EMBIRA: An Accelerator for Model-Based Iterative Reconstruction

Junshi Liu, Swagath Venkataramani, *Student Member, IEEE*, Singanallur V. Venkatakrishnan, *Student Member, IEEE*, Yun Pan, *Member, IEEE*, Charles A. Bouman, *Fellow, IEEE*, and Anand Raghunathan, *Fellow, IEEE*

***Abstract*— Tomographic reconstruction, which involves computing a 3-D volume from its 2-D projections, is an important problem in imaging with wide-ranging applications, including medical scanners, electron microscopy, nondestructive testing, and transportation security. Model-based iterative reconstruction (MBIR) is a popular approach to 3-D reconstruction that has demonstrated the state-of-the-art reconstruction quality on several applications, and has been deployed in commercial healthcare systems. However, software implementations of MBIR on commodity general-purpose processors demonstrate poor performance due to its high compute and data requirements and cache unfriendly data access patterns. In this paper, we develop an efficient MBIR accelerator (EMBIRA) that achieves significant performance and energy improvement over software implementations. EMBIRA utilizes arrays of three types of specialized processing elements that match MBIR's computation patterns, and is further operated as a two-level nested pipeline to fully exploit the parallelism present in the algorithm. Another important source from which EMBIRA derives its efficiency is by constraining the sequence in which voxels[1] in the 3-D volume are reconstructed. This enables better data reuse within the accelerator, thereby significantly reducing the number of off-chip memory accesses. To demonstrate the benefits of EMBIRA, we implemented a prototype on an Altera DE5 field-programmable gate array (FPGA) platform that includes an Altera Stratix V GX FPGA and DDR3 memory. Our implementation of EMBIRA, operating at 165 MHz, achieved 51.8× (5.8×) improvement in performance, and 355× (199×) improvement in energy, compared with optimized sequential (multithreaded) software implementations on a 48-core 2.3-GHz AMD Opteron-based server.**

***Index Terms*— 3-D tomographic reconstruction, energy efficiency, hardware accelerator, model-based iterative reconstruction (MBIR).**

Manuscript received March 26, 2015; revised July 21, 2015 and October 1, 2015; accepted October 6, 2015. Date of publication April 29, 2016; date of current version October 21, 2016. This work was supported in part by the National Science Foundation under Grant CNS-1423290 and in part by High-Performance Imaging, Inc.

J. Liu was with Purdue University, West Lafayette, IN 47906 USA. He is now with the Institute of VLSI Design, Zhejiang University, Hangzhou 310027, China (e-mail: stoneljsh@zju.edu.cn).

S. Venkataramani, C. A. Bouman, and A. Raghunathan are with the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47906 USA (e-mail: venkata0@purdue.edu; bouman@purdue.edu; raghunathan@purdue.edu).

S. V. Venkatakrishnan is with the Lawrence Berkeley National Laboratory, Berkeley, CA 94720 USA (e-mail: svenkata@purdue.edu).

Y. Pan is with the Institute of VLSI Design, Zhejiang University, Hangzhou 310027, China (e-mail: panyun@zju.edu.cn).

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/TVLSI.2016.2551204

[1]Each location in the 3-D volume represents a voxel. It is analogous to a pixel in a 2-D image.

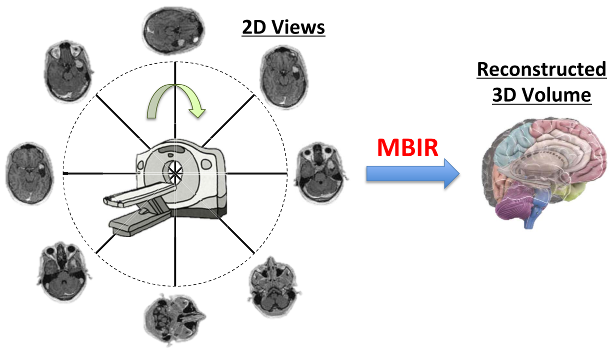


Fig. 1. 3-D reconstruction example: CT.

I. Introduction

Model-based image processing refers to a suite of recently developed algorithms and techniques that address a class of important problems, called inverse problems, in imaging systems [1]. One of the popular inverse problems is 3-D tomographic reconstruction, in which the objective is to piece together a 3-D volume (a physical object or a scene) from multiple 2-D observations acquired using an imaging system. The 3-D reconstruction finds applicability in a wide spectrum of domains, ranging from medical imaging scanners and explosive detection systems to electron and X-ray microscopy [2]–[9]. For example, Fig. 1 shows a computed tomography (CT) scan application, in which X-ray radiation is passed from several angles to record 2-D radiographic images (or virtual projections) of specific areas of the scanned object. These radiographic images are then composed using a reconstruction algorithm to form a 3-D image of the object, which is subsequently used for medical diagnosis. Other examples include forming 3-D object models in baggage scanners, mapping terrains and surfaces in 3-D, and studying objects at the nanometer scale, among many more.

Model-based iterative reconstruction (MBIR) is a promising approach to realize 3-D reconstruction [5], [6], [10]–[12]. The MBIR framework formulates the problem of 3-D reconstruction as the minimization of a high-dimensional cost function, in which each voxel in the 3-D volume is a variable. While several variants of MBIR exist based on how the cost function is minimized [10]–[12], we consider a popular variant, called iterative coordinate descent MBIR (ICD-MBIR), that has demonstrated the state-of-the-art performance on several tomographic applications [6], [13], and has also been commercially deployed in healthcare systems [14], [15]. For brevity, we refer to the ICD-MBIR algorithm as MBIR in the rest of this paper. Given a set of 2-D image observations, the MBIR algorithm reconstructs the 3-D image by iteratively updating each voxel in the volume. First, the voxels are randomly initialized (or based on simple precomputations), and are then refined in a random sequence. Each voxel is updated, such that an error function that depends on the difference between the input 2-D image observations and the 2-D observations derived from the reconstructed 3-D volume is minimized. The algorithm terminates either upon executing a fixed iteration count or if the error function falls below a prespecified convergence threshold.

A. ICD-MBIR Computational Characteristics

The ICD-MBIR algorithm is highly compute and data intensive, and general-purpose implementations of the algorithm often fail to meet the desired performance. For instance, the application considered in our experiments reconstructs a $512 \times 512 \times 256$ 3-D volume from 47 2-D images of size 512×512 each. In this case, the MBIR algorithm requires 50.33 GOPS (giga operations) to update all voxels in the volume once, which constitutes one full iteration of the algorithm. The algorithm can take tens of iterations to converge depending on the error threshold. Clearly, this represents a significant compute demand. Our software baseline (adopted from [16]) required ~1700 s per iteration on a 2.3-GHz AMD Opteron server with 196-GB memory, which is unacceptable for many practical applications.

In addition, unlike other optimization strategies for the MBIR cost function [17], [18], the ICD-MBIR algorithm is not easily amenable to parallel execution on multicores and many-core accelerators, such as General Purpose Graphics Programming Units for the following reasons. First, there is limited parallelism within the core computations that evaluate the updated value of the voxel. In addition, the per-voxel-update computations themselves are relatively small (in our implementation, the time per-voxel update was ~26 μs). Hence, the overheads of parallelization, such as task startup time, data communication, synchronization, and so on, become significant. In summary, parallelizing computations within each voxel update yield little or no performance improvement.

The bulk of the data parallelism in ICD-MBIR stems from the large number of voxels ($512 \times 512 \times 256$) that need to be updated. In the strict sense, the computations on different voxels are not parallel, as MBIR requires each voxel to be updated sequentially and in a random order. However, the previous studies [6], [7], [19], [20] have shown that convergence is not affected when multiple voxels are updated simultaneously, provided that they are located sufficiently far apart in the 3-D volume. While this enables considerable parallelism, the benefits saturate beyond a small number of parallel voxel updates. For example, in our parallel software implementation (adopted from [16]) on a 48-core server, we observed that speedup decreased beyond 12 parallel voxel updates (or threads). This is because, the memory access pattern for voxel updates is not cache-friendly (each voxel-update computation randomly accesses a small number of elements from different parts of a large array structure). Furthermore, voxels that are concurrently updated do not share any data. Therefore, when parallelized, the ICD-MBIR algorithm quickly becomes memory bandwidth-limited, and the benefits naturally saturate.

The above discussions indicate that the new approaches to improve the efficiency of MBIR are critical.

