Hardware efficient architecture for compressed imaging

Jun Luo, Qijun Huang, Sheng Chang, and Hao Wang
Department of Electronic Science and Technology, School of Physics and Technology, Wuhan University, 16 Luojiashan, Wuchang, Wuhan, China
a) huangqj@whu.edu.cn

Abstract: Compressed sensing has gained a wide application in image acquiring and reconstructing. Separable linear reconstruction has been shown to be effective in compressed imaging. This paper presents efficient hardware architecture based on adaptive sampling and separable reconstructing. By exploiting parallel properties in the architecture and timing scheme, high performance hardware has been proposed for both encoding and decoding sides. High performance Cholesky based matrix inversion has been implemented to solve the least square problem. Besides, high precision arithmetic element functions (reciprocal and square root reciprocal) have been presented by using of table interpolation and single iterated Newton-Raphson method. Experiment results show that the proposed hardware architecture can efficiently reduce the process time in encoding and decoding of a 512 × 512 image. The speedup is about 58× compared with the software-based approach (using LAPACK), and it is at least 1.92× faster than the state-of-the-art implementation.

Keywords: compressed imaging, adaptive sampling, separable reconstruction, hardware architecture

Classification: Electron devices, circuits, and systems

References
[1] D. L. Donoho: IEEE Trans. Inf. Theory 52 (2006) 1289. DOI:10.1109/TIT.2006.871582
[2] E. J. Candes and T. Tao: IEEE Trans. Inf. Theory 52 (2006) 5406. DOI:10.1109/TIT.2006.885507
[3] H. Zheng and X. Zhu: J. China Univ. Post Telecommun. 20 [3] (2013) 97. DOI:10.1016/S1005-8885(13)60056-4
[4] J. A. Tropp and A. C. Gilbert: IEEE Trans. Inf. Theory 53 (2007) 4655. DOI:10.1109/TIT.2007.909108
[5] J. Yang, Y. Zhang and W. Yin: IEEE J. Sel. Top. Sign. Proces. 4 (2010) 288. DOI:10.1109/JSTSP.2010.2042333
[6] J. Luo, Q. J. Huang, S. Chang and H. Wang: IEICE Electron. Express 11 (2014) 20140056. DOI:10.1587/elex.11.20140056
[7] Y. Chen and X. Zhang: IEEE Int. Conf. Acoust. Speech Signal Process. Proc. (2010) 1574.
[8] A. Septimus and R. Steinberg: IEEE Int. Symp. Circuits Syst. (2010) 3116.
1 Introduction

Compressed sensing (CS) [1] is a novel sampling scheme, which has significant impact on wide imaging applications. It goes beyond the Shannon-Nyquist theory, showing a bright future in reconstructing the original signal at a less sampling rate than Nyquist without aliasing. Low sampling rate can release the computational burden in signal acquiring while the exact recovery can be met under the restricted isometry property (RIP) [2] condition. Promising applications include medical imaging, wireless communication, image and audio processing. Although compressed imaging has several advantages, the reconstruction process is complex and computational intensive. Several recovery algorithms have been proposed in recent years. Non-linear algorithms such as smoothed projected landweber (SPL) [3], orthogonal matching pursuit (OMP) [4] and total variation (TV) minimization [5] are showing favorable image quality while they require significant computation effort, even for problems of moderate size. In our previous work [6], separable linear reconstruction (SLR) has been proposed. The use of separable recovery shows a higher image quality comparing with OMP. However, the software implementations of such algorithms are mostly suitable for applications that allow for off-line process. The implementations can become extremely challenging for applications requiring real-time processing or with tight power constraints on remote devices. Hence, to meet the stringent throughput, latency, and power-consumption constraints of real-time applications, the efficient dedicated hardware architecture becomes necessary.

Previous hardware reports of CS have been mainly focused on the algorithms of split bregman algorithm (SBA) [7], OMP [8, 9, 10, 11], and approximate message passing (AMP) [10]. In [7], it takes 38.2 ms to reconstruct a $128 \times 128$ image using SBA. In [8], a Cholesky based OMP has been implemented in Virtex-5 field gate programmable array (FPGA). It achieves a maximum speed 39 MHz and takes 24 us to reconstruct 128 samples (measurement rate: $K = 0.25$; sparsity: $S = 5$). Using the same device as in [8], the Q-R decomposition based OMP is implemented in [9]. It reaches 69 MHz and takes 27.14 us in recovering 256 samples ($K = 0.25; S = 8$). A threshold method is proposed in [11] to further reduce the process time of OMP. It reaches 85.69 MHz in Virtex-5 and takes 7.13 us to reconstruct 128 samples ($K = 0.25; S = 5$). Two algorithms (OMP and AMP) have been implemented in Virtex-6 in [10]. The maximum frequency is 100 MHz and 165 MHz for OMP and AMP, respectively. It takes 0.63 ms in processing a $32 \times 32$ image. However, the above mentioned reports have been mainly related to a small size of problems.
(no more than 256 × 256 samples). When the problem is large, they cannot afford real-time or near real-time processing. Besides, they all use high performance Xilinx FPGAs which are high cost. Thus, their performance of CS will be limited if a low cost FPGA is used.

