A Hybrid Architecture for Compressive Sensing 3-D CT Reconstruction

Jianwen Chen, *Senior Member, IEEE*, Jason Cong, *Fellow, IEEE*, Luminita A. Vese, John Villasenor, Ming Yan, and Yi Zou

***Abstract*—The radiation dose associated with computerized tomography (CT) is significant. Compressive sensing (CS) methods provide mathematical approaches to reduce the radiation exposure without sacrificing reconstructed image quality. However, the computational requirements of these algorithms is much higher than conventional image reconstruction approaches such as filtered back projection (FBP). This paper describes a new compressive sensing 3-D image reconstruction algorithm based on expectation maximization and total variation, termed EM+TV, and also introduces a promising hybrid architecture implementation for this algorithm involving the combination of a CPU, GPU, and FPGA. An FPGA is used to speed up the major computation kernel (EM), and a GPU is used to accelerate the TV operations. The performance results indicate that this approach provides lower energy consumption and better reconstruction quality, and illustrates an example of the advantages that can be realized through domain-specific computing.**

***Index Terms*—Compressive sensing, computerized tomography (CT) image reconstruction, expectation maximization (EM), field-programmable gate array (FPGA), graphics processing unit (GPU), iterative reconstruction, total variation (TV).**

I. Introduction

COMPUTERIZED tomography (CT) plays a critical role in modern medicine. However, the radiation associated with CT is significant, and researchers are exploring various approaches to reduce the radiation. Traditionally, image reconstruction requires that the number of samples (measurements or observations) be dictated purely by Nyquist limits. However, methods such as compressive sensing that exploit object sparsity can enable CT imaging with less data and therefore less radiation exposure, without sacrificing image quality.

Conventionally, the Feldkamp–Davis–Kress (FDK) algorithm has been used for 3-D cone-beam CT image reconstruction, and it is widely employed in the machines used in clinical settings. The computation kernel of the FDK algorithm is called filtered back projection (FBP). Generally, FBP is used for 2-D images in association with parallel beam CT [1]. Iterative reconstruction has also been proposed [2], [3] to reconstruct 2-D and 3-D images from the projections of an object. An iterative framework can be used in many applications, including computerized tomography (CT), positron emission tomography, and magnetic resonance imaging. The main advantages of iterative reconstruction over FBP are reduced sensitivity to noise and increased data collection flexibility [4]. For example, the data can be collected over any set of lines, the projections do not have to be distributed uniformly, and the reconstruction can also be performed when projections are available for a limited set of angles.

Various iterative reconstruction/compressive sensing algorithms have been proposed with different objectives or regularization terms. However, many of these algorithms, including expectation maximization (EM) [5] and simultaneous algebraic reconstruction technique (SART) [6], share a common underlying computational approach that includes a forward ray tracing step (forward projection) and a backward ray tracing step (backward projection).

In the present paper, we focus on a recent compressive sensing algorithm (EM+TV) that combines the expectation maximization (EM) method using Poisson noise with the total variation (TV) regularization. The only precondition of EM+TV algorithm is that the reconstructed image cannot have an excessive total-variation. For CT images, this precondition is true and the EM+TV algorithm can be applied. The effectiveness of this compressive sensing algorithm has been well documented [7].

The EM+TV method gives superior results to those obtained by FBP or EM-only and has the additional important advantage of involving reduced radiation dose.

For example, as illustrated in Fig. 1, the root mean square error (RMSE) metrics is used to evaluate the algorithm performance. The EM+TV algorithm using 36 views can obtain an image quality similar to that obtained using an FDK/FBP algorithm with 360 views, corresponding to an order of magnitude reduction in radiation. Traditionally, FDK/FBP algorithms, which directly calculate the image in a single backward reconstruction step, have been accelerated with GPUs or FPGAs [1], [8]–[11]. However, when the number of samples is reduced, FDK methods generally generate very poor-quality images.

Thus, there is a strong motivation to accelerate iterative reconstruction methods for practical CT systems. However, while there has been a substantial amount of previous work aimed at using a graphics processing unit (GPU) [12]–[14] to accelerate iterative reconstruction approaches like SART, there have

Manuscript received March 01, 2012; revised August 19, 2012; accepted September 12, 2012. Date of publication October 23, 2012; date of current version December 05, 2012. This work was supported by the Center for Domain-Specific Computing (CDSC) under the NSF Expeditions in Computing Award CCF-0926127. This paper was recommended by Guest Editor D. Allstot.

J. Chen, J. Cong, and Y. Zou are with the Department of Computer Science, University of California, Los Angeles, CA 90095 USA (e-mail: jianwen.chen@ieee.org; cong@cs.ucla.edu; zouyi@cs.ucla.edu).

L. A. Vese and M. Yan are with Department of Mathematics, University of California, Los Angeles, CA 90095 USA (e-mail: lvese@math.ucla.edu; yanm@math.ucla.edu).

J. Villasenor is with the Department of Electrical Engineering, University of California, Los Angeles, CA 90095 USA (e-mail: villa@ee.ucla.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JETCAS.2012.2221530

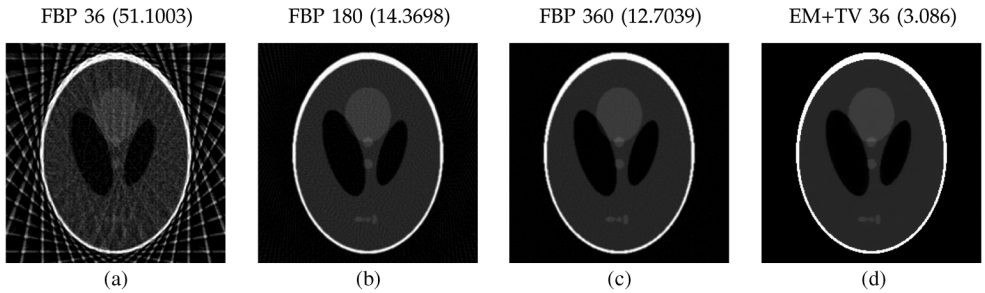


Fig. 1. Reconstruction results by FBP with 36, 180, 360 views and EM+TV with 36 views (RMSE numbers are shown in parenthesis). (a) FDK/FBP with 36 views. (b) FDK/FBP with 180 views. (c) FDK/FBP with 360 views. (d) EM+TV with 36 views.

been far fewer publications addressing field-programmable gate array (FPGA) implementations of iterative reconstruction. In [15], for example, backward projection was implemented on an FPGA, and the forward projection step was performed on a GPU. GPUs and FPGAs of course have very different features. GPUs can have hundreds of parallel computing cores, and FPGAs can support high performance logic customization for specific computations. If, for implementing an algorithm such as EM+TV that has significant computational diversity, the architecture advantages of both a GPU and an FPGA can both be exploited, a higher performance design can be expected. Moreover, the use of FPGA can help to significantly reduce the power consumption of the overall system.