B. EMBIRA: Key Features and Contributions

In this paper, we design a hardware accelerator to achieve improved efficiency in the execution of the MBIR algorithm. The proposed efficient MBIR accelerator (EMBIRA) is composed of processing elements called voxel evaluation modules (VEMs). Each VEM, in turn, contains three types of specialized processing elements viz., theta computation element, neighborhood processing element (NPE), and voxel-update element (VUE), which together perform all the computations required to update a given voxel. The different processing elements are operated as two-level nested pipeline to exploit the fine-grained pipelined parallelism present in the algorithm. The VEM also employs multiple on-chip memory arrays to enhance data reuse and reduce off-chip data accesses. Furthermore, we leverage optimizations, such as data packing and pipelining data computation with communication, to enhance the overall performance of voxel updates on VEMs. The EMBIRA architecture is organized as an array of VEMs, facilitating multiple voxels to be updated concurrently.

An important source of EMBIRA's efficiency is the significant reduction in external memory accesses achieved by constraining the sequence in which voxels in the 3-D volume are updated. In particular, we impose the following constraints: 1) voxels that are updated in parallel should lie on a straight line and 2) when a voxel is updated, a small volume around the voxel is also updated. These constraints greatly enhance data reuse within the accelerator. This leads to a reduction in the number of off-chip data transfers, lowering the memory bandwidth requirement and improving performance. It is worth noting that we ensure that voxels updated concurrently in EMBIRA lie sufficiently far apart in order to preserve convergence.

In summary, we make the following key contributions.

1) We propose an efficient hardware accelerator, EMBIRA, to realize the MBIR algorithm. EMBIRA features two-levels of nested pipelining and specialized processing elements that are tailored to the computations involved in voxel updates.
2) To improve the processing efficiency of EMBIRA, we constrain the sequence in which the voxels are updated to enforce better data sharing/reuse within the accelerator, thereby significantly reducing the memory bandwidth requirement.
3) We prototype EMBIRA on an Altera DE5 field-programmable gate array (FPGA) platform that includes an Altera Stratix V GX FPGA interfaced with a DDR3 memory. Our implementation of EMBIRA, operating at 165 MHz, demonstrates 51.8× and 5.8× improvement in performance compared with optimized sequential and multithreaded software implementations on a 48-core 2.3-GHz AMD Opteron-based server.

The energy consumption of EMBIRA is 355× (199×) lower than the sequential (multithreaded) implementations.

The rest of this paper is organized as follows. Section II provides a brief description of the MBIR algorithm. Section III details the EMBIRA architecture. Section IV outlines the system-level evaluation setup. Section V presents the experimental methodology, followed by the results in Section VI. Section VII describes the related research efforts and highlights the distinguishing features of this paper, and Section VIII concludes this paper.

II. MBIR: Background and Preliminaries

In this paper, we focus on the problem of 3-D tomographic reconstruction from a set of 2-D images obtained using a parallel beam source and by rotating the source/object about a single axis. Typically, the measurements in such a system are related to the unknown voxels via line integrals (projections). Thus, the goal of a reconstruction algorithm is to obtain the unknown volume from the set of noisy 2-D parallel beam images.

In order to describe the MBIR approach, it is useful to think of all the 2-D images (measurements) as well as the unknown 3-D volume of voxels as vectors. If y is an $M \times 1$ vector containing all the measurements, x is an $N \times 1$ vector containing all the voxels in the 3-D volume, and A is an $M \times N$ matrix implementing the line intergral through the 3-D volume, then the MBIR is obtained by minimizing the function

$$c(x) = \frac{1}{2}||y - Ax||_\Lambda^2 + \sum_{r,s \in \mathcal{N}_s} w_{rs}\rho(x_s - x_r) \qquad (1)$$

where r and s are voxels, $\mathcal{N}$ represents the set of all pairs of neighboring voxels in 3-D (using, say, a 26-point neighborhood system), $\rho(.)$ is a potential function that incorporates a model for the underlying image, Λ is a diagonal matrix whose entries weight each term by a factor inversely proportional to the noise in the measurement, and w_{rs} is a set of normalized weights, depending on the physical distance between neighboring voxels. While there are a variety of options for the $\rho(.)$ function, we choose the q-GGMRF potential [21] function given by

$$\rho(\Delta) = \frac{\left|\frac{\Delta}{\sigma_f}\right|^2}{c + \left|\frac{\Delta}{\sigma_f}\right|^{2-p}} \qquad (2)$$

where c is typically set to 0.01, $1 \leq p \leq 2$, and σ_f is a smoothing parameter. The first term in (1) has the interpretation of enforcing consistency of the desired reconstruction with the measurements while the second term enforces certain desirable characteristics in the reconstruction (sharp edges, low noise, and so on).

Note that each column of the matrix A represents all the projections of a single voxel on the detector. Hence, each column is a sparse vector. Furthermore, since we are dealing with a parallel beam geometry with rotation about a single axis, the A-matrix columns share values across slices in the 3-D volume (see Fig. 2). Hence, the A matrix needs to only be computed for voxels in a single slice and can then be reused across slices.

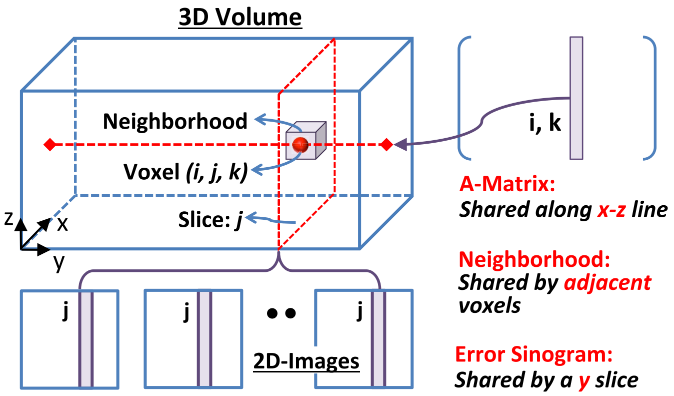


Fig. 2. Access pattern for voxel update in the MBIR algorithm.

In this paper, we will focus on minimizing the above cost using the ICD algorithm combined with the concept of majorization [5], [6]. The basic idea in ICD is to update the voxels one at a time so as to decrease the value of the cost function (1) with each update. The updates result in a monotonic decrease of the cost. Since the cost function is convex and is bounded from below, this method converges to the global minimum.

The cost function in (1) with respect to a single voxel (ignoring constants) indexed by s is given by

$$c_s(z) = \theta_1 z + \frac{\theta_2}{2}(z - x_s)^2 + \sum_{r \in N_s} w_{rs}\rho(z - x_r) \qquad (3)$$

$$\theta_1 = -e^t \Lambda A_{*,s}, \quad \theta_2 = A_{*,s}^t \Lambda A_{s,*} \qquad (4)$$

where $A_{*,s}$ is the sth column of A, $e = y - Ax$, and x_s is the current value of the voxel s.

Due to the complicated nature of the function $\rho()$, it is not possible to find a simple closed form expression for the minimum of (3). Hence, $\rho()$ is often replaced by a quadratic surrogate function, which makes (3) simpler to minimize. In particular, if

$$a_{rs} = \begin{cases} \dfrac{\rho'(x_r - x_s)}{(x_r - x_s)} & x_s \neq x_r \\ \rho''(0) & x_s = x_r \end{cases} \qquad (5)$$

then a overall surrogate function to (3) is given by

$$c_s(z) = \theta_1(z - x_s) + \theta_2(z - x_s)^2 + \sum_{r \in N_s} w_{rs}a_{rs}(z - x_r)^2. \qquad (6)$$

Taking the derivative of this surrogate function and setting it to zero, it can be verified that the minimum of the function is

$$z^* \leftarrow \frac{\theta_2 x_s - \theta_1 + \sum_{r \in N_s} w_{rs}a_{rs}x_r}{\theta_2 + \sum_{r \in N_s} w_{rs}a_{rs}}. \qquad (7)$$

Note that minimizing (6) ensures a decrease in (3), and hence, in the original function (1). The algorithm can be efficiently implemented by keeping track of the fitting error vector e (also called the error sinogram) along with each update [5].