In this paper, we expand our work [6] for CS image reconstruction on FPGAs, using efficient hardware architecture. No hardware architecture of SLR algorithm for CS has been reported to the best of my knowledge. The real-time processing can be achieved for solving a 512 × 512 image. The efficiency is achieved by exploiting parallel properties in the architecture and timing scheme. High performance Cholesky based matrix inversion is presented to solve the least square problem, and high precision arithmetic element functions (reciprocal, square root reciprocal) are implemented by a combination of table interpolation and Newton-Raphson (NR) method. The proposed architecture is synthesized target at a low cost Altera Cyclone IV E FPGA. Experiment results have shown that our report is 58× faster than the Linear Algebra PACKage (LAPACK) based software approach, and is 1.92× faster than the state-of-the-art implementation [11].

2 Proposed architecture

2.1 Algorithm

In CS theory, sparse represented signal can be exactly reconstructed when RIP is met. In [6], a fast reconstruction is proposed. It uses separable linear operator and adaptive sampling to yield a competitive image quality. The adaptation is employed by assigning different sampling rate to each block of an image according the variance. Suppose an image has $N$ pixels and $M$ samples, the measurement rate (MR) of the image is: $K = M/N$. The image is divided into $B \times B$ blocks, thus the mean ($\overline{\sigma^2}$) of all blocks’ variance is denoted by Eq. (1), where $\sigma_i^2$ is the variance of block $i$ ($i = 1, 2, \cdots, N/B^2$). The MR of each block can be acquired by Eq. (2) and Eq. (3), where $r_i$ is the measurement rate of block $i$, and $\gamma$ is a constant used to control the minimum sampling rate.

$$\overline{\sigma^2} = \left( \sum_{i=1}^{N/B^2} \sigma_i^2 \right) / (N/B^2) \quad (1)$$

$$r_i = K \cdot \sigma_i^2 / \overline{\sigma^2} \quad (2)$$

$$r_i = \begin{cases} 1, & r_i \geq 1 \\ K/\gamma, & r_i \leq K/\gamma \\ r_i, & \text{else} \end{cases} \quad (3)$$

Let $x$ be the original image, $y$ be the measurement sample and $\hat{x}$ be the recovered image. In the minimum mean square error (MMSE) based linear estimation, estimated error $e$ can be expressed by Eq. (4), where $H$ is the reconstruction operator. By solving Eq. (5), the optimal solution for $H$ can be expressed in Eq. (6), where $\phi_H$ is the Gaussian random measurement matrix and $R_{xx}$ is the auto-correlation matrix determined by Eq. (7). $\rho$ and $\tau$ is the correlation coefficient and Euclidean distance, respectively. The reconstruc-
tion process is accomplished by separable operator \((H_h \text{ and } H_v)\) determined in Eq. (8) sequentially. The operators are different block-by-block and can be affected by the measurement rate \((r_i)\) in each block.

\[
e = \hat{x} - x = Hy - x
\]

\[
\arg \min_{H} \{ R_{xx} = E[ee^T] \} \quad \text{subject to } e = Hy - x
\]

\[
H_{opt} = R_{xx} \phi_B^T \left( \phi_B R_{xx} \phi_B^T \right)^{-1}
\]

\[
R_{xx} = \rho^T
\]

\[
H_h = H_v = H_{opt}
\]

2.2 Hardware architecture

The system hardware architecture of compressed imaging is shown in Fig. 1, where \(\otimes\) denotes the matrix multiplication. The 512 \(\times\) 512 image is stored in an off-chip memory (SRAM). It is accessed and processed by the on-chip modules through the transferring bus. There are three functional modules in the proposed architecture. The module of \textit{adaptive} \_top, \textit{enc} \_top and \textit{dec} \_top is accomplishing the function of adaptive sampling, encoder and decoder, respectively. Adaptive sampling is used to get the block measurement rate \((r_i)\).

