High Performance Compressive Sensing Reconstruction Hardware with QRD Process

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

***Abstract*—This paper presents a high performance architecture for the reconstruction of compressive sampled signals using Orthogonal Matching Pursuit (OMP) algorithm. Q-R decomposition (QRD) process is used for the matrix inverse core and a new algorithm for finding fast inverse square root of a fixed point number is also implemented to support the QRD process. The optimized architecture takes 256-length input vector and 64 measurement data, and reconstructs a signal of sparsity 8. The design is implemented in 65 nm CMOS which runs at 165 MHz and occupies 0.69 mm^2, total reconstruction takes 13.7 μs. The implementation on Xilinx FPGA Virtex-5 takes 27.12 μs to reconstruct a 256–length signal of sparsity 8. The same architecture for 128–length signal of sparsity 5 on Virtex–5 is 2.4 times faster than the state–of–the–art implementation.**

I. Introduction

In many signal processing systems, the useful information is far less than the sampled data and all the redundant data are eliminated through compression. Examples include audio signal compression such as MP3 and color image compression such as JPEG-2000 where almost 97% of the data is thrown away [1]. The reason behind such a high data compression is due to the sparsity of the signal, i.e. the signal has small number of non-zero entries. High data compression can be achieved as long as the signal is sparse in any domain [2]. In all these cases, the signal is captured at the Nyquist rate and the data compression occurs later. Compressive sensing (CS) is a technology where sampling and compression occurs together which will enable us to sample at very low rate, sub-Nyquist rate.

In CS, sparse signals and images can be recovered from very few samples (sub-Nyquist rate) compared to the traditional Shannon's celebrated sampling [1]. Magnetic Resonance Imaging (MRI) is one of the applications of CS theory [3]. CS has two main stages - sampling and reconstruction. While sampling is performed on the transmitter, reconstruction is done on the receiver as shown in Fig. 1.

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

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

Now y is processed to reconstruct m values which will be the close estimate for x, denoted as $\hat{x}$. Since the reconstruction is NP-hard, efforts are made to find a close estimate of x. The

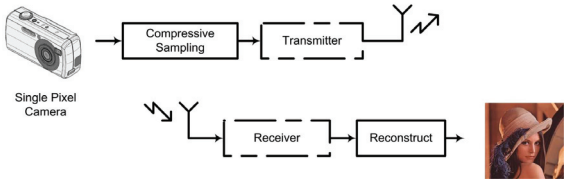

Fig. 1. Basic block diagram for compressive sensing

signal can be sparse in any domain and not necessarily in the sampling domain. Reconstruction requires high computation and the complexity increases with the dimension of the signal. Also, reconstruction is an application of information theory and the complexity increases with the accuracy and the total measurements. There are several reconstruction algorithms proposed in recent years and most of them are computational intensive. Software implementation of these algorithms are time consuming since they often require matrix multiplications for which the processors are poor performers. The above mentioned drawbacks create more interests in hardware implementation for real-time reconstruction for CS signals. The two mostly used reconstruction algorithms are ℓ_1–minimization and Orthogonal Matching Pursuit (OMP). ℓ_1–minimization algorithm is better in terms of accuracy, but its implementation is very complex and time consuming. OMP is less complex [4] and it is a Greedy algorithm that finds the closely correlated values in each iteration. The complexity of the design increases with data length.

In this paper, we propose a high speed architecture for OMP algorithm for 256– length input vector, and Q-R decomposition process (QRD) [5] is used for solving least square problem. The advantages of this approach are that the architecture is optimized for higher set of data length and a new architecture for finding matrix inverse using QRD process speeds up the reconstruction. To perform QRD process, a new algorithm for finding inverse square root has been implemented for fixed point arithmetic, derived from [6]. The design has been implemented on 65 nm CMOS technology whose results are provided at the end and FPGA implementation results are also provided to compare with the previous work.

