A Parallel and Reconfigurable Architecture for Efficient OMP Compressive Sensing Reconstruction

Amey Kulkarni
Department of Computer Science & Electrical Engineering
University of Maryland, Baltimore County
Baltimore,USA
ameyk1@umbc.edu

Houman Homayoun
Department of Electrical & Computer Engineering
George Mason University
Fairfax,USA
hhomayou@gmu.edu

Tinoosh Mohsenin
Department of Computer Science & Electrical Engineering
University of Maryland, Baltimore County
Baltimore,USA
tinoosh@umbc.edu

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. However, the signal reconstruction techniques are computationally intensive and power consuming, which make them impractical for embedded applications. This work presents a parallel and reconfigurable architecture for Orthogonal Matching Pursuit (OMP) algorithm, one of the most popular CS reconstruction algorithms. In this paper, we are proposing the first reconfigurable OMP CS reconstruction architecture which can take different image sizes with sparsity up to 32. The aim is to minimize the hardware complexity, area and power consumption, and improve the reconstruction latency while meeting the reconstruction accuracy. First, the accuracy of reconstructed images is analyzed for different sparsity values and fixed point word length reduction. Next, efficient parallelization techniques are applied to reconstruct signals with variant signal lengths of N. The OMP algorithm is mainly divided into three kernels, where each kernel is parallelized to reduce execution time, and efficient reuse of the matrix operators allows us to reduce area. The proposed architecture can reconstruct images of different sizes and measurements and is implemented on a Xilinx Virtex 7 FPGA. The results indicate that, for a 128×128 image reconstruction, the proposed reconfigurable architecture is 2.67× to 1.8× faster than the previous non-reconfigurable work which is less complex and uses much smaller sparsity.

Categories and Subject Descriptors

B.2 [**ARITHMETIC AND LOGIC STRUCTURES**]: Design StylesParallel,Pipeline; B.2.4 [**High-Speed Arithmetic**]: Algorithms

GLSVLSI'14, May 21–23, 2014, Houston, Texas, USA.

http://dx.doi.org/10.1145/2591513.2591598 .

Keywords

OMP;Compressive Sensing; FPGA; High Performance and Reconfigurable Architecture

1. INTRODUCTION

In recent years, Compressive Sensing (CS) has emerged as a novel technique which enables reconstruction of sparse signals sampled at sub-Nyquist rates. Reducing the number of measurements can reduce the time and cost of signal acquisition. CS reduces the amount of data collected during signal acquisition thereby, eliminating redundancy. Reducing the number of measurements can significantly reduce the communication power (e.g. space applications, wearable biomedical devices), scanning time (e.g. MRI) and cost of signal acquisition.

CS is used for radar imaging applications due to its fast and efficient signal processing ability. Radar Signal Processing encompasses a wide range of applications in processing techniques, sensing objectives, propagation media etc. Recently, it is mainly being used in military and civilian applications. The signal in these applications needs to be of high resolution. Therefore, it requires wider bandwidth and hence necessitates large amount of data for transmission, reception and processing. Similarly, applications such as Inverse Synthetic Aperture Radar (ISAR) imaging used for maritime targets, and through-the-wall Radar (TWR) imaging used to get vision into obscured areas. Therefore, both the systems need to reconstruct the signal in real-time to be effective.

On the other hand, in Medical Resonance Imaging (MRI), CS reconstructs the image from sparse measurements and reduces scan time, which is proportional to the number of samples acquired. Long term functional MRI requires continuous high speed imaging. The continuous acquisition of images results in high volume of data. Therefore, instead of considering the whole image, most of the devices allow pre-determined rectangular region of interest (ROI) to be sampled. In medical imaging, ROI is the area of an image which is very important for diagnosis. The ROI is variable and is based on the stage of disease. For example, Benign lesion has less ROI as compared to malignant lesion. Hence, the size of image to be sampled and reconstructed is dependent on ROI. Therefore, CS reconstruction is in intense need of reconfigurable architectures.