There are 256 blocks in a 512 \(\times\) 512 image. The size of each block is 32 \(\times\) 32. The Gaussian random matrix \(\phi_B\) is reduced by a random matrix \(Q\) with binary input 0 and 1. It stores in the two-port (read and write) random access memory (RAM). Thus the matrix manipulations in the encoder side consume no multipliers. On the decoder side, the auto-correlation matrix \(R_{xx}\) is pre-computed and stored in the read only memory (ROM). No multiplier is used in the \(Q \otimes R\) module and \(Q \otimes F^T\) module. In order to design a hardware efficient architecture, parallel properties are exploited in the architecture and timing scheme. 32 parallel channels are implemented in each sub-module and three-port (two read and one write) RAM is used to store the calculated data. Therefore, the matrix manipulated data can be updated circle by circle and be written to the RAM sequentially.

Fig. 1. Block diagram of hardware architecture for compressed imaging system

When the measurement rate of all blocks is computed, the encoder and decoder sides can be started at the same time. This parallel timing scheme
illustrated in Fig. 2 can be exploited further to reduce the process time. There are three tasks in Fig. 2. Task ➊ denotes the adaptive sampling process, task ➋ is the encoder process and task ➌ represents the decoder process. Task ➋ and task ➌ can be processed parallel as shown in Fig. 2. The decoding process is the critical path because it takes the longest time to process a block.

2.3 Adaptive sampling
Adaptive sampling includes computing the variance and calculating the measurement rate of each block as shown in Eq. (1), Eq. (2) and Eq. (3). Architectures of these two processes are shown in Fig. 3 and Fig. 4. 32 arrays of three-port RAM (index size: 32 × 32) is used to store the image blocks and 16 arrays of three-port RAM (index size: 16 × 16) is employed to store the variance and measurement rate of each block. In Fig. 4, CSrate (K) is the pre-defined measurement rate set for a 512 × 512 image. The square root (sqrt) module is implemented by the arithmetic element function (square root reciprocal) using a full pipelined architecture as introduced in section 2.5.2. Indicated in Fig. 3 and Fig. 4, parallel architectures have been proposed for adaptive sampling. There are 32 channels in Fig. 3 and 16 channels in Fig. 4. The parallel channels are exploited to reduce the process time.

2.4 Compression
Parallel processes have been exploited in the encoder side as shown in Fig. 5. The module of blk_enc_SRAM is used to load block image data from the off-chip SRAM to the on-chip RAM. As a tradeoff of complexity and performance, 32 channels are proposed in the hardware architecture. Multi-port RAM is used to store the buffer data. There are two Process Elements (PEs), which
are used to accomplish the accumulation process. Architecture of the PE1 is shown in Fig. 6, and the same structure is shared in PE2. The parallel channels and multiplier-free approach lead to fast hardware architecture for encoder.

2.5 Solving optimization problem

In order to get the optimal operator \(H_{opt}\), Eq. (6) has to be solved. After that, the estimated image is reconstructed through matrix product. Parallel
properties are exploited in matrix multiplication (32 channels) similar to the structure suggested in section 2.4. It also uses the Cholesky decomposition and exploits the parallel data dependency to construct high performance matrix inversion architecture.

2.5.1 Cholesky based matrix inversion

Matrix inversion is used to solve the least square problem in decoder side. When the input matrix is positive symmetric definite, it can be decomposed using Cholesky. The algorithms used for matrix inversion is shown in Fig. 7, where symbols \((i, j, \text{ and } k)\) are index numbers, \(n\) is the order of input matrix \((A)\), \(a\) is the element of \(A\). It will be updated by the Cholesky decomposition. \(U\) is the upper matrix derived by Cholesky factoring.

```
for k=1:n
    a(k,k)=sqrt(a(k,k));
for i=k+1:n
    a(i,k)=a(i,k)/a(k,k);
end
for j=k+1:n
    for i=j:n
        a(i,j)=a(i,j)-a(i,k)*a(j,k);
    end
end
mu=[U;eye(n)];
for i=1:n-1:1
    mu(i,:)=mu(i,:)/mu(i,i);
end
for j=1:n
    U^j=mu(:,n+1:2*n);
end
```

Fig. 7. Algorithms for matrix inversion: Cholesky decomposition (left), upper matrix inversion (middle), and matrix multiplication (right)

The architecture for Cholesky based matrix inversion is illustrated in Fig. 8. Cholesky decomposition, upper matrix inversion and matrix multiplication are the three sequence processes needed to accomplish the matrix inversion. Column division \((\text{Div}_\text{col})\) and column refresh \((\text{Ref}_\text{col})\) are employed in the Cholesky decomposition. Raw division \((\text{Division})\) and raw refresh \((\text{Ref}_\text{raw})\) are used in the upper matrix inversion. They use the same three-port RAM (memory size: 1024 \(\times\) 66 bits) to store processed data.