This paper presents a hybrid architecture for the EM+TV compressive sensing algorithm [7] for CT image reconstruction. In this design, for EM part, the computations involved in the EM forward/backward ray tracing steps are based on MADD (multiply and add), requiring significant off-chip random access. Moreover, the computation and the required data access for one ray is proportional to the intersection length between the ray and the object. These factors suggest that for the EM computations, an FPGA is a more suitable platform than a GPU. We implemented the ray-tracing forward projection and backward projection on a Convey HC-1ex multi-FPGA platform. By contrast, the stencil computation kernel for the TV regularization portion of the algorithm has attributes that make it well suited for a GPU. We also use CPUs for task preparation and scheduling. The main features of the implementation described here include the following.

- A hybrid architecture that combines multiple FPGAs, a GPU and a CPU.
- A shared hardware module that can support both forward projection and backward projection.
- Separation of the machine configuration and the tracing engine.
- Better performance in terms of latency or throughput than a pure GPU implementation on Tesla or Fermi.
- A mapping of high-level algorithmic specification in C into FPGA using the Xilinx AutoESL high-level-synthesis tool.

This paper introduces a new compressive sensing EM+TV algorithm for CT application. The related design and system implementation for this compressive sensing application are also provided. The limitations and considerations for compressive sensing applications are analyzed in detail. The related system design methodologies can be extended to other compressive sensing applications. The methodologies are the design flows, including the algorithm analysis, fixed point conversion, memory behavior analysis, computation kernel evaluation, parallel conflicts solution, data prefetching design, etc.

The remainder of this paper is organized as follows. Section II introduces the mathematical EM+TV algorithm. Section III describes the computation analysis and the computation kernels. Architectural design decisions and parallelism are discussed in Section IV. Details regarding Implementation and optimizations are given in Section V. Section VI contains experimental results, and conclusions are contained in Section VII. An extended abstract of this work was presented in [16].

II. EM+TV Algorithm

The EM+TV algorithm, like many iterative algorithms, is based on solving a system of linear equations

$$Ax = b$$

where $x = (x_1, \cdots, x_N)^T \in \mathbf{R}^N$ is the original image represented as a vector, $b = (b_1, \cdots, b_M)^T \in \mathbf{R}^M$ is the measurement, and A is a $M \times N$ matrix describing the mapping from the original image to the measurement. A is the discrete Radon transform [17], with each row describing an integral along one straight line, and all the elements of A are nonnegative.

The expectation maximization (EM) algorithm [18] is an iterative reconstruction algorithm. The noise can be represented in b as Poisson noise. Then, if x is given and A is known, the conditional probability of b is $P(b|Ax) = \prod_i^M (e^{-(Ax)_i}((Ax)_i)^{b_i}/b_i!)$. Given an initial estimate x^0, the EM iteration for $n = 0, \cdots$, is

$$x_j^{n+1} = \frac{\sum_i \left(a_{ij} \left(\frac{b_i}{(Ax^n)_i} \right) \right)}{\sum_i a_{ij}} x_j^n. \tag{1}$$

The summations over i and j are from 1 to M and N, respectively.

The total-variation regularization method was originally proposed by Rudin, Osher, and Fatemi [19] to remove noise in an image while preserving edges. This technique is widely used in image processing and can be expressed in terms of minimizing an energy function of the form $\min_x \int_\Omega |\nabla x| + \alpha \int_\Omega F(Ax, b)$, where x is viewed as a 2-D or 3-D image with spatial domain Ω, A is usually a blurring operator, b is the observed noisy-blurry

image, and $F(Ax,b)$ is a data fidelity term. For example, for Gaussian noise, $F(Ax,b)=\|Ax-b\|_2^2$.

In this paper, we combine the EM algorithm with TV regularization as in [7]. We first briefly described the method in [7]. In the classic EM algorithm, no prior information about the solution is provided. However, if we are given *a priori* knowledge that the solution has homogeneous regions and sharp edges, this information can be applied to reconstruct an image with both minimal total-variation and maximal probability. Under this framework, the problem becomes

$$\begin{cases} \underset{x}{\text{minimize}} & E(x):=\beta\int_{\Omega}|\nabla x|+\sum_i\left((Ax)_i-b_i\log(Ax)_i\right) \\ \text{subject to} & x_j\geq 0, \quad j=1,\cdots,N \end{cases} \tag{2}$$

where $\beta>0$ is a parameter for balancing the two terms: TV and EM. This is a convex constraint problem, and the optimal solution can be found by solving the Karush–Kuhn–Tucker (KKT) conditions [20]

$$-\beta\text{div}\left(\frac{\nabla x}{|\nabla x|}\right)_j+\sum_i\left(a_{ij}\left(1-\frac{b_i}{(Ax)_i}\right)\right)-y_j=0$$

$$j=1,\cdots,N, \quad y_j\geq 0, \quad x_j\geq 0, j=1,\cdots,N, \quad y^Tx=0.$$

Using the positivity of $\{x_j\},\{y_j\}$ and the complementary slackness condition $y^Tx=0$ gives $x_jy_j=0$ for all $j=1,\cdots,N$. Multiplying by x_j gives

$$-\beta\frac{x_j}{\sum_i a_{ij}}\text{div}\left(\frac{\nabla x}{|\nabla x|}\right)_j+x_j-\frac{\sum_i\left(a_{ij}\left(\frac{b_i}{(Ax)_i}\right)\right)}{\sum_i a_{ij}}x_j=0$$

$$j=1,\cdots,N.$$

The last term on the left-hand side is an EM step (1), which can be denoted by x_j^{EM}, giving

$$-\beta\frac{x_j}{\sum_i a_{ij}}\text{div}\left(\frac{\nabla x}{|\nabla x|}\right)_j+x_j-x_j^{EM}=0, \qquad j=1,\cdots,N.$$

To solve the above equation in x, with x_j^{EM} identified as noted above, we use a semi-implicit iterative scheme for several steps, alternated with a EM step. The algorithm is shown below (convergence was shown in [21]).

Algorithm 1: EM+TV algorithm.