Though CS has several advantages, reconstruction of CS is very complex and computationally intensive. Recently, there have been several reconstruction algorithms proposed which show trade-off between complexity and accuracy. Two such algorithms are ℓ_1–minimization and Orthogonal Matching Pursuit (OMP) [11]. ℓ_1–minimization algorithm is better in terms of accuracy, but its implementation is very complex and time-consuming. OMP is a greedy algorithm of less complexity that finds closely correlated values in each iteration. The complexity of the design increases with data length and sparsity number. OMP contains iterative interdependent modules which repeat iteratively up to sparsity count makes parallel implementation of the algorithm challenging. OMP has matrix multiplication, sort and inversion, which are known to be operator consuming operations. Therefore, the signals with large matrix lengths will require more resources and hence larger chip area. This motivates us to consider a semi-parallel architecture wherein operators are reused, thereby reducing chip area.

To address these challenges, this paper present a low complexity, parallel and reconfigurable architecture to accept variable matrix image lengths and measurements as input. In the OMP algorithm, at each iteration, matrix sizes vary. Therefore, the main challenging blocks in making the architecture reconfigurable are dot product calculations and least square block. Since the iteration count is dependent on sparsity, it is fixed to 32 based on our prior experiments and analysis.

The structure of this paper is as follows: in Section 2, paper overviews trends and related work. Section 3, goes over the OMP algorithm. Section 4 proposes the architecture for reconfigurable and parallel OMP reconstruction algorithm. Finally, in Section 5 paper discusses the FPGA implementation results and analysis.

2. BACKGROUND

The basic theory behind compressive sensing 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 x be a m-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.

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

Orthogonal Matching Pursuit (OMP) is a greedy algorithm, which performs the signal recovery from random measurements [20]. For the first few years, research on the CS OMP reconstruction algorithm was focused on reducing latency of operations. The focus has shifted over the past few year to reducing energy consumption as CS is implemented for wireless and battery operated devices [9], [10]. Most of the OMP designs are implemented on FPGA [12] [17], [18],[5], [1], some implement the designs on ASIC [12] [17], [21], [9] and others implement the architecture on ASIP [6]. Similarly, most of the papers compare their results with previous architectures and show improvements in terms of speed (latency of operations) [18]. Some prior work also shows the improvement in speed on software [1]. To the best of our knowledge, this is the first implementation which targets reconfigurability for OMP algorithm.

The first basic implementation of OMP is presented [17], which shows interdependence between modules. It implements OMP with sparsity up to $m = 5$ and input vector of 128 size on FPGA. OMP with QR decomposition is proposed in [18], to speed up the algorithm. The paper implements the design on CMOS 65nm and FPGA and is 2.4 times faster than [17]. Black et.al. [5] segregates OMP algorithm in three kernels and each kernel is parallelized to reduce latency of operations. The paper compares the speedup with a software implementation, and performs 38 times faster. L.Bai et.al. [1] propose the design and implementation of OMP and AMP algorithms on FPGA and compare hardware with software implementations. In contrast to other VLSI implementations, this implementation can deal with less sparse signals, which enables fast image reconstruction. The architecture works at 100MHz with a reconstruction time of 0.63 msec.

Tsai et.al. [21] present a versatile signal reconstruction platform. It is implemented in three different blocks. Pseudo Random Number Generator generates random numbers on the fly to reduce memory complexity. Matrix factorization engine contains Cholesky based linear solver and the multiprocessing core performs the other operations. The design is implemented in TSMC 40nm CMOS process and runs at 250 MHz.