2.5.2 Arithmetic element functions

Arithmetic element functions (reciprocal, square root and square root reciprocal) are playing important roles in digital signal processing and scientific computing. In this paper, table approximation and single iterated multiplicative method are combined to yield high performance arithmetic hardware architecture. Suppose the table size is: \(2^h \times m\) bits. The binary format of table input \((1 < T_i < 2)\) and output \((0.5 < T_o < 1)\) can be expressed in Eq. (9) and Eq. (10). The most significant bit of input is always 1 and does not need to be included in the look up table address. Also, the leading 0.1 of output does not need to be stored in each look up table entry. Therefore, the table entries of reciprocal and square root reciprocal can be determined by Eq. (11) and Eq. (12), where \(i\) is the index number of address.
TableEntry\((i)\) = \(\left(\frac{1}{1 + addr(i) \times 2^{h-h} + 2^{-k}}\right)_{m+1} - \frac{1}{2}\) \(\times 2^{m+1}\) (11)

The table approximation is used to get the initial guess \((p_0)\). Single iterated NR method is employed to achieve a high precision. The iterative formulas between input \((0 < t < 1)\) and output \((p)\) can be represented by Eq. (13) and Eq. (14) for reciprocal and square root reciprocal, respectively. \(n\) is the number of iterations and \(n = 1\) is chosen in this paper. To be pointed out, the result of square root can be computed by input value \((t)\) multiplied with its square root reciprocal, using an additional multiplier.

\[ p_{n+1} = p_n(2 - tp_n) \] (13)

\[ p_{n+1} = p_n(3 - tp_n^2)/2 \] (14)

Fig. 9 and Fig. 10 illustrate the hardware architecture for reciprocal and square root reciprocal. In these two figures, \(w\) is the width of input data \((t)\), \(num\) is the number of leading zero \((NLZ)\), and \(addr\) is address used to get the table entry. There are two multipliers and four multipliers in the reciprocal and square root reciprocal structures respectively. The multipliers are implemented using the Altera Intellectual Protocol (IP). There is a shifting process in the last stage in achieving result \((p)\). The shifting factor \((a)\) is varied depending on the width of \(t\) and the precision of \(p\). The parameters of arithmetic element functions presented in adaptive sampling and matrix inversion modules are shown in Table I. In order to design high throughput architectures for the mass data processes, full pipelined architectures are implemented. This allows the valid data can be refreshed at each circle.
3 Results and comparisons

To evaluate the efficiency of the proposed hardware, the architecture has been implemented and synthesized for Cyclone IV EP4CE115F29C7 FPGA. The synthesized result is shown in Table II, where the results for adaptive sampling (adaptive_top), encoder (enc_top), decoder (dec_top) and system (system_top) modules are illustrated. It can be found that the proposed system consumes 54% of logic elements (LEs), 46% of dedicated logic registers, 9% of memory bits and 100% of embedded multiplier 9-bit elements for the target FPGA device. The mass consumed multipliers can be explained that parallel matrix manipulation is employed in the proposed architecture. As a result of high performance architectures implemented for matrix inversion and arithmetic element functions, the proposed system can reach a maximum frequency of 122.22 MHz, which is about 50% of the extreme limit (nearly 250 MHz) for Cyclone IV E.

The functional and timing simulations are accomplished by Modelsim SE 10.1c. Verification has been completed based on the platform DE2-115 with the system clock frequency running at 100 MHz. Fig. 11 illustrates the verified result for Lena image when the measurement rate is 0.3. The dB loss of hardware is about 0.53 dB compared to the software based approach. The

Table I. Parameters of different arithmetic element functions

| Module            | Arithmetic element function | Input bits | Output bits | Table size | Pipelines for multipliers |
|-------------------|-----------------------------|------------|-------------|------------|--------------------------|
| adaptive sampling | reciprocal                  | 10         | 15          | $2^7 \times 7$ | 4                        |
|                   | square root reciprocal      | 10         | 14          | $2^7 \times 7$ | 4                        |
| matrix inversion  | reciprocal                  | 31         | 41          | $2^{10} \times 10$ | 6                        |
|                   | square root reciprocal      | 64         | 44          | $2^{10} \times 10$ | 6                        |
reason of the dB loss is that a fixed-point representation is employed in hardware other than a double-float point is adopted in software. The fixed-point representation can lead to low power consumption at the cost of negligible image quality loss.