Input: $x^0=1$;

for $Out=1:IterMax$ **do** /* `IterMax: number of outer iterations` */

$x^{0,0}=x^{Out-1}$;

for $k=1:1:K$ **do** /* `K: number of EMupdates` */

$x^{k,0}=EM(x^{k-1,0})$; /* `Including one` Ax `and one` A^Ty */

end

$x^{Out}=TV(x^{K,0})$;

end

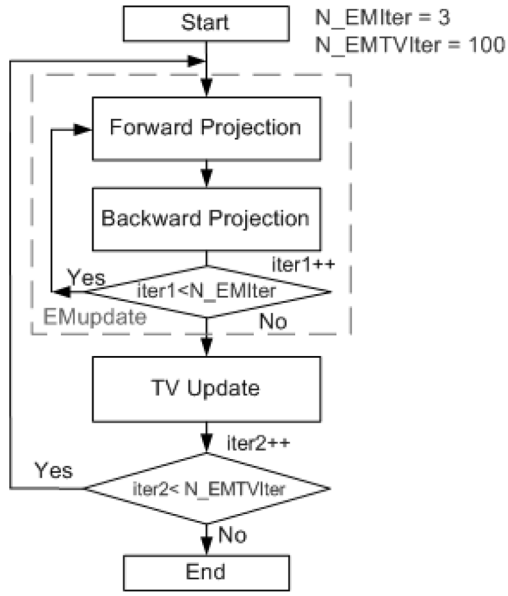


Fig. 2. EM+TV block diagram.

III. Computation Analysis

A. Algorithm Overview

To efficiently accelerate this compressive sensing algorithm, careful analysis of the computation is required. Fig. 2 shows a high-level flow chart for the EM+TV algorithm as implemented. It contains two updating modules: EMupdate and TVupdate. Since there is logic data dependency between EMupdate and TVupdate, the parallelism is found principally in the internal processing of each module. *EMupdate* is more critical for overall efficiency because it occurs in the inner-most loop.

Inside the *EMupdate* kernel, as illustrated in (1), a forward projection is performed to obtain Ax^n, followed by an element-wise division to get $y_i=b_i/(Ax^n)_i$. Backward projection is then performed to obtain A^Ty or $(\sum_{i=1}^M(a_{ij}y_i))$, and then the updated value x_j^{n+1} is obtained using element-wise scaling.

Because the matrix A is very large and sparse, A is never constructed explicitly. A ray-tracing based technique is used to compute the forward projection and backward projection. The EM+TV algorithm is very computationally intensive because it needs to invoke forward and backward projection repeatedly (on the order of 100×3 times in Fig. 2). By contrast, the conventional FDK algorithm only has a single backward projection. Scaling, which is also an important element of the overall computation, is included in the projection step in our implementation. Since *EMupdate* occupies the majority (93%) of the computation time, it is mapped to FPGAs for acceleration. For TVupdate, since there is no data dependency within one TV computation iteration, so it can be easily mapped to a GPU multicore architecture, giving significant acceleration.

B. Ray Tracer Engine

In this section, we focus on EMupdate. The EM algorithm is often implemented with a ray-driven forward projection and a voxel-driven back projection. To facilitate hardware resource

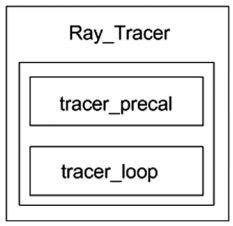


Fig. 3. Ray tracer block diagram.

sharing, we use a ray driven approach in both forward and backward projections. As a result, the forward and backward projection in EM+TV will have the same iterative hierarchical structure. The code for the forward and backward projection is shown in Fig. 4. The first level of iteration comprises the number of views (sources of the ray). The other two layers consist of the array of 2-D detectors/sensors. The ray tracer engine works on one source/detector pair and is the computation kernel for both forward and backward projections.

As illustrated in Fig. 3, the ray tracer is composed of two parts: tracer_precal and tracer_loop. For forward projection and backward projection, the tracer has a similar computational structure. The tracer_precal part operates in the same manner. The only difference is in tracer_loop. In forward projection, tracer_loop will read pixels along with the ray and output one sinogram value for each ray; while in backward projection, the tracer_loop will read and update pixels along each ray. The code first identifies the direction for the next voxel in the ray and then it performs a MADD operation to accumulate the sinogram or update the image. Note that $\lambda - \lambda_0$ provides the coefficients for the matrix A. The forward projection attempts to compute a line integral, while the backward projection attempts to distribute a line integral onto the points on the ray. The tracing stops if the voxel hits the boundary of the object.

C. Intersection Computation

The $tracer\_precal()$ function is responsible for computing the intersection point of the ray with the object and identifying the parameter required for the tracing. Given a source coordinate (s_x, s_y, s_z) and destination (d_x, d_y, d_z), the procedure finds out the intersection point with the object, which is a cube $0 \leq x < N_x, 0 \leq y < N_y, 0 \leq z < N_z$. The procedure first needs to identify the intersection ratio in each dimension

$$\lambda_{xmin} = min\left(\frac{0 - s_x}{d_x - s_x}, \frac{N_x - 1 - s_x}{d_x - s_x}\right). \tag{3}$$

This computes the x-dimension intersection ratio that is closer to the source. Similarly

$$\lambda_{xmax} = max\left(\frac{0 - s_x}{d_x - s_x}, \frac{N_x - 1 - s_x}{d_x - s_x}\right). \tag{4}$$

The procedure then finds out the min and the max of the ratios

$$\lambda_{min} = max(\lambda_{xmin}, \lambda_{ymin}, \lambda_{zmin}) \tag{5}$$

$$\lambda_{max} = min(\lambda_{xmax}, \lambda_{ymax}, \lambda_{zmax}). \tag{6}$$

The ray intersects with the object if and only if $\lambda_{min} < \lambda_{max}$. Once it has been established that the ray intersects with the object, we then compute the near-end integer intersection coordinate (v_x, v_y, v_z) using λ_{min}. Other parameters $\lambda_x, \lambda_y, \lambda_z, \lambda_0$, used in the tracing loop can be derived based on the coordinate (v_x, v_y, v_z). A number of divisions are used in the procedure.

IV. Overview of the Design

The design proposed in this paper is based on the combination of Xilinx FPGA and Nvidia GPU. As illustrated in Fig. 5, the EMupdate portion of the algorithm is accelerated on the FPGA and the TVupdate portion is accelerated using the GPU.

A. Convey HC-1(ex) Platform and GPU Accelerator

The reconstruction algorithms considered here are generally memory bound. The multi-FPGA platform Convey HC-1(ex) from Convey Computer Corporation was selected as the hardware platform due to its high external memory bandwidth and excellent support for random data access. It uses an interleaved memory scheme in which different FPGAs access the off-chip memory using a shared memory model. The system employs an on-board crossbar to realize the interconnection. Fig. 6 shows the system diagram for a Convey HC-1, which has Virtex5 LX330 FPGAs. In the HC-1(ex) version of the platform, Virtex6 LX760 FPGAs are used. The system supports two modes of interleaving schemes. In prime number interleave, the system uses a prime number of memory banks to better support power-of-two strides.

