The size of Q matrix depends on the number of iterations, which is a function of sparsity k (a predefined number). In this paper, we experimented different images based on information content with different sizes. We observed satisfactory range of PSNR for different Sparsity count. The experiments are repeated 100 times to measure accurate PSNR of the reconstructed image. Table I shows different sizes of images ($N \times N$) with different sparsity and PSNR results. The reconstructed images are shown in Figure 4. It can be observed from Table I that, variation in sparsity assumption has minimal effect on PSNR. Nonetheless from the hardware perspective, it is advantageous since it minimizes matrix to be inverted, thereby reducing memory transfers and reconstruction time. From the above observations we fixed the sparsity to 32, such that it reduces the hardware complexity while also meeting a satisfactory PSNR for the reconstructed image.

B. Measurement Analysis

In CS matching pursuit algorithms, there are different strategies for choosing measurements m for exact recovery of a signal [14]. Matrix sizes of dot product, least square kernels, and residue calculations are dependent on measurements (m). The dot product kernel remains the same at each iteration, whereas the least square kernel and residue calculation block iterates by sparsity(k) times, changing the Q matrix size at each iteration. For each image, we performed different experiments for various measurement counts with fixed sparsity of 32. Figure 2 shows measurement analysis for low detailed image (Figure 4A). It is observed that the hardware complexity increases with number of measurements, however PSNR of a reconstructed image remains constant after certain number of measurements. Therefore, we chose the optimal number of measurements for each image to maintain satisfactory PSNR for reconstruction, while keeping minimal hardware complexity.

C. Structure of the Proposed Architecture

Figure 3 A shows proposed architecture for CS Reconstruction OMP algorithm. The OMP algorithm has inter-dependent

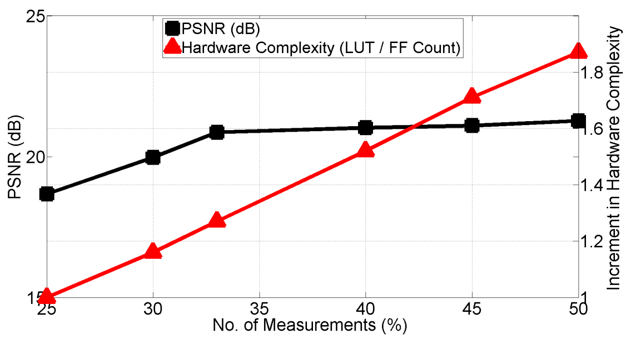

Fig. 2. Increase in Hardware complexity and PSNR of Reconstructed Image of OMP CS Reconstruction algorithm

kernels, which makes parallel implementation complex. For parallel implementation, we first analyze information dependency of each kernel. We determine the necessary computational system for problem solving and distributing such that it uses less resources and reduces reconstruction time. Figure 3B shows the task graph for CS reconstruction OMP algorithm. Sub-tasks are distributed such that the communications among the sub-tasks is less. Furthermore, for each task communication latency is hidden by computations.

For the matrix multiplication kernel, we use the block strip matrix multiplication method. On the FPGA, block strip multiplication is implemented by using tree multiplier to leverage parallel and pipeline architecture. A tree multiplier requires N multipliers and N adders where, N is the number of columns. Thus, the total number of operations come to be $2N - 1$. Trade-offs exist between resource utilization and latency of operations. The architecture is developed such that at every cycle multiplication product is calculated in parallel, while for GPU and CPU, each node computes the multiplication. The sort kernel is used to locate the maximum of $| < \phi R > |$. We implement a binary tree sort algorithm which has complexity of $O(N \log N)$. The kernel is implemented on each node, therefore the complexity is reduced. The algorithm needs N space of memory. We use cache and on-chip memories (BRAMs) to reduce communication overhead. Finally, least square, the most important kernel of the algorithm is implemented by using LU decomposition method. We use block LU algorithm for parallel implementation and MAGMA libraries to calculate LU decomposition for GPU implementation, and resources are reused to reduce the area of the architecture while implementing on hardware.

V. Result Analysis and Comparison

A. Image Quality

In this paper, we use different test images on the basis of information content of the image. Figure 4 shows reconstructed images with 33% measurements and 32 sparsity. It can be observed that, Figure 4A has low level of information content and the reconstruction PSNR is 35 dB, whereas Figure 4B has medium level of information content and the reconstruction PSNR is 26.3 dB, and Figure 4C has relatively large amount of information content and reconstruction PSNR is 24.5 dB.