II. Background

There are only a very few studies available for the implementation of OMP algorithm on the hardware [7][8]. Both studies are based on FPGA implementation, where [7] uses a

978-1-4673-0219-7/12/$31.00 ©2012 IEEE

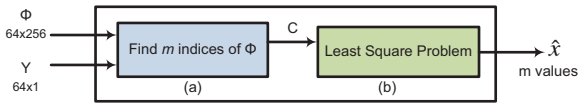

Fig. 2. Basic diagram for OMP reconstruction

.

128–length vector for a sparsity of 5. The same architecture will take a lot more time to compute 256–length vector with a sparsity of 8 due to the path delay in finding the dot product and also due to the division operations performed in the matrix inverse. GPUs have a bottleneck on memory bandwidth [9]. The optimized algorithm explained in [7] will be used as a base for this paper and the new architecture design optimizations are discussed.

III. Proposed Architecture

OMP reconstruction can be divided into two main stages as shown in Fig. 2 and it takes two inputs Φ and y. It should be noted that if the signal is not sparse in the sampling domain, Φ will be the resultant of measurement matrix multiplied by the linear transformation matrix Ψ. For example, an image captured in spatial domain can be brought to wavelet domain by the linear wavelet transformation function Ψ, where the image data becomes sparse. The proposed architecture has been optimized for 256–length vector and a sparsity of 8. Previous work [7] has the critical path delay due to the dot product calculation and also due to the division operation that resulted in the decreased maximum operating frequency. In this proposed architecture, datapath logic is highly optimized which nearly doubles the performance of the hardware. Since the Cholesky decomposition method for higher vector length is quadratically costly due to the matrix size, QRD method is proposed for finding matrix inverse. The two blocks, as shown in Fig. 2, operate at different clock rates that again improves the overall performance.

A. Algorithm

Consider a signal x of sparsity m is sampled using a random matrix Φ and y is the sampled data. We need to find m columns of Φ which contributed to y. To start with, residual R is initialized with y. A column of Φ is chosen for each iteration in such a way that the column has the best correlation with R. This correlation is subtracted from the residual R for next iteration. After finding m columns of Φ, the original signal $\hat{x}$ is recovered by solving an over-determined least square equation. The procedure is given below:

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

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

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

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

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

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

d. Calculate the new residual according to

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

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

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

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

B. Solving Optimization Problem

In this paper, the hardware has been implemented for $N = 256$, measurements $k = 64$ and a sparsity of $m = 8$. Each data uses 24-bit 10.14Q (10 integer bits and 14 fractional bits) fixed point format. It was observed that a larger number of fractional bits do not actually influence the result and 10.14Q format computation is comparable to the floating point simulation. To perform the dot product, 64 multipliers are operated in parallel and the resultants are added together. In this paper, multiply and add is divided into 3 pipeline stages that will decrease the delay of this block. Multipliation takes place in the first stage of pipeline. In the second stage, 8 additions are performed in parallel each adding 8 values. These eight results are added to produce the final dot product in the third stage. It is fully pipelined so that the data is pushed into the module on each clock cycle.

Finding the maximum, as given by (2), occurs in parallel on each cycle. Once the index which is closely correlated to y is found, the residual is updated by subtracting it with the correlation of the column of Φ as shown in (5). Figure 3(a) shows the block diagram for solving the optimization problem and is repeated to find m indices of Φ as given by (3) and (4).

C. Solving Least Square Problem

After finding m columns of Φ which are closely correlated to y, the second stage is to solve the least square problem given by (6). This often involves finding the inverse of an 8x8 matrix C where $C = \widetilde{\Phi}^T\widetilde{\Phi}$. The main purpose of this is to solve for $\hat{x}$ from:

$$(\widetilde{\Phi}^T\widetilde{\Phi})\hat{x} = \widetilde{\Phi}^T y \tag{7}$$

There are numerous methods available to solve C^{-1} and the method followed in [7] was Alternate Cholesky Decomposition which decomposes C as LDL^T, where L is a lower triangular matrix and D is a diagonal matrix. Though it is better than many other methods like Cholesky decomposition [10], this approach can exponentially take increase the latency with increasing dimension of C. This can be proven since the total number of values (k) to be calculated for L increases quadratically with the dimension (m) of C and is given by $k = m(m-1)/2$. In Q-R decomposition process [5], C is decomposed into QU where Q is an orthogonal matrix and U is a upper triangle matrix. Now the inverse of C is given by,

$$C^{-1} = U^{-1}Q^T \tag{8}$$

Unlike calculating L in ACD, in QRD process, each column is processed one at a time to generate Q. This proves that