The Convey system has a total of 16 dual in-line memory modules (DIMMs). As shown in Fig. 6, each memory controller is connected to two DIMMs. The HC-1(ex) platform has four user FPGAs. Each FPGA has eight physical memory ports connected to eight memory controllers which run at 300 MHZ. The core design runs at 150 MHZ. Thus, effectively each FPGA is connected to 16 memory access ports through time multiplexing. The peak off-chip memory bandwidth is 80 GB/s if each channel supplies one 64-bit data every cycle.

The Nvidia Tesla C1060 is connected with Convey HC1-ex platform. The Tesla C1060 is built on a 55 nm process and utilizes 240 CUDA Cores (Shaders). The Graphics Clock operates at 1.3 GHz. You will find 4 GB GDDR3 of memory on board, running on a 512-bit memory bus at 1.6 GHz. This will provide a maximum 102 GB/s of memory bandwidth.

B. Ray-by-Ray Parallelism Versus Voxel-by-Voxel Parallelism

As noted earlier, in the forward projection step, it is necessary to read the voxel values along the ray, and update (accumulate) the corresponding sinogram value based on voxel values. In backward projection, the voxel values on the ray are updated based on the sinogram value associated with the ray. The code shown in Fig. 4 is consistent with a ray-by-ray tracing approach.

However, there are two approaches to parallelize the ray-tracing forward/backward projection. One is a ray-by-ray approach in the manner of Fig. 4, while the other is a voxel-by-voxel approach. For the forward projection, a ray-by-ray approach is preferred because the accumulation of signogram data for each ray is independent, and the need for concurrent updates on the (shared) sinogram data can be avoided. For the backward projection, the voxel-by-voxel approach avoids access conflict. However, since the forward and backward projec-

```
//EMupdate : ray-tracing algorithm
for all the views
for all the detectors
{
  tracer_precal(); // find initial ray parameters
  //  λx ,λy ,λz ,λ0 , vx ,vy ,vz ,
  // Lenx ,Leny ,Lenz , signx , signy , signz
  if(mode==0) tempsino=0; //forward projection
  else  value= sinogram(..); //backward projection
  for (i=0;i<Nx+Ny+Nz;i++)//(tracer_loop)
     {
     if  (λx <= λy && λx <= λz) λ=λx;
     else if (λy <= λz)         λ=λy;
     else                       λ=λz;
      //Multiply accumulate (MAC) computation
     if(mode==0)              // forward projection
      tempsino+=imageData(vx,vy,vz)*(λ-λ0);
     else                     // backward projection
      imageData(vx,vy,vz)+=value*(λ-λ0);
     λ0=λ;
      //Find the next point on the ray
     if  (λx <= λy && λx <= λz){λx+=Lenx;vx+=signx;}
     else if (λy <= λz)        {λy+=Leny;vy+=signy;}
     else                      {λz+=Lenz;vz+=signz;}
    // Exit conditions
     if(vx<0||vx>Nx-1) break;
     if(vy<0||vy>Ny-1) break;
     if(vz<0||vz>Nz-1) break;
     }
    if(mode==0) sinogram(..)=tempsino;
}
```

Fig. 4. Ray tracing core engine.

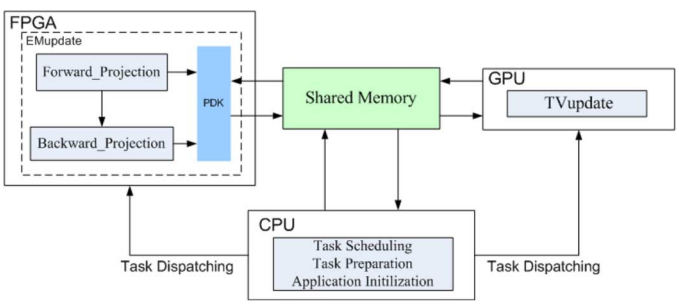


Fig. 5. Proposed hybrid system for EM+TV 3-D.

tion share many similar features, we use the ray-by-ray approach to enable the sharing of the hardware.

Using a ray-by-ray approach also enables the tracing engine more independence from the machine configuration. There are various source/detector configurations in CT, such as fan-beam, cone-beam, parallel-beam, etc. If a voxel-based approach is used, the list of sinograms that contribute to a voxel is highly dependent on the machine configuration. By contrast, in a ray-tracing approach that realizes ray-by-ray based parallelism, once the set of rays are known, the hardware for tracing can be reused. Using this architecture, it is much easier to migrate from one machine setup (e.g., cone-beam) to another (e.g., fan-beam). The procedure to cope with access conflicts for backward projection in a ray-by-ray mode will be described in Section IV-D.

C. No Cache Interleaved Access

Ray tracing involves a significant amount of random data access. Those accesses present certain degree of reuse, however,

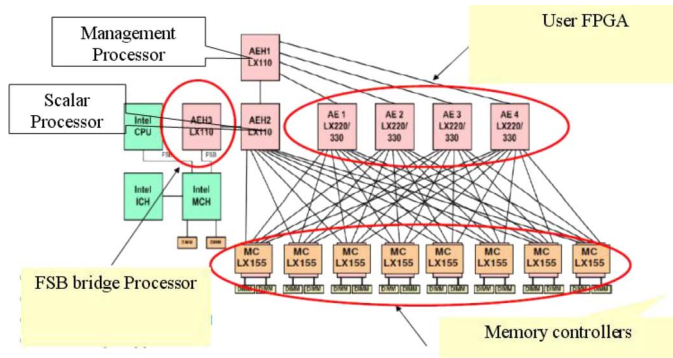


Fig. 6. System diagram of the Convey HC-1(ex) hybrid computer.

the reuses are hard to capture in the absence of a cache-based system. Note it is also possible to use a block RAM (BRAM) scratchpad to capture reuse within the application design [15]. However, that requires deep knowledge of the specific geometry of rays and how they intersect, and thus changes for different images and configurations. Based on these considerations the implementation in the present paper does not use caching.

Most existing FPGA computing boards prefer burst access. In Li *et al.* [9], how to obtain good memory bandwidth on a FPGA-based system that uses burst transfers is introduced. In the convey HC-1(ex) system, parallel data access is not done though a burst approach, but rather through interleaving. Requests from different channels can be processed in parallel if they fall into different banks. The system has 16 DIMMs and 1024 banks in total in the memory system. So the possibility of the bank conflict is low if the parallel accesses are quite random. Such an interleaved memory design is also seen in the on-chip scratchpad memory of Nvidia GPUs. Because of the bandwidth of the external memory is already quite high, we do not implement cache but talk to memory channels directly.

D. Resolving Access Conflicts in Parallel Backward Tracing

The forward projection can be easily parallelized by performing simultaneous computations for different source and detector pairs. For backward projection, however, there are dependencies among views. Moreover, even within one view there can be conflicts when two parallel units update one pixel. To resolve the data conflicts within one view, atomic functions that guarantee the mutual exclusion of an address in memory can be used. This approach has already been used to accelerate the backward projection in a GPU environment [14]. However, an FPGA platform does not provide atomic operations on the memory system.