Algorithm 1 Pseudocode of the MBIR Algorithm

```
Input: 2D Measurements: $y$
Output: Reconstructed 3D volume: $x$
1: Initialize $x$ at random
2: Error Sinogram: $e = y - Ax$
3: while Convergence criteria not met do
4:   for each slice $s$ ($y$ co-ord.) do
5:     for each voxel $v$ in $s$ in random order do
6:       $\theta_1$ and $\theta_2 = f(e, A_{*,v})$ (Eqn. 4)
7:       for voxels $u \in$ neighborhood $\mathcal{N}_v$ of $v$ do
8:         Compute surrogate fn. $a_{uv}$ for $u$ (Eqn. 5)
9:       end for
10:      Compute $z^* = g(\theta_1, \theta_2, x_v, a_{*v})$ (Eqn. 7)
11:      Update Error Sinogram: $e \leftarrow e - (z^* - x_v)A_{*,v}$
12:      Update voxel: $x_v \leftarrow z^*$
13:    end for
14:  end for
15: end while
```

The pseudocode of MBIR is summarized in Algorithm 1. Given a set of 2-D measurements (y) as inputs, the algorithm produces the reconstructed 3-D volume (x) as the output. First, the voxels in x are initialized at random (line 1). Next, the error sinogram (e) is computed as the difference between the 2-D measurements and the 2-D views obtained by projecting the current 3-D volume. The algorithm iteratively updates the voxels (lines 4–14) until the convergence criterion is met. In each iteration, every voxel in the volume is updated once in a pseudorandom order. To this end, the 3-D volume is decomposed into multiple slices along the y-axis. Thus, each slice is comprised of a plane of voxels in the x- and z-directions. We adopt the following two-step process to decide the order in which the voxels are updated. First, a slice in the volume is selected (line 4). Next, all voxels within this slice are updated (lines 5–13) by randomly picking their x- and z-coordinates. This process is repeated until all the slices are updated. The two-step process facilitates easy parallelization of the algorithm along the slice loop (line 4). Note that the slices updated in parallel should be located as far apart as possible to ensure convergence.

Lines 6–12 describe the steps involved in updating a voxel. First, the parameters θ_1 and θ_2 are computed using the A matrix and the error sinogram e (line 6). Next, the quadratic surrogate function is evaluated for each of the voxel's neighbors (lines 7–9). These are utilized to compute the new value of the voxel z^* (line 10). The error sinogram (e) and the 3-D volume (x) are then updated (lines 11 and 12). From a computational standpoint, each voxel update involves accessing three key data structures as shown in Fig. 2 and enumerated as follows: 1) a column of the A matrix, which is indexed by the x- and z-coordinates of the voxel (hence, voxels along an xz line share the same A-matrix column); 2) voxel neighborhood, which is unique to each voxel, but shared in part by adjacent voxels along all the directions; and 3) part of the error sinogram, which is determined by the slice ID of the voxel, and is, hence, shared by all voxels in a slice. In EMBIRA, we leverage these data access patterns to optimize computations within a voxel and maximize data reuse across voxels, thereby enhancing processing efficiency.

III. EMBIRA: Hardware Architecture

The EMBIRA architecture, shown in Fig. 3, is designed to efficiently realize the MBIR algorithm by exploiting its computational characteristics. The hardware architecture is scalable, i.e., its microarchitectural parameters can be easily modulated to achieve various levels of performance, power, and reconstruction quality. EMBIRA consists of an array of VEMs, a global control unit, local memory, and an external memory interface. Each VEM can perform all computations required to update a single voxel. The input 2-D images, the reconstructed 3-D volume, and other data structures (A matrix, error sinogram) are assumed to be stored in external memory. The global control unit of EMBIRA contains appropriate control registers to specify the locations of the data structures in external memory and control logic to generate the required interface signals.

At the high level, the operation of EMBIRA can be summarized as follows. First, the global control unit generates a pseudorandom voxel ID (x-, y-, and z-coordinates). Based on the coordinates, the data required to update the voxel viz., the column of the A matrix, a portion of the error sonogram, and the voxel neighbors, are transferred from the external memory to EMBIRA. One of the VEMs then computes the updated values of the voxel and error sinogram. This process may be concurrently performed on multiple voxels to utilize the VEM array. We adopt several techniques to reuse the data in the internal memory across multiple voxel updates, thereby significantly reducing the number of external memory accesses. The updated values of the voxel and the error sinogram are then stored back to the external memory. This process is repeated until the convergence criterion is met. In this section, we describe the architectural features of EMBIRA and the optimizations adopted to improve its processing efficiency.

A. Voxel Evaluation Module and Intravoxel Pipelines

The key computation element of EMBIRA is the VEM. A VEM performs the core computations that evaluate the updated value of a voxel. Fig. 4 shows the VEM design that operates on a single voxel. We enhance this design in the subsequent sections to enable multiple voxel updates and achieve significant bandwidth reduction in the process. The VEM comprises of three types of specialized processing elements: 1) theta evaluation module (TEM), in which the variables θ_1 and θ_2 are evaluated; 2) NPE, in which a complex one-to-one function is applied on each of the voxel neighbors; and 3) VUE, which uses the outputs of TEM and NPE to compute the updated value of the voxel and the error sinogram. The column of the A matrix is stored in a first-in-first-out (FIFO) buffer. Since the column of the A matrix is sparse, it is stored as an adjacency list, i.e., each memory location in the FIFO stores the index and the value of the nonzero entries. On the other hand, the error sinogram and the voxel neighbors are stored as dense arrays.

The VEM operates as follows. First, the elements of the A-matrix column are streamed into the TEM. The TEM

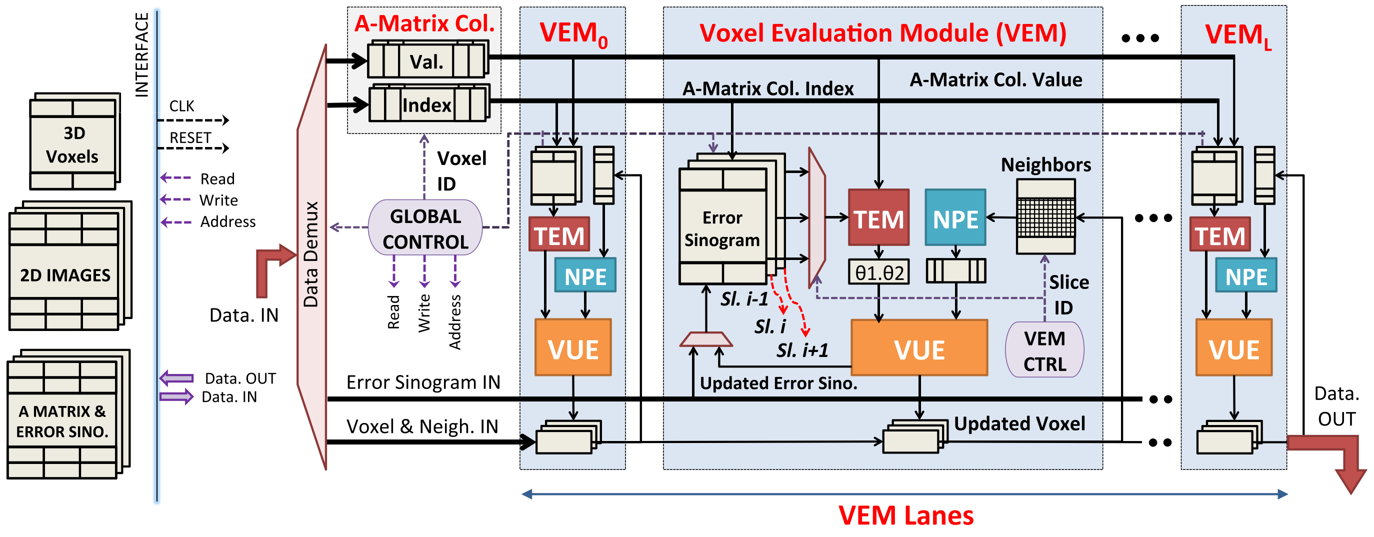


Fig. 3. EMBIRA: architecture block diagram.