The trend to reduce latency of operations and energy consumption begins with [9]. This work proposes OMP with Matrix Inversion Bypass to reduce computational complexity. The paper targets battery or renewable energy powered cyber-physical systems. It compares with OMP Batch algorithm, which is less complex and performs the operation in one iteration. The paper implements Moore-Penrose pseudo inverse referred to as the updated pseudo inverse solution, that utilizes the matrix $G - 1$ available from the previous operation to obtain current matrix $G - 1$. Specifically, this method calculates $G-1$ using the Schur-Banachiewicz block wise inversion (due to low complexity) . In other words, the computation of $G - 1$ for current inversion can be bypassed. As $G-1$ is only needed in the (k-1)th iteration, it can be computed in parallel with the computation of X to improve the speed of signal reconstruction. The design is implemented on 65nm CMOS process with the clock frequency of 500MHz.

Soft thresholding OMP technique to reduce the energy for reconstruction of the signal is implemented in [10]. This paper targets CS to exploit renewable energy sources in autonomous and distributed wireless sensor networks. The threshold of employing the efficient reconstruction is made dynamically adjustable according to the performance requirements and energy levels. The paper takes motivation from the fact that last iteration usually recovers less significant elements of the signal. Hardware is implemented on 65nm CMOS process with clock frequency of 500MHz. 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 msec to recover a signal and consumes 0.0205 mJ energy.

OMP algorithm with a sub-V_t Application Specific Instruction set Processor (ASIP) for exploiting specific operations of CS is implemented in [6]. The paper mainly targets battery operated devices i.e., sensing environment systems and wireless body sensor networks (WBSNs), where portable and autonomous devices are expected to operate for an ex-

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 $\Phi_t = [\Phi_{t-1}\ \phi_{\Lambda_t}]$
5: Solve the Least Squares Problem
$x_t = \min_x \| y - \Phi_y\ x\|^2$
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.

tended period of time with limited energy resources. Hence, an ultra low power (ULP) CS implementation is crucial for these energy limited autonomous systems. The processor has sleep mode which allows external clock gating of the entire core. The processor has sub-V_t latch-based memories. The paper shows power and performance trade-off. The simulation result shows that CS processor operates at 0.37 V for required clock frequency of 100KHz with total power 288nW and critical path of 5.2 nsec.

3. ALGORITHM

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, correct set of columns are determined [20] [11].

The variables used in the algorithm are defined below:

- N× N = Images Size (e.g 128 × 128...768 × 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)

4. PROPOSED WORK

Based on the algorithm description, OMP is partitioned into three main kernels: *dot product, sort and least square* (which involves *matrix inversion*). These blocks are shown in Fig. 1. As shown in the figure, these three kernels are interdependent, therefore OMP cannot be fully parallellized. This paper implements a semi-parallel architecture for each of the kernels to fit different data sizes of reconstruction.

4.1 Sparsity Analysis

Computational / hardware complexity is a major factor in parallel implementation of the design on FPGA for different sizes of a matrix. From Fig. 1,it is observed that Least Square is the most complex kernel of the three. In OMP algorithm, each column is repeated k (Sparsity) times. Therefore, $\phi^T \times \phi$ increments in each iteration till $k \times k$. Therefore,

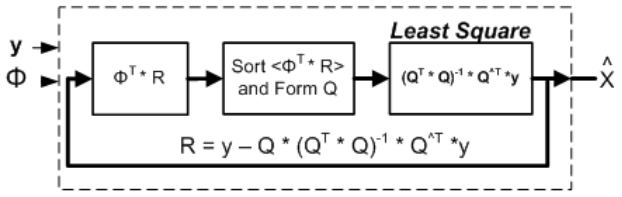

Figure 1: Basic Block Diagram for OMP Reconstruction Algorithm

Table 1: PSNR and Sparsity Analysis numbers for variety of Image Sizes with OMP Reconstruction Algorithm

Image Sizes	Sparsity k	PSNR (dB)
256×256	8	27.22
256×256	**32**	**34.70**
256×256	48	34.89
384×384	8	21.24
384×384	**32**	**22.88**
384×384	48	22.91
512×512	8	21.19
512×512	**32**	**25.65**
512×512	48	25.71
768×768	**32**	**22.55**
768×768	90	22.95

complexity of Least Square algorithm is dependent on the size of sparsity. Hence, the primary focus is to reduce the hardware complexity by setting the sparsity to a constant value. This reduces the size of the matrix during inversion.