To address this, we ensure that the computations for different views (sources) are done sequentially. For the same view, the detectors that are far enough apart are associated with one group. This ensures that there will be no conflicts within the group and that all tracers in the group can be processed in parallel. As illustrated in Fig. 7, tracer lines having the same pattern can be chosen. The selection of the distance between two adjoint detectors involves a tradeoff between parallelism granularity and algorithm performance. In our implementation, we choose the distance to be 5. The relationship between different distance choice and the final reconstructed image quality is shown in Fig. 8. The

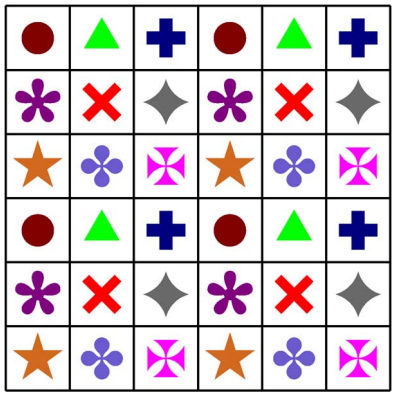


Fig. 7. Ray-based parallel mapping.

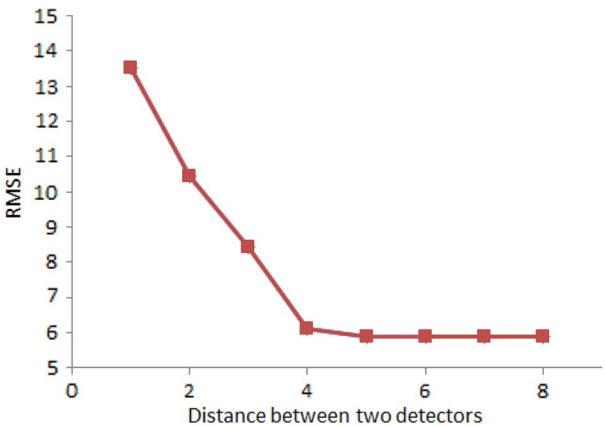


Fig. 8. RMSE performance with different parallel intervals.

figure shows the RMSE for the results with different intervals. As shown in the figure, when the interval is $5 \sim 8$, the algorithm without atomic operation can obtain the same RMSE result as with atomic operation.

E. Memory Bandwidth Calculation

Generally, most CT reconstruction applications are memory bound. For the EM+TV 3-D algorithm, there is data dependency between the projections in the EMupdate step. Therefore, a global data synchronization is required after forward projection and backward projection in each iteration. For each iteration on data sets consistent in size with medical data ($512\times512\times256$ with 4 bytes data type size), the sino data and image data synchronization will require about 350 MB ($(512 \times 512 \times 256 + 736\times64\times500)^*4$) of data communication. When, as is typically the case, hundreds of iterations are needed, the overall bandwidth requirement will be enormous. This places significant constraints on the solution architecture. For example, simply using a platform with many computational cores will not be sufficient.

To explore this experimentally, we implemented the EM+TV 3-D application on a server with 24 cores with multithread techniques support. As illustrated in Fig. 9, when the core number increases, the speedup ratio does not increase linearly. Furthermore, since the bottleneck is memory bandwidth, a maximum speedup will be reached when the memory bandwidth on one

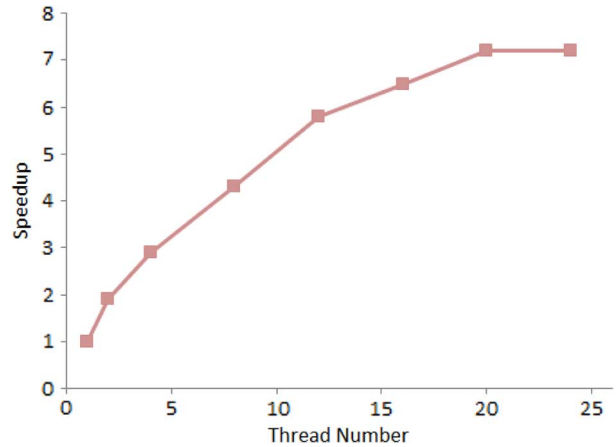


Fig. 9. EMTV speedup on a multicore cluster.

server is consumed, and the performance can not be further improved through the addition of more CPU cores. We also explored a cloud-based solution using Message Passing Interface (MPI) framework on hundreds of cores. In this environment, the data communication, which is performed through a network and involves high synchronization overhead, once again becomes a critical factor limiting performance.

V. Implementation and Optimization

In this section, the implementation details of this compressive sensing application are provided. And the optimization methods used to improve the performance are also introduced.

A. Fixed Point Conversion

In order to realize an efficient FPGA implementation and obtain the maximum speedup, fixed-point operations are required. The associated quantization errors must be carefully balanced against the requirements of the application. We use a range analysis technique to obtain the range of all the values in our datapath. Because the algorithm is iterative, for each iteration, the errors caused by truncation on precision can accumulate. To address this, we used a dynamic precision analysis method to determine the number of fractional bits needed.

In the EM+TV algorithm, the computation kernel ray tracer is the most precision sensitive aspect of the algorithm. To explore this, we use an original phantom image as a reference, and examine the mean square error (MSE) between the original phantom and the reconstructed image as a function of precision. As illustrated in Fig. 10, the precision has a significant impact on the reconstructed image quality. When 18 bits (corresponding to a precision of about 10^{-5}) are used, a fixed point implementation can achieve the same reconstruction quality as a floating point implementation. To provide an extra margin of accuracy, we used 20 fractional bits. Given the many multiplications and divisions in the operations, to preserve the precision of 10^{-5}, 64-bit arithmetic is used for the intermediate core operations.

B. Streaming Architecture

Function $tracer\_precal$ and the $tracer\_loop$, as illustrated in Fig. 3, can be executed in a task-level pipeline. We synthesize

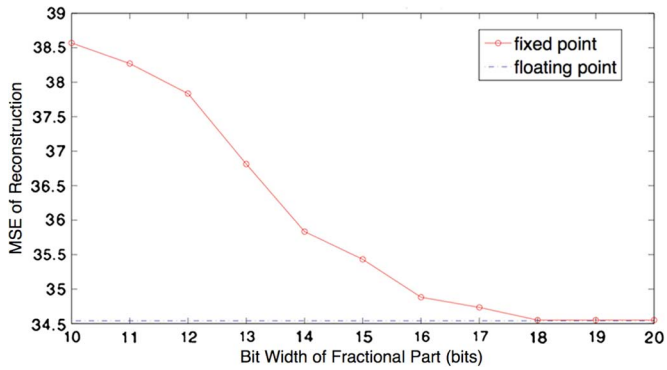


Fig. 10. Fractional bit width and reconstruction quality.