In order to evaluate the image quality of different algorithms, peak signal to noise ratio (PSNR) according to the measurement rate for 512 × 512 Lena image is shown in Fig. 12. It can be seen that SLR in [6] outperforms other algorithms (SPL [3], OMP [4] and RecPF [5]) in terms of objective image quality. There is an average 4.14 dB improvement in SLR over OMP for Lena image. Compared to [6], the average dB loss in the proposed hardware architecture is 0.55 dB for different measurement rate. Thus, the average dB improvement for the proposed hardware architecture is 3.59 dB over OMP.

To evaluate the efficiency of the proposed hardware architecture, comparisons of software- and hardware-based approaches are given for the proposed blocked adaptive sampling SLR algorithm in [6] (*AS_BCS_SLR* in Fig. 12). The software is running in Matlab based on four Intel i5-3470 3.2 GHz CPUs and 8G RAM. There are two software based approaches as shown in Fig. 13. One approach (A1) uses no LAPACK and the other (A2) uses it. The process time denoted in Fig. 13 includes the necessary process time (adaptive sampling, compression and reconstruction) in reconstructing one 512 × 512 image. In Fig. 13, the process time of the proposed hardware is measured running at 100 MHz. More process time will be consumed when the measurement rate (*r*) is higher. It can be seen that the proposed hardware is 5931 ×
faster than A1 and 58× faster than A2 on average. The average process time of the proposed hardware is 51 ms (512 × 512 image, running at 100 MHz), which can meet the real-time processing with an average of 19.6 frame/s. It supports the real-time processing from 13.5 frame/s ($K = 0.5$) to 45.4 frame/s ($K = 0.1$). Thus, a hardware efficient architecture for compressed imaging has been proposed. It can achieve the real-time processing for a 512 × 512 image by using a low cost Cyclone IV E FPGA.

Table III illustrates the comparison to previous reports. Previous reports have been mainly focused on OMP algorithm. They are all targeted at a high cost Virtex FPGA and the handled problem size is no more than 256 × 256 samples. No hardware architecture based on SLR algorithm for CS has been reported so far. In Table III, the recovery time for a 512 × 512 image is measured at: $K = 0.25$. OMP is an iterated algorithm, the number of iterations affect the recovery time. For a fairly comparison, the sparsity as suggested in the corresponding paper is used as the number of iterations so as to indicate the total recovery time for a 512 × 512 image. It can be seen that the proposed hardware is 5.87× faster than [9], 4.25× faster than [10] and 1.92× faster than the state-of-the-art implementation [11]. It also should be
pointed out that the reconstructed image quality from the proposed hardware is better than OMP as it is indicated in Fig. 12. Besides, a higher speed is achieve in this paper for a low cost FPGA compared to previous reports which are targeted at high cost FPGAs. Therefore, a hardware efficient architecture by exploiting the parallel properties in both architecture and timing scheme is proposed in this paper.

| Report | Algorithm | Platform | Fmax/MHz | Recovery time/ms |
|--------|-----------|----------|----------|------------------|
| [7]    | SBA       | —        | —        | 611.2            |
| [8]    | OMP       | Virtex-5 | 39       | 245.76           |
| [9]    | OMP       | Virtex-5 | 69       | 222.33           |
| [10]   | OMP       | Virtex-6 | 100      | 161.28           |
| [11]   | OMP       | Virtex-6 | 165      | 73.01            |
| Proposed | SLR     | Cyclone IV E | 122.22  | 37.87 |

4 Conclusions

A hardware efficient architecture for blocked compressed imaging has been proposed in this paper. The proposed hardware architecture is based on the separable linear recovery algorithm for CS. By exploiting the parallel properties in both architecture and timing scheme, a real-time compressed imaging system is achieved. The real-time process supports 13.5 frame/s to 45.4 frame/s. High performance matrix inversion architecture is presented. It bases on the Cholesky decomposition and exploits the parallel data dependency. High precision arithmetic element functions are gained using the combination of table interpolation and single iterated NR method. The full pipeline scheme also leads to a high throughput structure. Compared to the software based approach (LAPACK), our proposed hardware is about 58× faster. It also shows at least 1.92× speedup compared with the state-of-the-art implementation.

Acknowledgments

The work is supported by the Fundamental Research Funds for the Central Universities, Wuhan University (201202020204 and 2042014kl0238), and by the National Natural Science Foundation of China (61204096 and J1210061).