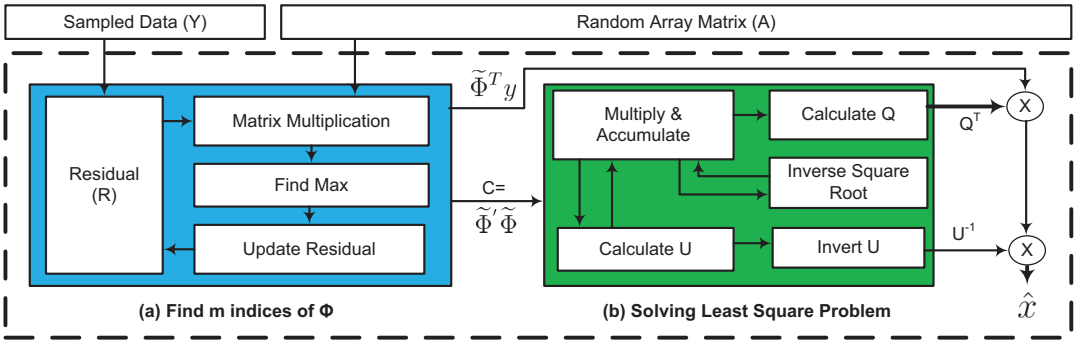

Fig. 3. Detailed architecture diagram of OMP reconstruction algorithm. (a) This block iterates 8 times to solve the optimization problem. Operates at 85 MHz. (b) Finds inverse of a 8x8 matrix after finding 8 columns of Φ. Operates at 69 MHz.

the calculation of Q is linear with the dimension of C. U is generated while calculating Q and is given by the pseudocode:

for $i = 1$ **to** n **do**
  $U_{i,j} = M_j^{'} \cdot C_i, \quad j = 1, 2, ..., i-1$
  $G_i = C_i - \sum_{k=1}^{i-1}(M_k \times U_{i,k})$
  $U_{i,i} = norm(G_i)$
  $M_i = G_i / U_{i,i}$
end for

where $norm(X) = \sqrt{X_1^2 + X_2^2 + .. + X_n^2}$. A set of 8 multipliers are used in parallel to perform these steps. The critical problem in this method is to find the square root and also the division in the final stage of each column processing. In this proposed architecture, a pipelined fixed point inverse square root algorithm has been implemented which takes 6 clock cycles. Since division is a more complex and time consuming arithmetic operation than multiplication, it is clear that this algorithm does not involve any division arithmetic as compared to ACD [7]. Now, we have to find U^{-1} (denoted as V) and since it is an upper triangular matrix, the steps to find the inverse are given,

for $i = 1$ **to** n **do**
  $V_{i,i} = 1/U_{i,i}$
  for $j = i + 1$ **to** n **do**
    $V_{j,i} = -V_{i,i} \times \sum_{k=i}^{j-1}(U_{j,k} \times V_{k,i})$
  end for
end for

Since we already found $1/U_{i,i}$ through inverse square root algorithm, the division is again eliminated here. Finding V requires 7 multipliers in parallel and the 8 multipliers used for finding Q is reused here. Hence the inputs are given to the multipliers through a multiplexer and V is calculated in parallel. It can be noted that the residual registers are not used after the optimization problem and these registers are reused for C and Q, thus not taking additional area. The detailed block diagram for QRD is shown in Fig. 3(b).