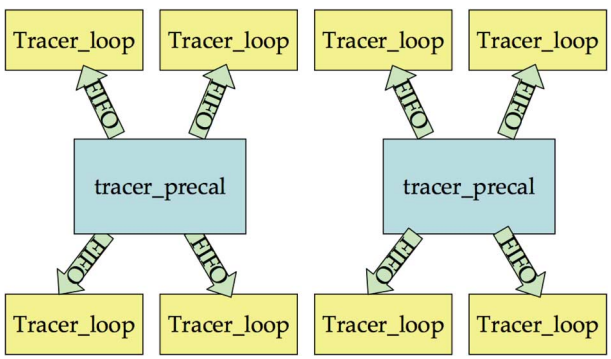


Fig. 11. Overall streaming architecture in one FPGA AE.

the $tracer\_precal$ and the $tracer\_loop$ individually to obtain their corresponding latency reports. Because the loop bound of the $tracer\_loop$ is not known, a simulation of test data is used to compute the average-case latency of the $tracer\_loop$. The throughput of the memory interfaces is also considered. The latency of the $tracer\_precal$ is approximately 1/4 of the latency of the $tracer\_loop$ for a data set of size $512 \times 512 \times 256$. We implement two $tracer\_precal$ modules and eight $tracer\_loop$ module in a single FPGA. As noted previously, each FPGA has 16 virtual memory channels. Each $tracer\_loop$ module exchanges data with two of them (one for read and one for write). The multi-FPGA system has four user FPGAs (Application Engine or AE), and the workload distributed in a SIMD manner.

EM implementation in one FPGA is shown in Fig. 11. To realize such a diagram in C, we invoke the function $tracer\_precal$ twice and invoke the function of the tracer loop eight times. These different invocations take different FIFO channels and memory interfaces as parameters, and the compiler parallelizes function calls that are independent. The round robin distribution logic is coded in the $tracer\_precal$ function. At the receiver side of $tracer\_loop$, the control is just a simple counter to maintain the number of rays processed. Each $tracer\_loop$ processes a predetermined number of rays. In the case where a ray does not intersect with the object, the $tracer\_precal$ sends an appropriate flag to denote that no processing is needed, and the counter is updated to maintain a correct exit condition. The controls that identify the list of sources and detectors are also coded in the function, along with the lookup tables ROM for $sin\ cos$ functions. This framework makes it easy to change these controls to migrate this code for use with data from a machine with a different scanning setup.

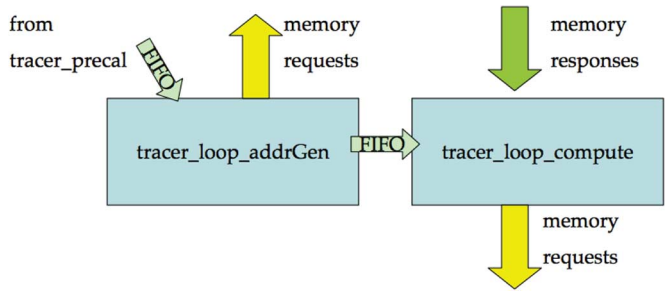


Fig. 12. Streaming architecture inside one $Tracer\_loop$ kernel.

```
if (mode==0) // forward projection
        tempsino+ = imageData(v_x, v_y, v_z) * (λ − λ_0);
else // backward projection
     if (image_denote(v_x, v_y, v_z)==1)
        imageData(v_x, v_y, v_z)+ = value * (λ − λ_0);
```

Fig. 13. Masking for backward projection.

C. Prefetching

Off-chip memory access has a long latency. For example, on the Convey platform, the latency is 125 cycles at 150 MHz. And, the latency can be even longer if congestion (bank conflicts) occur. Given the large amount of random access in EM+TV, prefetching is critical to overall system performance. To address this, we model each memory access port with a request FIFO and a response FIFO. As shown in Fig. 12, it is necessary to invoke two parallel functions inside the hierarchy of $tracer\_loop$. One function is the "helper thread" $tracer\_loop\_addrGen$, which is responsible for sending memory requests for reads, and the other function is the "compute thread" $tracer\_loop\_compute$, which obtains data from the response FIFO and write out the computed result into another request FIFO. This way, the helper threads can keep sending as many requests as possible, until the FIFO is full. Thus, the helper thread in essence is performing the function of prefetching the required data, and the response FIFO serves as the prefetch buffer. Fig. 12 depicts the architecture inside the $Tracer\_loop$ function.

D. Reducing the Data Accesses via Sparsity

Based on knowledge that the output image is sparse and that voxel data is nonnegative, we develop a simple heuristic to reduce the amount of data access. In the beginning of an iteration, a single forward projection is performed. If any accumulated sinogram value falls below a threshold, we conclude that any image value on that ray is likely to be close to zero. Based on this, a 1-bit mask of the image called $image\_denote$ is constructed. When the backward projection is performed, it is only necessary to update the voxels that are not masked, thereby reducing the number of data access in the backward projection. Fig. 13 shows the modified pseudo code.

TABLE I
PERFORMANCE AND ENERGY NUMBERS FOR EMUPDATE COMPUTING KERNELS ($512 \times 512 \times 256$)

	Power	Forward Projection		Backward Projection		Forward+Backward	
		Latency/Throughput(s)	Energy(J)	Latency/Throughput(s)	Energy(J)	Latency/Throughput(s)	Energy(J)
CPU	372W	20.70	724.26	34.45	1205.3	55.15	1929.6
FPGA	1002W	3.25	306.13	3.29	309.3	6.54	615.5
GPU	2134W	3.65	729.6	7.13	1425.1	114.91	2154.7

E. Simultaneous Reconstruction of Two Images

After fixed point conversion, the external data accesses are all 32-bits wide. However, the memory interface of the multi-FPGA platform supports 64-bit memory interface. Because the data access in the tracing is random, it is not feasible to use the 64-bit interface to directly enlarge the application bandwidth. However, if there are two images to be reconstructed from data acquired using the same machine setup, then the reconstruction can be performed simultaneously by packing two 32-bit data words from different images into each 64-bit word, and the $tracer\_precal$ does not need to be modified.

F. GPU Accelerated TVupdate

In this EM+TV algorithm, TVupdate has three layers of iterations. The first layer comprises the number of views (sources of the ray). The other two layers consist of the array of 2-D detectors/sensors. There is no data dependency between the iterations, and a fully parallel implementation can be used with a GPU. The Nvidia GPU has three layers of procesing units, which are called thread, block and grid layers, respectively. Given the natural mapping of three layer parallelism, TVupdate can be implemented very efficiently on the GPU platform.