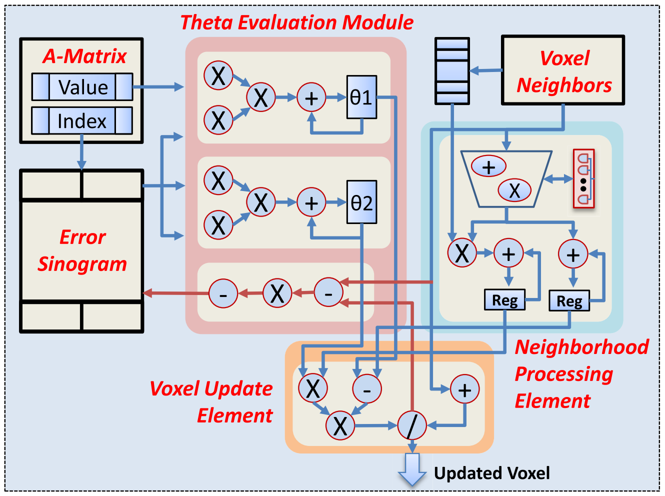


Fig. 4. Detailed block diagram of the VEM.

utilizes the index of the A-matrix elements to address the error sinogram memory to obtain the corresponding error sinogram value. The TEM performs a vector reduction operation on the A matrix and error sinogram values to obtain θ_1 and θ_2. In parallel to the TEM, the NPE sequentially operates on each of the voxel neighbors and stores the processed neighbor values in an FIFO memory. Since the TEM and the NPE operate in parallel, the performance of the VEM is maximized when their latencies are equal. This is achieved by proportionately allocating hardware resources in their implementation. After the TEM and the NPE complete execution, the VUE utilizes their outputs to compute the updated value of the voxel. As shown in (4), this involves performing a vector reduction operation on the voxel neighborhood, followed by multiple scalar operations. The entries in the error sinogram memory are also updated based on the new value of the voxel. Finally, the voxel is written back to the external memory. Note that the MBIR algorithm updates all the voxels (in a random order) in a given slice before choosing the next slice. Since all the voxels in a slice access the same region of the error sinogram (Fig. 2), the error sinogram memory in the VEM can be reused across all of them. However, the A-matrix column and the neighborhood voxels are unique to each voxel in a slice, and need to be fetched from an external memory before every voxel update.

To improve performance, the VEM is operated as a two-level nested pipeline. The first-level pipeline is within the TEM module. In this case, we leverage pipeline parallelism across the different elements of the vector reduction. When the TEM computes on a given A-matrix element, the error sinogram value for the successive element is fetched from the error sinogram memory in a pipelined manner. The second-level pipeline exploits the parallelism across successive voxels. In this case, we concurrently transfer data required by the next voxel, even as the previous voxel is being processed in the VEM. Thus, both the levels of pipelining improve the performance by overlapping communication (memory access) with computation.

B. VEM Array With Shared A-Matrix Memory

While the specialized processing elements and the two-level pipelined VEM operation aim to improve the performance of a single voxel update, much of the parallelism in MBIR stems from updating multiple voxels in parallel. We leverage this intervoxel parallelism by designing a VEM array, as shown in Fig. 3. The VEMs are arranged as an array with L lanes, each comprising its TEM, NPE, and VUE modules. One key constraint to ensure the convergence of the MBIR algorithm is that the voxels that are updated in parallel need to locate far apart in the 3-D volume. To maximize the separation, the voxels are typically selected from different slices that are spread equidistant from each other across the length of the 3-D volume. In our implementation of EMBIRA, we choose the number of VEM lanes to be 12, and the 3-D volume even in the case of our smallest benchmark data set contains 512 slices. As a result, voxels updated in parallel are approximately 40 slices apart, and therefore, updating voxels in parallel does not impact the convergence rate of the algorithm.

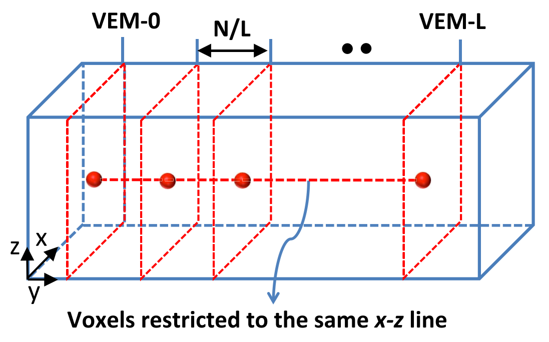


Fig. 5. Constraining voxels to lie on an xz line enables sharing of A-matrix column.

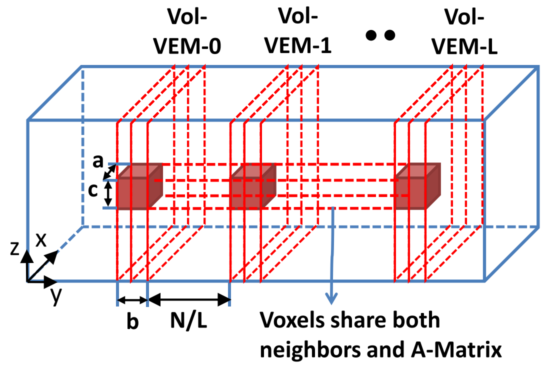


Fig. 6. Vol-VEMs sequentially update a volume around the selected voxel.

If the 3-D volume is smaller or if the number of VEMs is increased, then the voxels get progressively closer and may require additional iterations to achieve similar reconstruction quality.

While processing voxels from multiple slices in parallel linearly improves computation time, the amount of data transferred from the external memory remains constant. This is because the voxels updated in parallel are independent, and use different A-matrix columns and neighborhood. Since data transfer is pipelined with voxel evaluation, reducing one without the other has little impact on the overall performance of EMBIRA. We reduce the amount of data transferred by leveraging the manner in which the voxels access the different data structures of the MBIR algorithm. To this end, as shown in Fig. 5, we constrain how the voxels updated concurrently are chosen, by restricting them to lie on a straight line parallel to the y-axis. In other words, all voxels updated in parallel have the same x- and z-coordinates. Since A-matrix columns are indexed only using the x- and z-coordinates (Fig. 2), concurrently updated voxels share the same A-matrix column. Therefore, the data transfers from external memory for the A matrix are reduced in direct proportion to the number of concurrently updated voxels. This approximation does not impact convergence, as the slices from which the voxels are picked lie sufficiently far apart.

As shown in Fig. 3, the A-matrix memory is placed outside the VEMs and shared by all the VEMs during their execution. Before a batch of voxels is processed by the VEM array, the neighborhood for all voxels and one A-matrix column are transferred from the external memory. This results in a net reduction of $L - 1$ A-matrix column transfers per execution, where L is the number of VEM lanes. Thus, by constraining the sequence in which voxels are selected, we reduce the number of off-chip memory accesses.

C. Vol-VEM Design for Neighborhood Reuse

The reduction in the amount of data transferred due to the sharing of A-matrix columns yields diminishing improvements in the overall performance of EMBIRA as the number of VEM lanes is increased beyond a point. This is because, when a set of voxels are processed by the VEM array, the time taken to transfer the voxel neighbors to all VEMs increasingly outweighs (the already reduced) A-matrix column transfers as the number of VEM lanes is increased. Therefore, the neighborhood transfer time eventually limits the overall performance. To reduce the neighborhood data transferred, we leverage the observation from Fig. 2 that the adjacent voxels in the 3-D volume share part of their neighborhoods. Therefore, as shown in Fig. 6, we enhance the VEM design from updating a single voxel to updating a small 3-D volume of size $a \times b \times c$ (along the x-, y-, and z-directions) around the voxel. We term this the voxel volume evaluation module (Vol-VEM) design.

The reduction in neighborhood transfers can be computed as follows. Each voxel update requires a $3 \times 3 \times 3$ neighborhood (one neighbor voxel in all the directions) around the voxel to be brought in from the external memory. In the case of Vol-VEM updating an $a \times b \times c$ volume, a neighborhood of $(a + 2) \times (b + 2) \times (c + 2)$ voxels is required to collectively update the entire volume. In the previous case, when the VEM is designed to evaluate a single voxel, updating abc voxels would require $27 \times abc$ transfers. Therefore, the total reduction in data transferred is given by

$$\text{NeighborDataRed}_{abc} = \frac{(a+2)(b+2)(c+2)}{27 * abc}. \quad (8)$$

For example, if the Vol-VEM evaluates a $3 \times 3 \times 3$ volume, we achieve a 5.8× reduction in neighborhood transfers. Clearly, the benefits increase as the size of the volume is increased. However, increasing the volume further constrains the order in which the voxels are updated, and hence, may impact convergence. Therefore, it is key to choose the largest volume that yields the desired reconstruction quality. In addition, from an efficiency standpoint, increasing the volume may not be advantageous beyond a point, as the impact of neighborhood data transfers quickly diminishes. Another important point to note is that the voxels in the volume are updated sequentially within the Vol-VEM, as there exist data dependencies between adjacent voxels. For example, voxels in the same slice (y-coordinate) share the error sinogram memory, which is modified to reflect the decrease in error after each voxel is evaluated. This updated error sinogram memory should be used to compute the next voxel in the slice. However, if multiple voxels from the slice are updated in parallel, then updates to the same error sinogram location from parallel voxels would conflict. Thus, the Vol-VEM optimization is not intended to decrease the compute time of VEMs.