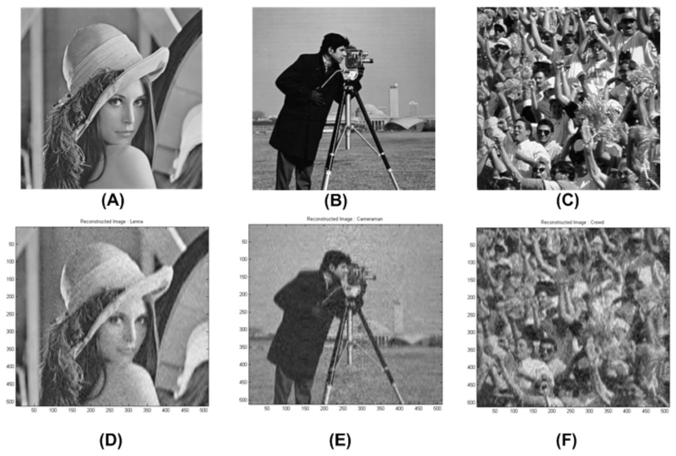

Fig. 4. OMP Reconstructed Images with fixed point arithmetic and Sparsity 32. Left Image with a low level of detail, center Image with a medium level of detail and right Image with a relatively large amount of detail (Image Courtesy: [15])

B. Comparison with Previous Work

In this paper we compare our results with various OMP algorithm architectures on different and appropriate hardware platforms (Table II). From previous FPGA implementation of OMP, the reconstruction time on Xilinx Virtex-5 FPGA for signal size of 128×128 with sparsity 5, is found to be 24 μs [6]. The OMP implementation on Xilinx Virtex-6 FPGA for signal vector of 128 with sparsity 5, requires 16 μs to reconstruct the signal [7]. Whereas, for signal size 128×128 with sparsity 5 and 33% measurements, it takes a total run time of 10 μs [4]. Therefore, compare to previous FPGA implementations of OMP algorithm our implementation is 3× faster.

On the other hand, on nVIDIA GeForce GPU, for signal vector of 1024 with sparsity 12, OMP algorithm takes 37.5 ms to reconstruct the signal. Compare to [10], the proposed architecture is 4× faster on nVIDIA GeForce 640. While comparing with previous CPU implementation [9], the proposed architecture is 6× faster than Intel Xeon. Our domain specific many-core reconstructs 128×128 image in 2.8 ms, which is 11× and 4.4× faster than the same implementation on Intel Core i7 and Intel Xeon, respectively.

The proposed architecture works with different signal sizes and fixed sparsity. Though the sparsity is higher for proposed architecture the comparison table II shows that, the proposed architecture perform better than previous work irrespective of hardware platform.

VI. Conclusions

In this paper, we proposed a platform independent architecture for OMP compressive sensing reconstruction. Several parameters including sparsity, number of measurements and fixed point hardware optimization were analyzed to reduce the hardware complexity. The proposed architecture is implemented on various platforms including general purpose CPUs, GPUs, Virtex-7 FPGA and a domain specific many-core and results are compared. Depending on the platform implementation, the proposed architecture performs 3× to 24× faster than the previously published papers.

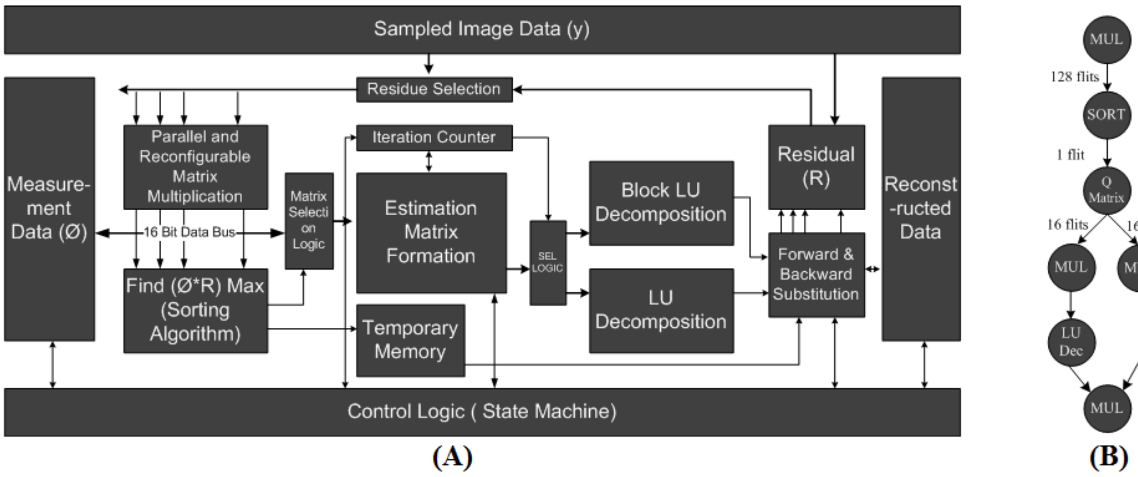

Fig. 3. (A) Proposed Architecture (B) Task Graph for CS Reconstruction OMP algorithm