VI. EXPERIMENTAL RESULTS

The operations destined for hardware implementation are described in C and synthesized into verilog RTL using AutoESL HLS tool ver. 2011.1. The software operations is implemented with Compute Unified Device Architecture (CUDA) Toolkit 3.2 for parallel computing on a CUDA-enabled NVIDIA GPU. The target application is a Cone-Beam CT system. An image of size $512\times512\times256$ is tested. It has 500 views (sources) and 736×64 detector vectors. We parallelize the CPU code using p-thread and implement the GPU kernel using CUDA.

A. Kernel Performance and Energy Consumption

Table I presents the performance and the energy consumption of the forward projection kernel and the backward projection kernel. The values in the table are obtained by averaging 1000 invocations. The performance on a dual-core CPU and many-core GPU is also reported. The CPU used is Intel Xeon 5138 with 2.13 GHz clock frequency and 35 W TDP. The GPU column denotes Nvidia Tesla C1060 with 240 cores and 200 W TDP.

From Table I, the throughput of the FPGA design is the highest. The power of the FPGA application engine is measured using the Xilinx xPower tool. When the latency of forward and backward projection is combined, the multi-FPGA engine is about 50% faster than the CUDA implementation on Tesla C1060. Since it is possible as described above to perform two reconstructions simultaneously, the FPGA-engine can be three times faster than a Tesla C1060. As shown in the table, the FPGA platform is advantageous from an energy standpoint as well.

TABLE II
FPGA AREA CONSUMPTION OF EMUPDATE

	BRAM	DSP	LUT	FF	Slice
Consumed	79	68	113,355	104,099	36511
Total Available	720	864	474,240	948,480	118,560
Utilization	11%	7%	23%	10%	30%

It is also notable that the execution time for backward projection is noticeably slower on other platforms. This is because the amount of data access is up to two times larger on these platforms (due to the need to read the voxel value and then write it back). Also, more invocations (and synchronization) are needed to avoid the conflicts and ensure correctness, which reduces the available parallelism. In the FPGA design, by contrast, the same architecture is used for both forward and backward projection. Each processing element (PE) is connected to two memory channels, one for read and one for write. Thus, the execution times of forward projection and backward projection are similar.

The hardware area consumption for the complete EMupdate FPGA design is listed in Table II. The core computing RTL consumes fewer logic slices, because the Convey's Personality Development Kit (PDK) infrastructure also consumes about 10%–15% area. Most of the BRAM utilization is due to the PDK infrastructure. It should be emphasized that since the computation kernels are independent of the size of the image data, the designed kernel can work for different machine setups. The area of the EMupdate will keep almost same for different data.

B. Application Performance and Energy Consumption

The EM+TV 3-D CT application has been tested on the proposed hybrid system. The EM portion is done by the FPGA-subsystem and the TV portion is done by the GPU. The flowchart of the application is shown in Fig. 2, where the outer EM+TV iterates 100 times, and the inner EM step iterates three times for each EM+TV iteration. The hybrid configuration connects the Tesla C1060 onto the Convey HC-1(ex) platform. After one EM iteration completes, the image data is copied into the GPU memory space and the TV CUDA kernel starts; the data transfer does not add substantial overhead in this case. We experimentally confirmed that the pipelined data transfer (FPGA coprocessor-side memory to PCI-e) can achieve close to 1 GB/s, while each EM iteration only needs to copy 256 MB image data to GPU. And similarly we need to do the transfer backwards when one TV invocation finishes. The data communication only adds

TABLE III
APPLICATION PERFORMANCE AND ENERGY CONSUMPTION

	Throughput(s)	Energy(J)
CPU	12682	443.7E3
Hybrid	981.0	135.5E3
GPU	3850.7	770.1E3

a modest amount of extra time for each EM+TV iteration. Because the TV kernel is a highly regular stencil computation, about a 10× speed up can be achieved with GPU. The execution time of the TV is much shorter than EM (no more than 5% of the overall time consumption). The performance of the proposed hybrid architecture is compared with that of the CPU only or GPU only architectures, as illustrated in Table III. The proposed architecture has the highest performance and minimum energy consumption.

Compared with the GPU/CPU only [14] or FPGA only [15] implementations, the proposed hybrid architecture takes advantages of the FPGA and GPU at the same time. Since the area consuming parts, such as TVupdate, has been efficiently accelerated with GPU, the area consumption is lower than that of the FPGA only implementations. For the power consumption, with the usage of FPGA, the power consumption of the proposed architecture is less than 1/3 of the CPU implementation and 1/5 of the GPU implementation.

VII. CONCLUSION

We have proposed a new architecture for implementing compressive sensing reconstruction. This system is based on a hybrid involving both FPGA and GPU computations, and has significant performance advantages over a GPU-only or cloud-based multi-server approach. The advantages involve not only speed, but also energy consumption. While we have explored this in the context of EM+TV, we believe that there is a broader opportunity to apply hybrid computing approaches to a wide variety of compressive sensing processing. Solutions such as the approach presented here reduce the computation time associated with compressive sensing, and thus make it more practical to perform medical imaging at lower radiation exposure levels.

REFERENCES

[1] S. Coric, M. Leeser, E. Miller, and M. Trepanier, "Parallel-beam backprojection: An FPGA implementation optimized for medical imaging," in *Proc. Int. Symp. Field-Programmable Gate Arrays*, 2002, pp. 217–226.

[2] L. Shepp and B. Logan, "The Fourier reconstruction of a head section," *IEEE Trans. Nucl. Sci.*, vol. 21, pp. 21–34, 1974.

[3] X. Pan, E. Sidky, and M. Vannier, "Why do commercial CT scanners still employ traditional filtered back-projection for image reconstruction?," *Inverse Problems*, vol. 25, p. 123009, 2009.

[4] A. Kak and M. Slaney, *Principles of Computerized Tomographic Imaging*. Philadelphia, PA: SIAM, 2001.

[5] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *J. R. Stat. Soc.*, vol. 39, no. 1, pp. 1–38, 1977.

[6] A. H. Andersen and A. C. Kak, "Simultaneous algebraic reconstruction technique (SART): A superior implementation of the ART algorithm," *Ultrason. Imag.*, vol. 6, pp. 81–94, Jan. 1984.

[7] M. Yan and L. A. Vese, "Expectation maximization and total variation-based model for computed tomography reconstruction from undersampled data," in *Proc. SPIE Conf. Med. Imag.: Phys. Med. Imag.*, 2011, vol. 7961, p. 79612x.

[8] N. Gac, S. Mancini, M. Desvignes, and D. Houzet, "High speed 3-D tomography on CPU, GPU, and FPGA," *EURASIP J. Embedded Syst.* vol. 2008, pp. 5:1–5:12, Jan. 2008.

