High Performance Architectures for OMP Compressive Sensing Reconstruction Algorithm

Amey Kulkarni and Tinoosh Mohsenin
Department of Computer Science & Electrical Engineering
University of Maryland, Baltimore County

***Abstract*—Compressive Sensing (CS) is a novel scheme, in which a signal that is sparse in a known transform domain can be reconstructed using fewer samples. The signal reconstruction techniques are computationally intensive and power consuming, which makes them impractical for embedded applications . The paper presents novel architectures for Orthogonal Matching Pursuit algorithm, one of the popular CS reconstruction algorithms. We show the implementation results of proposed architectures on FPGA, ASIC and on a custom many-core platform. For FPGA and ASIC implementation, a novel thresholding method is used to reduce the processing time for the optimization problem by at least 25%. Whereas, for the custom many-core platform, efficient parallelization techniques are applied, to reconstruct signals with variant signal lengths of N and sparsity of m. The algorithm is divided into three kernels. Each kernel is parallelized to reduce execution time, whereas efficient reuse of the matrix operators allows us to reduce area. Matrix operations are efficiently paralellized by taking advantage of blocked algorithms. For demonstration purpose, all architectures reconstruct a 256-length signal with maximum sparsity of 8 using 64 measurements. Implementation on Xilinx Virtex-5 FPGA, requires 27.14 μs to reconstruct the signal using OMP i.e without thresholding method. Whereas, with thresholding method it requires 18 μs. ASIC implementation reconstruct the signal in 13 μs. However, our custom many-core operating at 1.18 GHz, takes 18.28 μs to complete. Our results show that compared to previous published work of the same algorithm, and matrix size, proposed architectures for OMP are approximately 1.3 times to 3 orders of magnitude times faster.**

I. Introduction

Compressive Sensing (CS) is a novel sampling scheme that can have significant impact on different applications. For Magnetic Resonance Imaging (MRI), CS reconstructs the image from sparse measurements and this helps reduce the scan time since it is proportional to the number of samples acquired. Compressive Sensing has been used for Radar Imaging applications such as Synthetic Aperture Radar (SAR), through-the-wall Radar (TWR).

Radar Signal Processing encompasses a wide range of applications in processing techniques, sensing objectives,propagation media etc. Radar Imaging is nowadays mainly used in military and civilian applications. These applications need to be of high resolution. Therefore, these applications require wider bandwidth and hence necessitates large data to transmit, receive and process.

Traditionally, radar receivers consist of highly complex and expensive high rate A/D converters followed by analog /digital matched filters. Similarly, signal has to be sampled at Nyquest rate to reconstruct the signal without aliasing. Whereas, obtaining the signals at Nyquest rate is wasteful if signal is sparse in some domain. Another method is under sampling or down sampling which leads to loss of information. In CS, instead of obtaining few samples, fewer measurements are used to reconstruct a signal under certain conditions. Reducing the number of measurements can reduce the time and cost of signal acquisition.

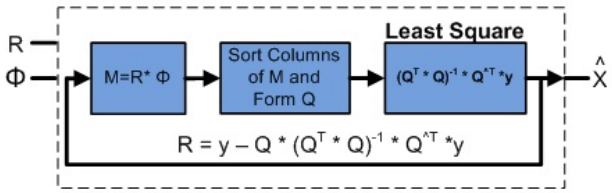

Fig. 1. Basic Block Diagram for OMP Reconstruction Algorithm

Though the CS has several advantages, reconstruction of CS is very complex and computational intensive. In Radar applications, both Inverse Synthetic Aperture Radar (ISAR) imaging and Through-the-Wall Radar (TWR) imaging achieve great benefit from CS. ISAR imaging is used for maritime targets, whereas TWR imaging is used to get vision into obscured areas. Therefore, both the systems need to reconstruct the signals in real time to be effective. In this paper, we present both algorithmic modification and novel hardware implementation on FPGA, ASIC and on a custom many-core platform for Orthogonal Matching Pursuit (OMP), one of the popular CS reconstruction algorithms.

II. Proposed Work

The OMP algorithm used for this project is adopted from [1]. OMP reconstruction takes two inputs, the measurement matrix ϕ and the sampled data y, with a sparsity m. The high level block diagram for OMP reconstruction is shown in Figure 1.

The goal is to minimize the difference between the estimated signal (from reconstruction) and y values. This difference is called residue, R, which is used as the input for the next iteration. R is initialized with y values for the first iteration. The algorithm starts by finding m columns of ϕ, which is mostly correlated to the y values. This is achieved by first multiplying y vector and ϕ matrix and then finding m columns that have the largest numbers. Next step is to form the Q matrix whose columns correspond to m largest values of multiplication.Then using Q which is much smaller size than original ϕ matrix the signal is reconstructed using least square problem. The estimated result is multiplied by y value and subtracted from y to form R which is used for the next iteration. This is repeated m times or until R becomes near zero.

A. Proposed Architecture for FPGA and ASIC

Dot product calculation and fixed point division operations increase datapath latency. Therefore, architecture proposed here has highly optimized datapath logic which nearly doubles the performance of the hardware. Proposed new thresholding method eliminates eliminates columns of ϕ whose absolute

TABLE I
FPGA IMPLEMENTATION SUMMARY FOR OMP RECONSTRUCTION HARDWARE ON VIRTEX-5, XC5VL110T

Design	Signal Length N	Sparsity m	Cycles	Time (μS)
Without Threshold (α)	256	8	2100	27.14
Without Threshold (α)	128	5	820	10
With Threshold (α) = 0.25	256	8	1280	18
With Threshold (α) = 0.25	128	5	444	7.13

TABLE II
ASIC IMPLEMENTATION SUMMARY FOR OMP RECONSTRUCTION HARDWARE WITH 85×256 AND SPARSITY m = 8, 65NM CMOS

	Design Specifications
Technology	65 nm, 1 V
Logic utilization	90%
Total area (mm^2)	0.69
Performance (MHz)	165
Reconstruction Time (μs)	13.7*

dot product values ($|R*\phi|$) are less than a predefined threshold value (α). Thus in each iteration the ϕ matrix becomes smaller than previous iteration and therefore it reduces the computation latency [2]. The simulation results show that, with $\alpha = 0.25$, the error difference of 0.0056 from original implementation. The architecture proposed here has a highly optimized datapath logic which nearly doubles the performance of the hardware.