D. Fast Inverse Square Root

Fast inverse square root method was developed and appeared in *Quake III Arena* source code for a 32-bit floating point number [6]. The original algorithm takes a 32-bit unsigned floating point number and stores half its value. The floating point number is right shifted by 1 and the result is subtracted from a number 0x5f3759df [6]. This gives a close approximation of the inverse square root. Then the Newton's method is used multiple times to achieve a more accurate results. In this paper, we optimize this method to calculate inverse square root for a 24-bit fixed point number. Here, all the 24 bits are considered as fraction bits and the input is left shifted (2 bits at a time) untill the first 2 MSB bits become either 01,10 or 11, denoted as $X(X_{24}X_{23}..X_1)$. This is to normalize any number to a fraction between 0.25 and 0.5. Then the fixed point equivalent of floating point right shift is obtained as given by (9). The results is in 2.22Q format.

$$X_{new} = \begin{cases} 00X_{23}X_{22}...X_2 & \text{if } X_{24} = 1 \\ 11X_{22}X_{21}...X_1 & \text{if } X_{24} = 0 \end{cases} \tag{9}$$

This X_{new} is subtracted from the value 0xB759DF for 24-bit 2.22Q fixed point. This gives the first approximation (Y_1) given by,

$$Y_1 = 0xB759DF - X_{new} \tag{10}$$

$$Y_{i+1} = Y_i \times (1.5 - (Y_i^2 \times X_{new}/2)) \ , \ i = 1, 2 \tag{11}$$

Then the Newton's method is performed for 2 iterations to obtain an approximation of inverse square root. The final value Y_2 is left shifted (1 bit at a time) by exactly the same number of times done for generating X. It uses a single multiplier and is pipelined to perform one multiplication per cycle thus producing the result in 6 clock cycles.

E. Finding The Estimated Values

The ultimate goal is to find the estimate $\hat{x}$ which is given by $\hat{x} = (\widetilde{\Phi}^T\widetilde{\Phi})^{-1}\widetilde{\Phi}^T y$ where $C = \widetilde{\Phi}^T\widetilde{\Phi} = QU$. From the above steps we will have $\widetilde{\Phi}$, Q and U^{-1}. Instead of calculating C^{-1}, we will optimize the computations by first calculating $\widetilde{\Phi}^T$y and then calculating $Q^T\widetilde{\Phi}^T y$ and finally calculating:

$$\hat{x} = U^{-1}Q^T\widetilde{\Phi}^T y \tag{12}$$

Bypassing the C^{-1} calculation and using it in the decomposed form proved to be very efficient as opposed to the method implemented in [7]. $\hat{x}$ has 8 values, each is a 24-bit fixed point number. These values correspond to the indices denoted by Λ as seen in (3) while the remaining values are just zeros.

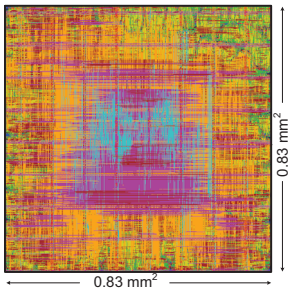

Fig. 4. Post layout view of the proposed CS reconstruction hardware

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

TABLE I
IMPLEMENTATION SUMMARY FOR THE PROPOSED CS RECONSTRUCTION HARDWARE. * DENOTES THE RECONSTRUCTION FOR 256–LENGTH INPUT AND A SPARSITY OF 8.

Since the residual length is 64 and each matrix has 64 values, the above calculations are performed in just 3 clock cycles using the parallel 64 multipliers.

IV. CMOS Implementation and Comparison

The high performance compressive sensing hardware is implemented in 65 nm CMOS technology that operates at 1 V supply voltage. The hardware architecture was designed in Verilog, synthesized using RTL Compiler, and placed and routed using Cadence Encounter. Fig. 4 shows the chip layout of the proposed hardware. Place and route results indicate that the chip occupies a total area of 0.69 mm^2 and operates at 165 MHz. The design is also sythesized for Xilinx Virtex 5 FPGA to compare with the previously published implementations. It runs at two different clocks, 85 MHz and 69 MHz for blocks (a) and (b), respectively as in Fig. 2. For sparsity m = 8, it requires 2100 clock cycles to find 8 columns of Φ and 160 cycles for QRD block. Hence the total reconstruction time is 27.14 μs. The real bottleneck of the algorithm is finding the dot product of Φ and residual R that takes $256 \times 8 = 2048$ cycles.