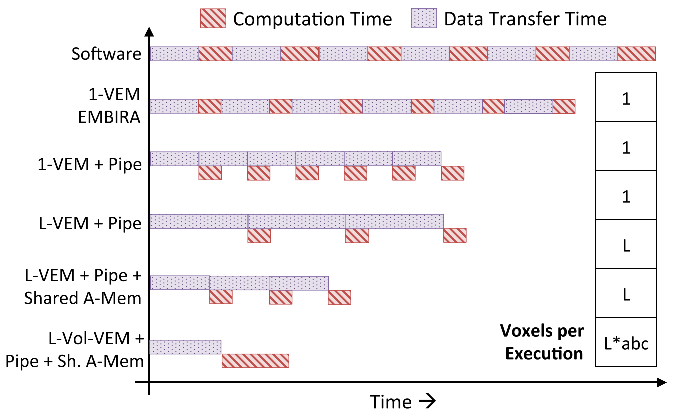


Fig. 7. Summary of optimizations proposed in EMBIRA.

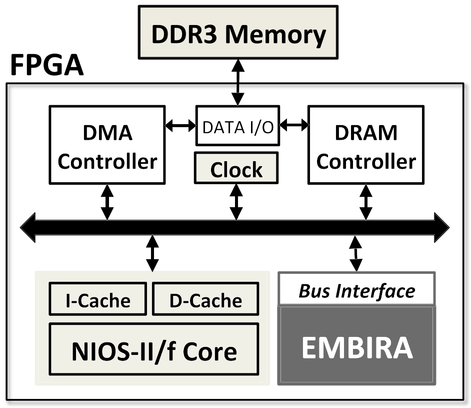


Fig. 8. System to evaluate EMBIRA.

Instead, the number of external memory accesses is decreased due to increased data reuse.

To realize Vol-VEM, the following enhancements are made to the VEM design described in Section III-A. First, the capacity of the neighborhood memory is increased to hold a larger number of voxels. In addition, once the updated value of a voxel is computed, it needs to be written to the neighborhood memory (in addition to the external memory), as subsequent voxels in the volume use the updated value. Also, since adjacent voxels along the y-direction belong to different slices, the error sinogram memory in the Vol-VEM is replicated to hold data corresponding to each slice. Along similar lines, voxels within the slice use different A-matrix columns, and correspondingly the A-matrix memory is also replicated. The TEM, NPE, and VUE modules are not replicated in the Vol-VEMs as voxels in the volume are evaluated sequentially. Finally, the global and VEM controllers are modified to appropriately index these memories and evaluate all voxels within the volume.

The different optimizations proposed in EMBIRA and their qualitative impact on its execution time are summarized in Fig. 7. First, the VEM design utilizes three types of specialized processing elements to improve the computation time for voxel evaluation (1-VEM). Next, the two-level pipelined operation further improves the performance by overlapping data transfer with computation (1-VEM + Pipe). In order to exploit intervoxel parallelism, multiple VEMs are arranged as a 1-D array (L-VEM + Pipe). However, this does not improve the overall performance as the data transfer time, which is the critical path in the pipeline, is not reduced. Therefore, we impose an additional constraint that the voxels updated in parallel should lie along a straight line parallel to the y-axis. This enables the A-matrix column to be shared among the VEMs, resulting in reduced data transfer time (L-VEM + Pipe + Sh. A-Mem). We note that the performance improvements from this optimization saturate for a large number of parallel VEMs, as the A-matrix transfer time is no longer dominant. Therefore, we propose the Vol-VEM design, in which each VEM is enhanced to update a small $a \times b \times c$ volume, instead of single voxel. This significantly reduces the neighborhood transfer time, further enhancing performance (L-Vol-VEM + Pipe + Sh. A-Mem). In summary, through the above optimizations, EMBIRA achieves significant efficiency in realizing 3-D construction using the MBIR algorithm.

IV. EMBIRA: System-Level Evaluation

To evaluate the benefits of EMBIRA, we integrate it as part of a larger system, and prototype the system on a commodity FPGA platform. This section details the different components of the system design.

A. System Design

Fig. 8 shows the system used for evaluating EMBIRA. The system is comprised of four key components: 1) the EMBIRA accelerator; 2) an NIOS II/f general-purpose scalar processor; 3) a DRAM memory controller; and 4) a direct memory access (DMA) controller. The components communicate with each other through a system bus. The NIOS II/f core uses a 32-bit, six-stage, in-order pipelined embedded processor architecture, along with an instruction and data cache. The system operates as a single frequency island, i.e., all components share the same clock frequency. To this end, an on-chip PLL module is utilized to generate the clock signal. The system is the interfaced with an external DDR3 memory that stores the data required by the MBIR algorithm.

B. System Operation

First, the DDR3 memory is initialized with the input 2-D images and other data structures (error sinograms, A matrix, and so on) associated with the MBIR algorithm. Next, the general-purpose processor core begins execution and generates a random slice-id (y-coordinate of the voxel). If multiple voxels are updated in parallel, then the slice-ids of the other voxels are obtained by locating them equidistant from each other, in order to maximize the distances between the voxels. The DMA controller is then used to transfer the error sinograms corresponding to the slices to the VEMs in EMBIRA. Once the transfer is complete, the general-purpose processor core generates an (x and z) coordinate at random, and the appropriate column of the A matrix is fetched from memory. Note that all the VEMs in EMBIRA share this column. The VEMs then begin to compute and write updated voxel values to the external memory. Concurrently, the general-purpose

TABLE I
PARAMETERS OF SYSTEM COMPONENTS

EMBIRA	*Config1*: 1 VEM, 200 MHz frequency, 100 KB memory *Config2*: 12 VEMs, 165 MHz frequency, 1.2 MB memory *Config3*: 12 Vol-VEMs, 165 MHz frequency, 4 MB memory
NIOS II/f core	32-bit processor, 6-stage in-order pipeline with branch prediction, 407 MIPS, 64 KB cache, 350 MHz max. frequency
DDR3 Memory	3 GB, 8 banks, 400 MHz frequency

processor core generates the next (x and z) coordinate and begins transferring the required column of the A matrix. Once all the (x and z) coordinates in the slices are exhausted, the next set of slices is determined and the process is repeated until voxels in all the slices are updated. This completes one full iteration of the reconstruction. The general-purpose processor core then determines if the convergence criterion is met and initiates the next iteration if required. Thus, the entire 3-D reconstruction from the 2-D images is achieved using EMBIRA.

V. EXPERIMENTAL METHODOLOGY

In this section, we describe the methodology adopted in our experiments.

A. System Implementation Toolflow

The EMBIRA accelerator was implemented at the register-transfer level using Verilog Hardware Description Language. The other components in the system, including the NIOS II/f processor core, were from the Altera IP library. The system bus was implemented in compliance with Avalon memory-mapped interface specifications. The external DDR3 memory was 3 GB in capacity. The system was developed using the Altera Qsys system integration tool and prototyped on an Altera DE5 FPGA platform [22] with a Stratix V GX FPGA. The parameters associated with the key system components are listed in Table I. Quartus was used to synthesize, map, and place-and-route the design to obtain the FPGA bitstream. We used Altera's NIOS II IDE to generate the application software that executes on the NIOS II/f processor. Finally, a JTAG interface was used to download the bitstream on to the Stratix V FPGA, and was further utilized for evaluation and debug purposes.

The sequential and multithreaded software baselines of the MBIR algorithm were adopted from [7]. The software was implemented in C++ and executed on a server with 48 AMD Opteron cores operating at 2.3 GHz and 196-GB memory.

B. Runtime, Energy, and Quality Evaluation

To measure runtime, we instrumented the EMBIRA system and the software implementation with performance counters. The number of clock ticks for the full 3-D reconstruction was obtained in both the cases. We utilized the Quartus PowerPlay Analyzer to measure the power consumed by the full system after place-and-route of the design. In the case of the software implementation, we obtained the power consumed by the processor chip used in the server from its datasheet. The energy was then computed as a product of the runtime and power. Since we slightly modify the original MBIR algorithm (by constraining the order in which the voxels are updated) to improve the efficiency of the implementation, we quantify the quality loss incurred in the execution by computing the root mean square error (RMSE) between the outputs produced by the sequential software implementation and the EMBIRA implementation.