TABLE II
Comparison with Previous Work

	Signal Length	Spars -ity	Reconstruct Time	Improve -ment
FPGA (Virtex-5) [6]	128×128	5	24μs	Base Architecture
FPGA (Virtex-6) [7]	128×1	5	16μs	1.5×
FPGA (Virtex-5) [4]	128×128	5	10μs	2.4×
ASIC (65nm) [8]	500×1000	-	0.16 ms	0.15×
Intel Core i7 [9]	1024×1	12	68 ms	Base Architecture
nVIDIA GeForce [10]	1024×1	12	37.5 ms	Base Architecture
FPGA Virtex-7 (This Work)	128×128	32	8.97μs	2.67×
	256×256		9.32μs	2.57×
	512×512		10.12μs	2.37×
nVIDIA GeForce (This Work)	128×128	32	11.004 ms	3.4×
	256×256		11.25 ms	3.34×
	512×512		11.4889 ms	3.26×
nVIDIA Tesla (This Work)	128×128	32	47.1 ms	-
	256×256		48.5 ms	-
	512×512		51.7 ms	-
Intel Xeon (This Work)	128×128	32	12.43 ms	5.47×
	256×256		16.8 *ms*	4×
	512×512		51.5 *ms*	1.32×
Intel Core i7 (This Work)	128×128	32	31.2 *ms*	2.17×
	256×256		125 *ms*	-
	512×512		620 *ms*	-
BlueGrid (This Work)	128×128	32	13.11*ms*	-
			12.48 *ms*	-
			13.97 *ms*	-
Custom Many-Core 61 Cores (This work)	128×128	8	2.8 *ms*	11x, 4.4x i7 and Xeon

References

[1] Yuejie Chi and R. Calderbank, "Knowledge-enhanced matching pursuit," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, May 2013, pp. 6576–6580.

[2] Asmita Korde, Damon Bradley, and Tinoosh Mohsenin, "Detection performance of radar compressive sensing in noisy environments," *International SPIE Conference on Defense, Security, and Sensing*, May 2013.

[3] J.L.V.M. Stanislaus and T. Mohsenin, "Low-complexity fpga implementation of compressive sensing reconstruction," in *Computing, Networking and Communications (ICNC), 2013 International Conference on*, Jan 2013, pp. 671–675.

[4] J.L.V.M. Stanislaus and T. Mohsenin, "High performance compressive sensing reconstruction hardware with qrd process," in *Circuits and Systems (ISCAS), 2012 IEEE International Symposium on*, 2012, pp. 29–32.

[5] Amey M. Kulkarni, Houman Homayoun, and Tinoosh Mohsenin, "A parallel and reconfigurable architecture for efficient omp compressive sensing reconstruction," in *Proceedings of the 24th Edition of the Great Lakes Symposium on VLSI*, New York, NY, USA, 2014, GLSVLSI '14, pp. 299–304, ACM.

[6] A. Septimus and R. Steinberg, "Compressive sampling hardware reconstruction," in *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*, May 2010, pp. 3316–3319.

[7] Guoxian Huang and Lei Wang, "Soft-thresholding orthogonal matching pursuit for efficient signal reconstruction," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, May 2013, pp. 2543–2547.

[8] Alexandre Borghi, Jrme Darbon, Sylvain Peyronnet, TonyF. Chan, and Stanley Osher, "A simple compressive sensing algorithm for parallel many-core architectures," *Journal of Signal Processing Systems*, vol. 71, no. 1, pp. 1–20, 2013.

[9] Lin Bai, P. Maechler, M. Muehlberghuber, and H. Kaeslin, "High-speed compressed sensing reconstruction on fpga using omp and amp," in *Electronics, Circuits and Systems (ICECS), 2012 19th IEEE International Conference on*, Dec 2012, pp. 53–56.

[10] M Andrecut, "Fast GPU implementation of sparse signal recovery from random projections," 2008.

[11] 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.

[12] J. Bisasky, H. Homayoun, F. Yazdani, and T. Mohsenin, "A 64-core platform for biomedical signal processing," in *Quality Electronic Design (ISQED), 2013 14th International Symposium on*, March 2013, pp. 368–372.

[13] J. Bisasky, D. Chandler, and T. Mohsenin, "A many-core platform implemented for multi-channel seizure detection," in *Circuits and Systems (ISCAS), 2012 IEEE International Symposium on*, May 2012, pp. 564–567.

[14] Deanna Needell and Joel A. Tropp, "Cosamp: Iterative signal recovery from incomplete and inaccurate samples," *Commun. ACM*, vol. 53, no. 12, pp. 93–100, Dec. 2010.

[15] Rafael C. Gonzalez and Richard E. Woods, *Digital Image Processing (3rd Edition)*, Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2006.

[16] Amey Kulkarni and Tinoosh Mohsenin, "Parallel heterogeneous architectures for efficient omp compressive sensing reconstruction," *International SPIE Conference on Defense, Security, and Sensing*, May 2014.

[17] Mohammad Khavari Tavana, Amey Kulkarni, Abbas Rahimi, Tinoosh Mohsenin, and Houman Homayoun, "Energy-efficient mapping of biomedical applications on domain-specific accelerator under process variation," in *Proceedings of the 2014 International Symposium on Low Power Electronics and Design*, New York, NY, USA, 2014, ISLPED '14, pp. 275–278, ACM.