The proposed architecture for 128–length vector of sparsity 5 on Xilinx FPGA Virtex-5 will take a maximum of 688 cycles for finding 5 columns of Φ and 132 cycles for QRD process. This gives a reconstruction time of 10 μs which is 2.4 times [11] faster than the previous implementation specified in [7] whose reconstruction time is 24 μs and nearly 3 times faster in reconstructing a 128×128 image as compared to [12].

A reconstructed 256×256 image using the proposed method is shown in Fig. 5. The image is divided into 256 sub-images of dimension 16×16 and the reconstruction is performed sequentially. The reconstructed image is enhanced using a 6×6 median filter.

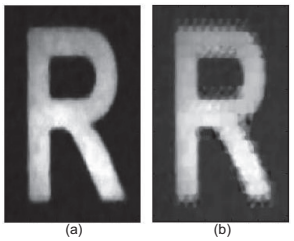

Fig. 5. (a) Original Image (b) Reconstructed Image (enhanced*) using the proposed method, PSNR = 31.85 dB. The 256×256 image is divided into sub-images of dimension 16×16 and iterated for 256 times. * denotes the reconstructed image is enhanced using a 6×6 median filter.

V. Conclusion

This paper presents an improved architectural design and implementation of a high performance CS reconstruction hardware. This hardware supports vector of length 256 and a fast Q-R decomposition process has been implemented to find matrix inverse for higher sparsity. This is supported by the implementation of a fully pipelined fast inverse square root algorithm. The reconstruction time for a 256–length signal of sparsity 8 is 13.7 μs. The same architecture implementation on Xilinx FPGA Virtex–5 for 128–length vector takes 10 μs which is 2.4 times faster than the state–of–the–art implementation.

References

[1] E. Candès and M. Wakin, "An introduction to compressive sampling," *Signal Processing Magazine, IEEE*, vol. 25, no. 2, pp. 21–30, Mar 2010.
[2] J.-L. Starck, F. Murtagh, and J. Fadili, *Sparse Image and Signal Processing*. Cambridge University, 2010.
[3] E. Candès, "Compressive sampling," in *Proceedings of the International Congress of the Mathematicians*, 2006, pp. 1433–1452.
[4] 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.
[5] M. Karkooti, J. Cavallaro, and C. Dick, "FPGA Implementation of Matrix Inversion Using QRD-RLS Algorithm," *Signals, Systems and Computers, 2005. Conference Record of the Thirty-Ninth Asilomar Conference on*, pp. 1625–1629, 2006.
[6] "Fast inverse square root," Aug. 3 2011. [Online]. Available: http://en.wikipedia.org/wiki/Fast_inverse_square_root
[7] A. Septimus and R. Steinberg, "Compressive sampling hardware reconstruction," in *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*, 2010, pp. 3316–3319.
[8] D. Yang, H. Li, G. Peterson, and A. Fathy, "Compressed sensing based UWB receiver: Hardware compressing and FPGA reconstruction," *Information Sciences and Systems, 2009. CISS 2009. 43rd Annual Conference on*, pp. 198–201, 2009.
[9] M. Andrecut, "Fast GPU implementation of sparse signal recovery from random projections," 2008. [Online]. Available: http://www.arxiv.org/PS_cache/arxiv/pdf/0809/0809.1833v1.pdf
[10] O. Maslennikow, P. Ratuszniak, and A. Sergyienko, "Implementation of Cholesky LLT-Decomposition Algorithm in FPGA-Based Rational Fraction Parallel Processor," *Mixed Design of Integrated Circuits and Systems, 2007. MIXDES '07. 14th International Conference on*, pp. 287–292, 2007.
[11] J. Stanislaus and T. Mohsenin, "Low-complexity fpga implementation of compressive sensing reconstruction," *SPIE Conference on Defense, Security, and Sensing*, April 2012.
[12] Y. Chen and X. Zhang, "High-speed architecture for image reconstruction based on compressive sensing," *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, pp. 1574–1577, 2010.