In this paper, the sparsity of the matrix is set based on experimenting on the different sizes of images while observing satisfactory range of PSNR. The random measurement matrix (One of the input to the OMP) changes each time while running the experiment and hence the result may vary from 0.09% to 0.158%. Therefore, average Peak Signal-to-Noise Ratio (PSNR) is calculated by repeating the experiment 100 times. Setting sparsity size to 32 helps to reduce operations of the complex Least Square algorithm. Table 1 shows different sizes of images (N × N) with different sparsity and their PSNR results. It clearly indicates that variation in image sizes for a constant sparsity has minimal effect on PSNR. However, from a hardware stand point, it is advantageous since it reduces memory transfers, area and speed.

4.2 Fixed Point Optimization

Floating point arithmetic is complex and requires more area. At the same time, OMP algorithm requires huge number of multiplications and additions. Hence floating point operations increase hardware complexity. This motivates the selection of fixed point arithmetic. As shown in fig. 3, fixed point arithmetic increases the hardware complexity when accuracy reaches threshold. In this paper, we experimented output accuracy and hardware complexity with different binary points. The quality of reconstructed image is mainly dependent on accuracy of the fixed point output. Hence, binary points are chosen close to their corresponding floating point numbers (100 % accuracy) at the output. Ac-

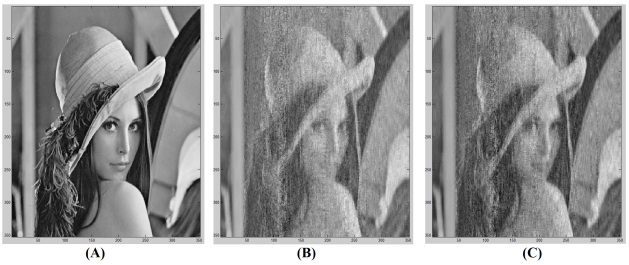

Figure 2: (A) 256 ×256 Original Image, (B) and (C) reconstructed images with sparsity values of 8 and 32.

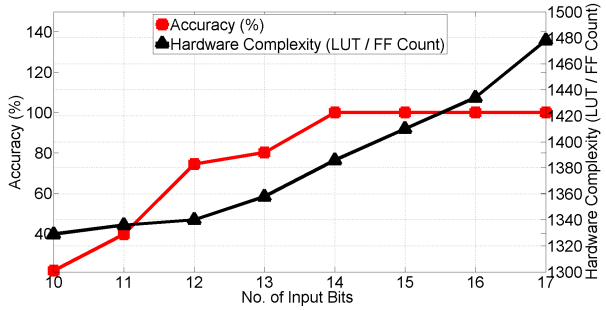

Figure 3: Hardware complexity (LUT/FF count in *Virtex*-7 FPGA) and accuracy (fixed point vs floating point) of OMP CS Reconstruction algorithm when the number of input bits increases Accuracy with 0.0005 margin of error

curacy of the output is verified at each hardware step along the data path post truncation/saturation. Similarly, by incrementing bits at the input, the data path word width is changed. This affects area of an architecture and the amount of switching activity on wires and logic gates, consequently affecting the power dissipation.

4.3 Reconfigurable DOT Product

At each iteration, OMP computes dot product of ϕ^T, which is a N × M matrix, with a residual matrix R, which is a M×1 vector. This has a computational complexity of O(MN). Tree multiplier is implemented to leverage the parallel and pipeline architecture (Figure 4). It is implemented for each 42×1 size array. For our implementation measurements are 30% of image sizes.

Tree multiplier requires N multipliers and $N-1$ adders where, N is the number of columns. Thus total number of operations come to be $2N-1$. As discussed earlier, this architecture considers N columns, which are multiple of 42. Therefore, at every cycle dot product of 42×1 and 1×42 is available. For the image size of 128×128, column size of ϕ^T is 42. Therefore, the dot product of 128×42 and 42×1 is computed in 128 clock cycles.