B. Proposed Architecture for the Custom Many-core

The 256-core custom manycore platform is designed by EEHPC lab and is targeted for DSP applications. The algorithm can be implemented in three important kernels as shown in Figure 1. First kernel, which is matrix-vector multiplication can be parallelized and reconfigured using maximum multipliers [3]. Whereas, the second kernel, sort algorithm can be parallelized using binary trees [4], or by decomposing the computation into independent tasks that perform minimal global communication [5]. Third kernel is Least Square Minimization which consists of mainly matrix inversion. Matrix inversion is the most complex and it needs large memory. Hence, we focus on its optimization. Blocked algorithms are used for implementing matrix inversion while taking the advantage of parallelism. For matrix inversion, block LU decomposition algorithm is used,which decomposes matrix into lower matrix L and upper matrix U. Inverse of matrix is achieved with efficient use of parallel backward and forward substitution method, in combination with LU algorithm.

III. Implementation Results

A. FPGA and ASIC Implementation Results

Results for basic and improved OMP reconstruction algorithm on FPGA and ASIC are shown in Table I & II respectively. As expected, the cycle counts for improved OMP decreases with increasing α and is improved by 64% to 84% with $\alpha = 0.25$. The post layout view of ASIC implementation in 65nm is shown in Figure 2.

B. Many-Core Implementation Results

The results for a measurement matrix of size [85×256] are described in Table III. One iteration of OMP algorithm takes 21576 cycles on a custom many-core platform and the execution time for this is approximately 18μs.

TABLE III
CYCLE COUNT FOR OMP ALGORITHM WITH 85×256 AND SPARSITY m = 8 ON CUSTOM MANY-CORE PLATFORM, OPERATING AT 1.18GHz

Program	Cycles	Time (μS
$R*\phi$	340	0.28
Sort	1870	1.58
Least Square Minimization	19366	16.41
Total	21576	18.28

TABLE IV
COMPARISON OF OMP RECONSTRUCTION TIME FOR THE PREVIOUS IMPLEMENTATIONS

Device	Signal Length	Sparsity	Frequency	Time
FPGA (*Virtex-5*) [1]	128	5	39 Mhz	24 μs
Intel Core i7, 920 [1]	128	-	2.6 GHz	68 ms
CUBLAS (GPU) [1]	128	-	-	37.5 ms
This work (Many-Core)	256	8	1.18 GHz	**18.28 μs**
This Work FPGA (*Virtex-5*) [2]	256	8	165 MHz	**18 μs**
This Work ASIC [6]	256	8	165 MHz	**13.7 μs**

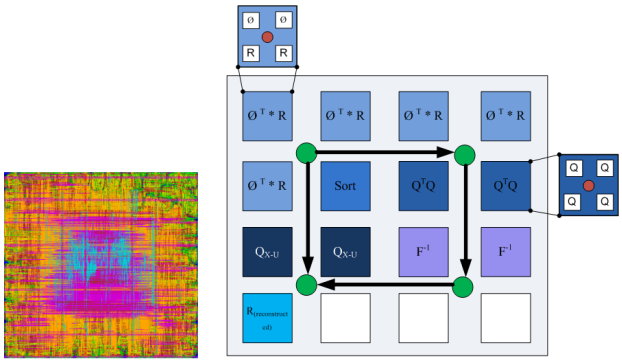

Fig. 2. (a) Post layout view of proposed OMP Reconstruction Algorithm in 65nm CMOS (b) Mapping of OMP algorithm on our custom many-core platform. For the many- core each 4-processor cluster communicate through one shared-router (red circle) and clusters communicate through green routers [7], [8].

C. Comparison of OMP Implementation on Low-Power Many-core with Others

Table IV summarizes comparison of proposed architectures with the previous implementations.

References

[1] A. Septimus and R. Steinberg, "Compressive sampling hardware reconstruction," in *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*, 30 2010-june 2 2010, pp. 3316 –3319.

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

[3] L. Zhuo and V. Prasanna, "High-performance designs for linear algebra operations on reconfigurable hardware," *Computers, IEEE Transactions on*, vol. 57, no. 8, pp. 1057–1071, 2008.

[4] D. Mihhailov, V. Sklyarov, I. Skliarova, and A. Sudnitson, "Parallel fpga-based implementation of recursive sorting algorithms," in *Reconfigurable Computing and FPGAs (ReConFig), 2010 International Conference on*, 2010, pp. 121–126.

[5] N. Satish, M. Harris, and M. Garland, "Designing efficient sorting algorithms for manycore gpus," in *Parallel Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on*, 2009, pp. 1–10.

[6] J. Stanislaus and T. Mohsenin, "High performance compressive sensing reconstruction hardware with qrd process," *IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2012.

[7] J. Bisasky, J. Chander, and T. Mohsenin, "A many-core platform implemented for multi-channel seizure detection," *IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2012.

[8] J. Bisasky, T. Mohsenin, and H. Homayoun, "A many-core platform for biomedical signal and image processing," *International Symposium on Quality Electronic Design (ISQED)*, March 2013.