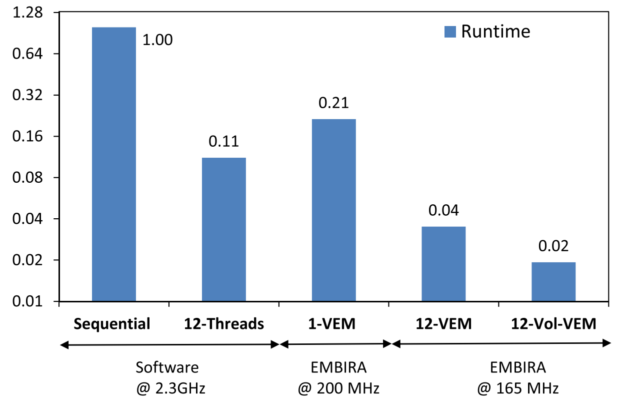


Fig. 9. Comparison of EMBIRA runtime with sequential and multithreaded software.

C. Application Benchmarks

We evaluate EMBIRA using two representative bright-field electron tomography (BF-ET) data sets. In BF-ET, a sequence of images are obtained by repeatedly tilting and imaging a sample about a single axis in an electron microscope (EM). The goal of BF-ET is to reconstruct the 3-D volume of the sample from the set of acquired images. The first data set contains 47 images with dimensions 512 × 512 obtained by simulating the measurement process of aluminum nanoparticles in an EM [6]. A 3-D volume of size 512 × 512 × 256 voxels is reconstructed using the 2-D images. The second data set is from the electron microscopy of gold nanoparticles [23]. It contains 28 2-D measurements of size 1024 × 1024, from which a volume of size 1024 × 1024 × 200 is reconstructed.

VI. RESULTS

In this section, we present the results of various experiments that demonstrate the benefits of EMBIRA. All the results were obtained by implementing the EMBIRA system (Section IV) on the Altera DE5 FPGA platform and executing the entire reconstruction application on the same.

A. Runtime and Energy Improvement

First, we compare the runtime and energy of EMBIRA with sequential and multithreaded software implementations. Fig. 9 shows the runtime[2] of three different EMBIRA configurations normalized to the sequential software implementation across both the applications. We note that the runtime includes the execution time for a complete reconstruction, including the times taken by the control software on the NIOS processor and DDR3 memory transfers. We find that the multithreaded software implementation with 12 threads

[2]The runtime is measured in absolute wall-clock seconds required for the entire 3-D reconstruction, without any normalization for the different operating frequencies of FPGA versus server. An Application Specific Integrated Circuit implementation of EMBIRA would achieve higher improvements.

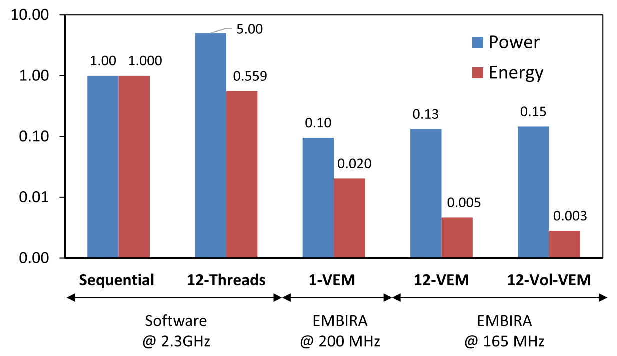
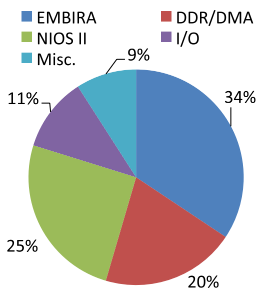


Fig. 10. Power and energy benefits using EMBIRA.



Module	LUT	DSP	Embedded Memory
EMBIRA	57%	98%	59%
NIOS II	1%	1%	10%
DDR	3%	0%	1%
Misc.	1%	0%	2%
Total	**62%**	**99%**	**72%**

Fig. 11. Breakdown of power (left) and FPGA resource utilization (right).

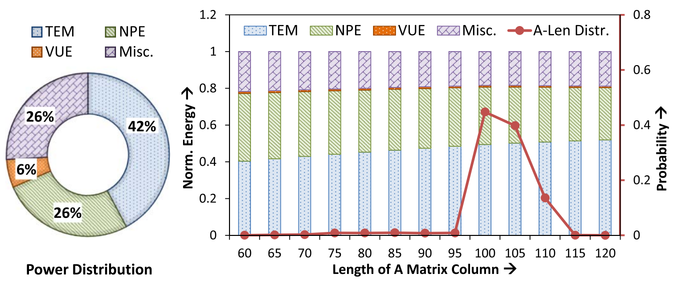


Fig. 12. Power and energy distribution within different modules in EMBIRA.

is ∼9× faster than the sequential implementation. We do not increase the thread count beyond 12, as it yields no further improvement in performance due to limited memory bandwidth, increased synchronization overheads, and other factors. The first EMBIRA configuration is the pipelined single VEM design that updates one voxel per execution. We achieved a frequency of 200 MHz when the design was synthesized on the FGPA. Even at 200 MHz, the 1-VEM design is ∼4.75× faster than the sequential implementation that is operated at 2.3 GHz. The speedup stems from the specialized processing units used in the design of VEM and the two-level nested pipelined operation. However, the 1-VEM configuration is ∼1.9× slower than the multithreaded implementation.

The second EMBIRA configuration utilizes 12 VEMs with the shared *A*-matrix memory, with each VEM evaluating a single voxel. In this case, the maximum operating frequency of the FPGA implementation was reduced to 165 MHz due to increased design complexity. The 12-VEM design yielded 28.33× and 3.16× improvement in performance compared with sequential and multithreaded software implementations, respectively. The final configuration is the 12-Vol-VEM design, in which the VEM array is comprised of 12 lanes and each VEM evaluates nine voxels in sequence. The reduced neighborhood data transfer time in Vol-VEMs yields an additional ∼1.9× performance improvement compared with the 12-VEM configuration. Overall, the performance improvement is 51.8× compared with the sequential software, and 5.8× compared with the multithreaded software implementation. Thus, the architecture and optimizations proposed in EMBIRA can yield significant improvement in the runtime of the MBIR algorithm.

Next, Fig. 10 compares the power and energy of the EMBIRA and software implementations. The benefits are normalized to the sequential software implementation. The 1-VEM configuration is 10.5× more power efficient than the sequential software, and combined with its 4.75× performance improvement, yields a significant 49.15× improvement in energy. The 12-VEM and the 12-Vol-VEM are more energy efficient, as their power grows only by 1.35× and 1.52×, respectively, than the 1-VEM design, while they achieve larger performance improvements. Overall, the energy of the 12-VEM and 12-Vol-VEM configurations is 215× and 355× lower, respectively, than the sequential software implementation. The multithreaded software implementation is ∼2× more energy efficient than its sequential counterpart, as the improvement in performance outweighs the increase in power. Accordingly, the 1-VEM, 12-VEM, and 12-Vol-VEM configurations are 27.4×, 120.1×, and 199× more energy efficient than the multithreaded software implementation, respectively.

B. Energy Breakdown and Analysis

Fig. 11 shows the power and FPGA resource utilization of the different system components. We utilize EMBIRA with 12 Vol-VEMs for this analysis. We find that EMBIRA is the most dominant component of the design, consuming over 34% of the overall power and a significant fraction of the lookup tables, DSP units, and embedded memory blocks on the FPGA. The NIOS II/f processor and its caches amount to 25% of the system power. The memory controllers and the data I/O peripherals also account for a considerable fraction of the system power. The remaining power is expended in other components, such as the system bus. Note that in all our experiments, we limit the number of VEM lanes to 12, as it exhausts all the DSP resources on the FPGA.