[9] J. Li, C. Papachristou, and R. Shekhar, "An FPGA-based computing platform for real-time 3-D medical imaging and its application to cone-beam CT reconstruction," *J. Imag. Sci. Technol.*, vol. 49, pp. 237–245, 2005.

[10] H. Scherl, B. Keck, M. Kowarschik, and J. Hornegger, "Fast GPU-based CT reconstruction using the common unified device architecture (CUDA)," in *IEEE Nucl. Sci. Symp. Conf. Rec.*, 2007, vol. 6, pp. 4464–4466.

[11] J. Xu, N. Subramanian, A. Alessio, and S. Hauck, "Impulse C vs. VHDL for accelerating tomographic reconstruction," in *Proc. 18th IEEE Annu. Int. Symp. Field-Programmable Custom Comput. Mach. (FCCM)*, May 2010, pp. 171–174.

[12] B. Keck, H. Hofmann, H. Scherl, M. Kowarschik, and J. Hornegger, "GPU-accelerated SART reconstruction using the CUDA programming environment," in *Proc. SPIE*, E. Samei and J. Hsieh, Eds., Lake Buena Vista, 2009, vol. 7258.

[13] F. Xu and K. Mueller, "Accelerating popular tomographic reconstruction algorithms on commodity PC graphics hardware," *IEEE Trans. Nucl. Sci.*, vol. 52, no. 3, pp. 654–663, Jun. 2005.

[14] J. Chen, M. Yan, L. A. Vese, J. Villasenor, A. Bui, and J. Cong, "EM+TV for reconstruction of cone-beam CT with curved detectors using GPU," in *Proc. Int. Meeting Fully Three-Dimensional Image Reconstruct. Radiol. Nucl. Med.*, 2011, pp. 363–366.

[15] D. Stsepankou, K. Kommesser, J. Hesser, and R. Manner, "Real-time 3-D cone beam reconstruction," in *Nucl. Sci. Symp. Conf. Rec.*, Oct. 2004, vol. 6, pp. 3648–3652.

[16] J. Chen, J. Cong, M. Yan, and Y. Zou, "FPGA-accelerated 3-D reconstruction using compressive sensing," in *20th ACM/SIGDA Int. Symp. Field-Programmable Gate Arrays (FPGA 2012)*, Feb. 2012, pp. 163–166.

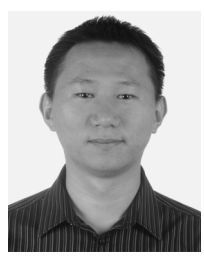
[17] G. Beylkin, "Discrete radon transform," *IEEE Trans. Acoust., Speech Signal Process.*, vol. 35, no. 2, pp. 162–172, Feb. 1987.

[18] L. Shepp and Y. Vardi, "Maximum likelihood reconstruction for emission tomography," *IEEE Trans. Med. Imag.*, vol. 1, no. 2, pp. 113–122, Oct. 1982.

[19] L. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Phys. D.*, vol. 60, pp. 259–268, 1992.

[20] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[21] M. Yan, J. Chen, L. A. Vese, J. D. Villasenor, A. A. T. Bui, and J. Cong, "EM+TV based reconstruction for cone-beam CT with reduced radiation," in *ISVC*, G. Bebis, R. D. Boyle, B. Parvin, D. Koracin, S. Wang, K. Kim, B. Benes, K. Moreland, C. W. Borst, S. DiVerdi, Y.-J. Chiang, and J. Ming, Eds., 2011, vol. 6938, pp. 1–10.



Jianwen Chen (SM'12) received the Ph.D. degree in electrical engineering from Tsinghua University, Beijing, China, in 2007.

From 2007 to 2010, he was a staff researcher in IBM research, where he conducted research on wireless communications systems and multi-core video coding architectures. In September 2010, he joined the Image Communications Lab and the Center of Domain-Specific Computing (CDSC), University of California, Los Angeles (UCLA), where he is currently focusing on the research of high efficiency video coding (HEVC) techniques, high performance computing architecture and applications. His research interests include high performance computing architecture, video signal processing and compression, video communication over challenging networks, cloud computing, and wireless communications system. He has over 40 publications in these areas.

Dr. Chen served as the Chairman of the MPEG Internet Video Codec Ad-hoc Group from February 2012. He was nominated as the Chancellor's Postdoctoral Researcher of UCLA in 2012.

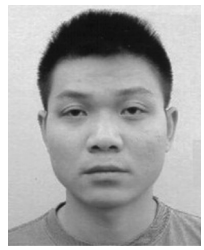
Jason Cong (F'00) received the B.S. degree in computer science from Peking University, Beijing, China, in 1985, and the M.S. and Ph.D. degrees in computer science from the University of Illinois, Urbana-Champaign, in 1987 and 1990, respectively.

Currently, he is a Chancellor's Professor at the Computer Science Department of University of California, Los Angeles (UCLA), the Director of Center for Domain-Specific Computing (CDSC), co-Director of UCLA/Peking University Joint Research Institute in Science and Engineering, and co-Director of the VLSI CAD Laboratory. He also served as the Department Chair from 2005 to 2008. His research interests include synthesis of VLSI circuits and systems, programmable systems, novel computer architectures, nano-systems, and highly scalable algorithms.

Luminita A. Vese received the M.S. degree in mathematics from West University of Timisoara, Romania, in 1993, and the M.S. and Ph.D. degrees in applied mathematics from University of Nice, Sophia, Antipolis, France, in 1992 and 1997, respectively.

She is currently a Professor of Mathematics at the University of California, Los Angeles (UCLA). Before joining the UCLA faculty, she held postdoctoral research and teaching positions at the University of Nice, the University of Paris IX Dauphine, and UCLA. Her research interests include variational methods and partial differential equations, inverse problems, image analysis, and computer vision.

John Villasenor is a Professor of electrical engineering at the University of California, Los Angeles and a nonresident Senior Fellow in Governance Studies and the Center for Technology Innovation at the Brookings Institution. His current research focuses on the technology and policy challenges associated with how information is acquired, processed, and delivered in an efficient and secure manner.



Ming Yan received the B.S. and M.S. degrees in computational mathematics from University of Science and Technology of China, Hefei, China, in 2005 and 2008, respectively. He received the Ph.D. degree from the Department of Mathematics, University of California, Los Angeles in 2012.

He is a postdoctoral fellow in the Department of Computational and Applied Mathematics, Rice University, Houston, TX. His current research interests include variational and optimization methods for image and signal processing.

Yi Zou received the B.E. and M.E. degrees in computer science from Tsinghua University, Beijing, China, in 2004 and 2006, respectively, and the Ph.D. degree in computer science from University of California, Los Angeles, in 2012.

His research interests include medical image processing and high performance computing using FPGAs and GPUs.