4.4 Sort Algorithm

The operation of the second kernel is to locate the maximum of $|<\phi R>|$ (N×1 vector). This has a computational complexity of O(N). Sort algorithm is implemented by using

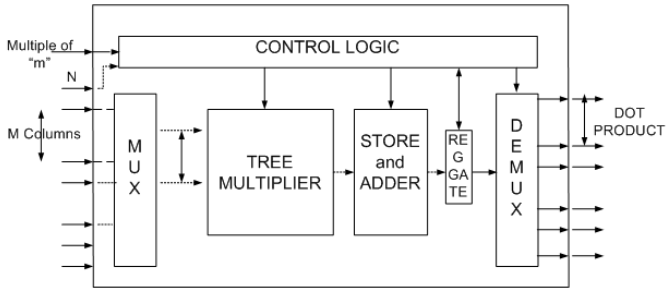

Figure 4: High level block diagram of Reconfigurable Dot Product

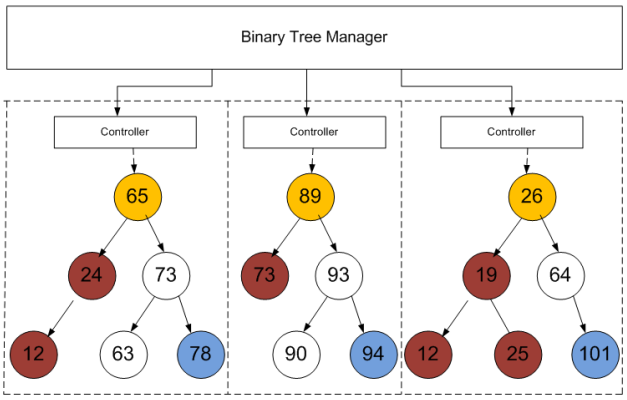

Figure 5: Example of Parallel Implementation of Binary Tree

a binary tree structure [15]. $N/2^S$ trees are implemented, where S is variable and dependent on size of an image. Concurrent sorting is applied to $N/2^S$ to efficiently use parallelism as shown in Fig. 5. In this architecture, since the algorithm needs only the highest number, we could reduce the memory usage as compared to [15], by pruning the left sub-tree. Each concurrent binary tree gives highest number, thereby generating $N/2^S$ highest numbers which are fed to another binary tree. Reconfigurability is achieved by changing the number of trees in parallel, dependent on the size of an image. Since the architecture is two staged, for every two cycles, a maximum of 128 elements is obtained.

4.5 Least Square Method

This is the most important kernel of the algorithm. As mentioned, $(\phi^T\ \phi)^{-1}$ has the highest hardware implementation complexity. At each iteration t, ϕ has t columns of size M. Hence, the new matrix, ϕ, is of size t×M. Computing $(\phi^T\ \phi)$ gives a t×t resulting matrix. Least square $(x{=}(\phi^T\ \phi)^{-1}\ \phi^T\ y)$ has three sub-blocks. Matrix transpose is achieved by calling matrix index in transpose order, which reduces the hardware complexity and utilizes minimal resources. Matrix multiplication is based on tree architecture mentioned previously in section 4.3. Whereas, matrix inversion is obtained by LU decomposition leveraging the symmetric matrix. As shown in Fig. 6, blocked algorithm for LU decomposition is used for efficient parallel implemen-