Next, we examine the energy breakdown within the different modules of EMBIRA in greater detail. Fig. 12 shows the power and energy distribution of the three processing elements in EMBIRA. From the power distribution, we find that the TEM is the most complex of the processing elements, followed by NPE and VUE, respectively. The other components, such as memory and control, also contribute a significant fraction of the total power. An interesting aspect of the design is that its latency and energy for voxel evaluation are a strong function of the number of nonzero entries in the *A*-matrix column corresponding to the voxel. Therefore, Fig. 12 shows the energy distribution for various values of *A*-matrix column length. We observe that the proportion of energy consumed by the TEM increases with the length of *A*-matrix column,

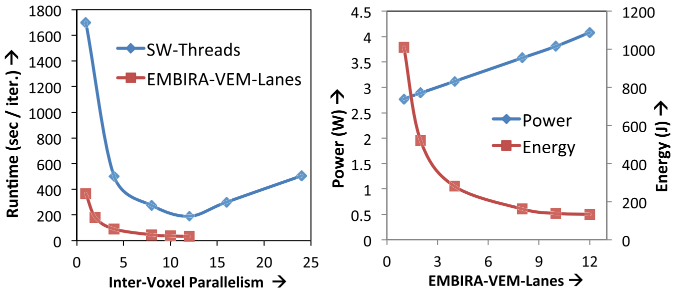


Fig. 13. Runtime (left), and power and energy (right) of EMBIRA with increasing VEM lanes.

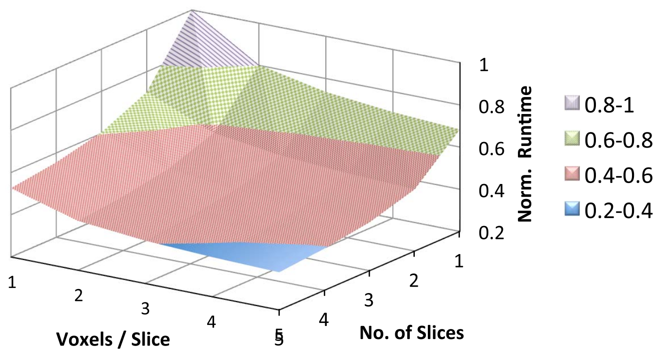


Fig. 14. Normalized runtime with varying slices and voxel per slice in Vol-VEMs.

from 38% at a length of 60 to 52% at a length of 120. Since the TEM and NPE operate in parallel, we need to balance their latencies to maximize throughput. However, this may not be always possible, as the A-matrix column length dynamically differs based on the voxel being processed. Therefore, we examined the distribution of the A-matrix column length (Fig. 12) for our application and optimized the design for its most common range (95–105). In this region, we find that the ratio of the TEM and NPE energies is roughly equal to the ratio of their powers, indicating that their latencies are balanced.

C. Design Space Exploration

One of the key attributes of the EMBIRA architecture is that its performance and energy can be easily scaled by modulating its microarchitectural parameters. The EMBIRA architecture has three key parameters: 1) number of VEM lanes; 2) number of slices within each Vol-VEM; and 3) number voxels per slice within the Vol-VEM. In this section, we perform a design space exploration by varying these parameters and studying their impact on performance and energy.

Fig. 13 shows the performance, power, and energy of EMBIRA with increasing VEM lanes. The performance is also compared with the multithreaded software baseline, in which the number of threads is increased. In the case of the software implementation, we find that the runtime improves for smaller thread counts, but increases beyond 12 threads due to various parallelization bottlenecks. Similarly, the performance of EMBIRA improves with increased VEM lanes. However, the rate of increase saturates beyond a point as the contribution of the A-matrix column data transfer to the overall runtime grows progressively smaller. Clearly, the neighborhood data transfer dominates the runtime, and this underscores the need for the Vol-VEM optimization. Fig. 13 also shows the power and energy of the overall system as the VEM lanes are increased. We observe that the increase in power consumption is outweighed by the improvement in performance, which results in significant improvement in energy efficiency. However, beyond a point, the benefits in energy also saturate.

We now analyze the impact of the Vol-VEM optimization, in which each VEM updates a volume around a selected voxel. The number of voxels in the volume is the product of the number of slices in the Vol-VEM (length of the y-direction) and the number of voxels in each slice. Fig. 14 shows the normalized runtime obtained by varying these parameters. We observe a 2.65× improvement in performance as the number of voxels in the volume is increased from 1 (1, 1) to 25 (5, 5). Fig. 14 also reveals another interesting trend. We find that the improvement in performance is more significant (larger slope) if the number of slices in the Vol-VEM is increased compared with the number of voxels in each slice—1.9× versus 1.46×. This stems from the fact that voxels in a slice index different A-matrix columns. Therefore, if the number of voxels in the slice is increased, multiple A-matrix columns corresponding each voxel should be transferred from the external memory. On the other hand, if the number of slices is increased, adjacent voxels will automatically lie along the same xz line, and therefore, share the same A-matrix column. In summary, increasing slices enables sharing of both the neighborhood as well as the A-matrix column, whereas increasing voxels in a slice allows only the neighborhood to be shared. Hence, it is preferable to increase the number of slices rather than the number of voxels in a slice. However, it is noteworthy that increasing both the parameters simultaneously allows for better sharing of the neighborhood from all the directions. Therefore, it is key to holistically evaluate these factors in determining the dimensions of the volume updated within the Vol-VEM to achieve the best performance.

D. Impact of Constrained Voxel-Update Order and Fixed-Point Implementation

In this section, we evaluate the impact of constraining the order in which voxels are updated on the final 3-D reconstruction quality. Due to limited resources on the FPGA, the MBIR algorithm was implemented using fixed point representation. We also quantify the impact of fixed point implementation on the output quality by comparing it with a floating point baseline implemented in software. Note that the output quality is measured as the RMSE between the actual output and the output produced using EMBIRA.

First, we study the impact of fixed point representation on the output reconstruction quality. We individually optimized the precision of each data structure in the MBIR algorithm to minimize power (and FPGA resources), with negligible loss in reconstruction quality. The fixed-point number format used to represent the key data structures in MBIR is shown in Table II.

Fig. 15 shows the RMSE of the sequential fixed point and floating point implementations after each iteration of

TABLE II
DATA PRECISION AND FIXED-POINT NUMBER FORMATS OF THE DIFFERENT DATA STRUCTURES IN MBIR

Data Structure	Data Bitwidth	Fixed-point Format
2D images	8	0.8
3D volume	16	0.16
A matrix	8	3.5
Error sinogram	24	3.21
θ_1, θ_2	36	24.12

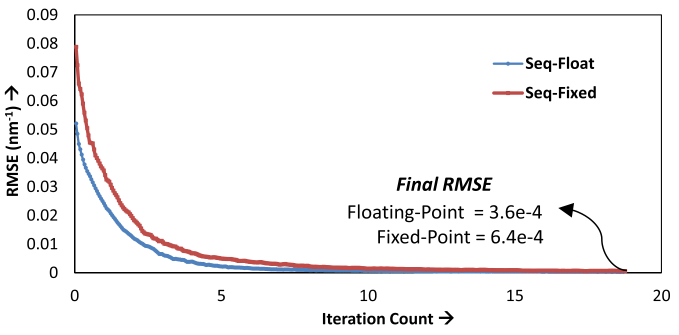


Fig. 15. RMSE after each iteration with sequential floating point and fixed point implementations.

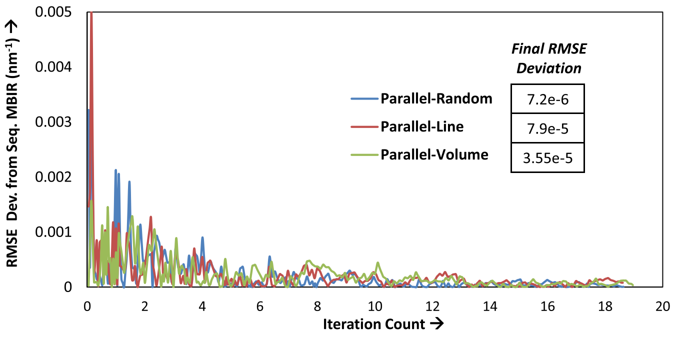


Fig. 16. RMSE deviation due to constrained order of voxel evaluation.

the algorithm. In this context, one update of all voxels in the entire 3-D volume constitutes an iteration. As observed in Fig. 15, the RMSE decreases exponentially with the number of iterations. The difference between the fixed and floating point implementations is noticeable in the initial iterations. However, the fixed point implementation quickly converges and achieves an RMSE very similar to the floating point case.