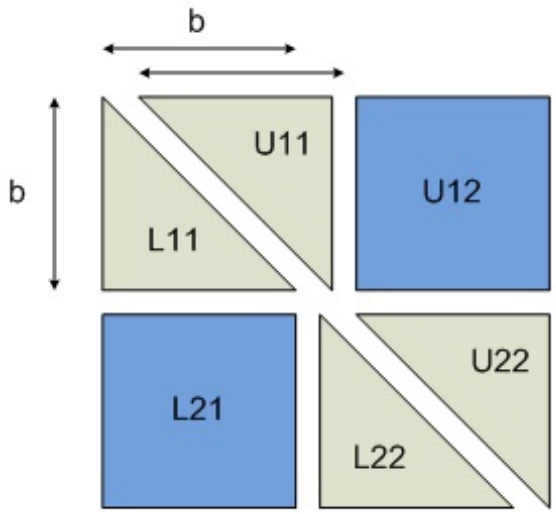

Figure 6: Blocked LU Algorithm for Parallel Implementation of Matrix Inversion

Table 2: *Virtex-7* post layout implementation results of DOT product for different image sizes

$< \phi R >$	Parallel Implementation		Serial Implementation	
	Time (ns)	Dynamic Power(W)	Time (ns)	Dynamic Power(W)
42×128	3.67	0.032	0.76	0.012
84×256	7.34	0.032	1.52	0.012
168×512	14.72	0.034	3.04	0.013

tation [14]. Finally, the resources are reused to reduce the area of the architecture.

5. IMPLEMENTATION RESULTS

The proposed reconfigurable architecture is implemented on a *Xilinx XC7VX485T Virtex-7* FPGA, which uses the sparsity of $k = 32$ for different size of images.

Parallel dot product implementation, gives one results each cycle. Therefore, after n cycles, the dot product for the whole image will be obtained, where n is the row size of an image. Table 2, shows the implementation results for serial and parallel architecture for different sizes of matrix.

As discussed in section 4.4, sort algorithm is implemented in parallel. Each binary tree sorts 32 elements. Therefore, for 128 × 1 array , the architecture has four concurrent binary trees in first stage. Second stage has a simple comparator tree to segregate the last four elements from each binary sub-tree.

For the sparsity of $k = 32$ (i.e 32 iterations), OMP takes 6208 cycles to reconstruct each column of image and hence $8.97\mu S$ for a 128 × 128 image, 9.32 μS for a 256 × 256 image and 10.12 μS for a 512 × 512 image size. The 256 × 256 reconstructed image has the PSNR of 35 dB , and 512 × 512 image achieves the PSNR of 26 dB in both fixedpoint hardware and floating point software. Post place and route timing results show that the proposed architecture is 62.62 % (2.67 ×) faster than the previous implementation [17], which takes 24 μS to reconstruct a 128 × 128 image, whereas 43.93 % (1.8 ×) faster as compared to [5]. Both previos work use a fixed image with much smaller sparsity (i.e. k=5).

Table 3: *Virtex-7* post layout implementation results of Sort algorithm for different image sizes

$\lambda_{j=1...n}$	Parallel Implementation		Serial Implementation	
	Time (ns)	Dynamic Power(W)	Time (ns)	Dynamic Power(W)
128	3.85	0.041	9.47	0.010
256	8.7	0.083	19.25	0.010
512	10.48	0.16	23.1	0.010

6. CONCLUSION

The paper presents the first reconfigurable architecture for the OMP compressive sensing reconstruction algorithm. The hardware can take different image sizes with sparsity up to 32. The accuracy of reconstructed image is analyzed for different sparsity values and fixedpoint wordlength reduction. The results indicate that OMP peforms well with fixing the sparsity to be 32 for different image sizes of 128×128 to 768× 768. This reduces the hardware complexity in terms of the number of iterations as well as matrix inversion size. Datapath wordlength is optimized such that the hardware has minimum area with a highly accuracy reconstructed image. Different parallelization techniques are used for implementing the main three kernels of OMP algorithm which are dot product, sort and matrix inversion. The reconfigurable architecture is implemented on a Xilinx Virtex 7 FPGA. The results indicate that, for a 128×128 image reconstruction, the proposed architecture is 2.67× to 1.8× faster than the previous non-reconfigurable works which use much smaller sparsity.

7. REFERENCES

[1] L. 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*, pages 53–56, 2012.

[2] A. BanaiyanMofrad, H. Homayoun, and N. Dutt. Fft-cache: A flexible fault-tolerant cache architecture for ultra low voltage operation. In *Compilers, Architectures and Synthesis for Embedded Systems (CASES), 2011 Proceedings of the 14th International Conference on*, pages 95–104, Oct 2011.

[3] 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*, pages 564–567, May 2012.

[4] 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*, pages 368–372, March 2013.

[5] P. Blache, H. Rabah, and A. Amira. High level prototyping and fpga implementation of the

orthogonal matching pursuit algorithm. In *Information Science, Signal Processing and their Applications (ISSPA), 2012 11th International Conference on*, pages 1336–1340, 2012.

[6] J. Constantin, A. Dogan, O. Andersson, P. Meinerzhagen, J. Rodrigues, D. Atienza, and A. Burg. Tamarisc-cs: An ultra-low-power application-specific processor for compressed sensing. In *VLSI and System-on-Chip (VLSI-SoC), 2012 IEEE/IFIP 20th International Conference on*, pages 159–164, 2012.

[7] H. Homayoun, V. Kontorinis, A. Shayan, T.-W. Lin, and D. Tullsen. Dynamically heterogeneous cores through 3d resource pooling. In *High Performance Computer Architecture (HPCA), 2012 IEEE 18th International Symposium on*, pages 1–12, Feb 2012.

[8] H. Homayoun, S. Pasricha, M. Makhzan, and A. Veidenbaum. Improving performance and reducing energy-delay with adaptive resource resizing for out-of-order embedded processors. *SIGPLAN Not.*, 43(7):71–78, June 2008.

[9] G. Huang and L. Wang. High-speed signal reconstruction with orthogonal matching pursuit via matrix inversion bypass. In *Signal Processing Systems (SiPS), 2012 IEEE Workshop on*, pages 191–196, 2012.

[10] G. Huang and L. Wang. Soft-thresholding orthogonal matching pursuit for efficient signal reconstruction. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 2543–2547, 2013.

[11] A. Korde, D. Bradley, and T. Mohsenin. Detection performance of radar compressive sensing in noisy environments. *International SPIE Conference on Defense, Security, and Sensing*, May 2013.

[12] A. Kulkarni and T. Mohsenin. Parallel heterogeneous architectures for efficient omp compressive sensing reconstruction. *International SPIE Conference on Defense, Security, and Sensing*, May 2014.

[13] A. Kulkarni, J. Stanislaus, and T. Mohsenin. High performance architectures for omp compressive sensing reconstruction algorithm. *39th Annual GOMACTech Conference*, April 2014.

[14] M. Kumar Jaiswal and N. Chandrachoodan. Fpga-based high-performance and scalable block lu decomposition architecture. *Computers, IEEE Transactions on*, 61(1):60–72, 2012.

[15] 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*, pages 121–126, 2010.

[16] A. Sasan, H. Homayoun, A. Eltawil, and F. Kurdahi. Inquisitive defect cache: A means of combating manufacturing induced process variation. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 19(9):1597–1609, Sept 2011.

[17] A. Septimus and R. Steinberg. Compressive sampling hardware reconstruction. In *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*, pages 3316–3319, 2010.

[18] J. Stanislaus and T. Mohsenin. High performance compressive sensing reconstruction hardware with qrd process. In *Circuits and Systems (ISCAS), 2012 IEEE International Symposium on*, pages 29–32, 2012.

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

[20] J. Tropp and A. Gilbert. Signal recovery from random measurements via orthogonal matching pursuit. *Information Theory, IEEE Transactions on*, 53(12):4655–4666, 2007.

[21] Y.-M. Tsai, T.-J. Yang, and L.-G. Chen. A 401gflops/w 16-cores signal reconstruction platform with a 4g entries/s matrix generation engine for compressed sensing and sparse representation. In *VLSI Circuits (VLSIC), 2013 Symposium on*, pages C256–C257, 2013.