Next, we compare the RMSE deviation from the sequential fixed point implementation when the order of voxel updates is constrained. In particular, Fig. 16 considers three different cases: 1) parallel random, in which multiple voxels are updated in parallel but are chosen independent of each other; 2) parallel line, in which the voxels updated in parallel are constrained to lie on along an xz line (12-VEM EMBIRA configuration in Section VI-A); and 3) parallel volume, in which the voxels lie on a straight line, and a volume around each voxel is updated (12-Vol-VEM configuration in Section VI-A). We find that in all the three cases, the final RMSE deviation is quite small, and is over an order of magnitude smaller than the actual RMSE value.

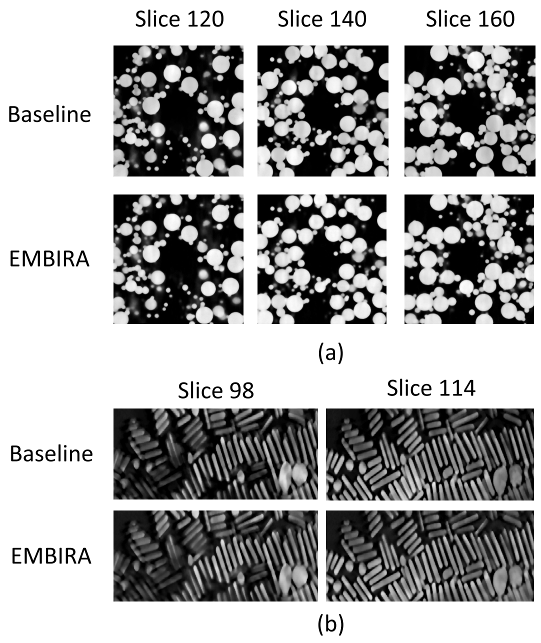


Fig. 17. Selected slices of 3-D volume reconstructed using different implementations. (a) Aluminum. (b) Gold.

Finally, Fig. 17 shows the selected slices of the 3-D volume reconstructed using the baseline and EMBIRA implementations in the context of both the applications. These images visually reaffirm the fact that constraining the order of voxel updates has virtually no impact on the reconstruction quality of the MBIR algorithm.

In summary, the above results demonstrate the efficiency of the architecture and optimizations proposed in EMBIRA.

VII. RELATED WORK

Tomographic reconstruction is key to several important application domains viz., medical imaging, electron microscopy, nondestructive testing, and explosive detection. Realizing the compute intensive nature of the algorithms involved, the previous research efforts have proposed different optimizations to improve their implementation efficiency. In this section, we present a brief overview of related efforts and highlight the distinguishing features of this paper.

A number of previous works have been devoted to developing custom accelerators for tomographic reconstruction in the context of medical imaging. These efforts target different algorithms, all of which are used for CT. The first class of efforts [24] focus on noniterative reconstruction algorithms, i.e., a closed form solution is used to evaluate the value of each voxel. The second class of approaches, such as [25] and [26], utilizes iterative algorithms, such as expectation maximization, that are not fully model-based. Studies have shown that both the noniterative and nonmodel-based approaches are less robust and yield poor reconstruction quality [4].

The final set of efforts [27], [28] employs a model-based method, whose cost function is similar to the one shown in (1). However, they differ in the type of optimization algorithm used to minimize the high-dimensional cost function. In particular, they employ the so-called simultaneous methods [10], [12], such as gradient descent, conjugate gradient, and so on, which are computationally quite different from the ICD approach considered in this paper. A comprehensive comparison of both the approaches is provided in [29]. Both the optimization strategies theoretically yield similar reconstruction qualities as they minimize the same cost function. However, their computational patterns and convergence rates are quite different. In the case of simultaneous methods, all the voxels in the volume are updated in parallel, and therefore, the forward/back projection steps resemble a large matrix-vector product, which parallelizes well on commercial multi/many core platforms. On the other hand, in the coordinate descent approach, only voxels that are far away from each other can be updated in parallel and these voxels are selected at random. Therefore, the processed voxels do not share any data, and as explained in Section I, the implementation quickly becomes memory bandwidth-limited when parallelized on multi/many core platforms. Thus, due to the fundamental differences in the computation patterns between the two optimization methods, the strategies employed in [27] and [28] cannot be leveraged in the context of ICD-MBIR, and hence, EMBIRA.

The key advantage of coordinate descent approaches is that their rate of convergence is faster than the simultaneous methods. As demonstrated in [29], coordinate descent approaches require $\sim 6\times$ fewer number of iterations to converge, and therefore, have relatively low compute complexity. Hence, although the computation patterns are more irregular, ICD-MBIR has the potential to yield computationally efficient implementations. We note that we are unable to provide a quantitative comparison between EMBIRA and [27] and [28] as they implement only specific functions of their algorithms (but not the entire reconstruction), and the data sets used were quite different.

Another effort [30] implements selected commonly used functions in the context of medical imaging. Other recent efforts have also proposed custom architectures that enable medical imaging with other sensing modalities. Some examples in this direction include [31]–[33], where custom architectures are used to improve the efficiency of ultrasound imaging and magnetic resonant imaging.

VIII. Conclusion

Inverse problems, such as 3-D tomographic reconstruction, find wide applicability in many imaging systems. The MBIR algorithm, which iteratively reconstructs a 3-D volume from multiple 2-D observations, places significant compute and data demand, leading to its general-purpose software implementations yielding poor performance and energy efficiency. We address this important problem by proposing EMBIRA, an efficient hardware accelerator that implements the MBIR algorithm. The key compute element of EMBIRA is the VEM that consists of three specialized processing elements, and is operated as a two-level nested pipeline. Multiple VEMs are arranged as a 1-D array to enable voxels to be updated concurrently. To further improve efficiency, we constrain the order in which the voxels are updated by EMBIRA to enhance data reuse within the accelerator and reduce the number of external memory accesses. We prototyped EMBIRA on an FPGA platform and demonstrated significant improvements in both performance and energy over well-optimized software implementations. We believe that such efficiency improvements can greatly facilitate the adoption of MBIR algorithms in commercial applications.

References

[1] C. A. Bouman, *Model Based Image Processing*, 1st ed. 2013. [Online]. Available: https://engineering.purdue.edu/~bouman/publications/pdf/MBIP-book.pdf

[2] S. J. Kisner *et al.*, "Innovative data weighting for iterative reconstruction in a helical CT security baggage scanner," in *Proc. 47th Int. Carnahan Conf. Secur. Technol. (ICCST)*, Oct. 2013, pp. 1–5.

[3] E. A. Smith, J. R. Dillman, M. M. Goodsitt, E. G. Christodoulou, N. Keshavarzi, and P. J. Strouse, "Model-based iterative reconstruction: Effect on patient radiation dose and image quality in pediatric body CT," *Radiology*, vol. 270, no. 2, pp. 526–534, Feb. 2014.

[4] M. Katsura *et al.*, "Model-based iterative reconstruction technique for radiation dose reduction in chest CT: Comparison with the adaptive statistical iterative reconstruction technique," *Eur. Radiol.*, vol. 22, no. 8, pp. 1613–1623, Aug. 2012.

[5] C. A. Bouman and K. Sauer, "A unified approach to statistical tomography using coordinate descent optimization," *IEEE Trans. Image Process.*, vol. 5, no. 3, pp. 480–492, Mar. 1996.

[6] S. V. Venkatakrishnan, L. F. Drummy, M. Jackson, M. De Graef, J. Simmons, and C. A. Bouman, "A model based iterative reconstruction algorithm for high angle annular dark field-scanning transmission electron microscope (HAADF-STEM) tomography," *IEEE Trans. Image Process.*, vol. 22, no. 11, pp. 4532–4544, Nov. 2013.

[7] S. V. Venkatakrishnan, L. F. Drummy, M. Jackson, M. De Graef, J. Simmons, and C. A. Bouman, "Model-based iterative reconstruction for bright-field electron tomography," *IEEE Trans. Comput. Imag.*, vol. 1, no. 1, pp. 1–15, Mar. 2015.

[8] K. A. Mohan, S. V. Venkatakrishnan, L. F. Drummy, J. Simmons, D. Y. Parkinson, and C. A. Bouman, "Model-based iterative reconstruction for synchrotron X-ray tomography," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2014, pp. 6909–6913.

[9] P. A. Midgley and R. E. Dunin-Borkowski, "Electron tomography and holography in materials science," *Nature Mater.*, vol. 8, no. 4, pp. 271–280, Apr. 2009.

[10] J. A. Fessler and S. D. Booth, "Conjugate-gradient preconditioning methods for shift-variant PET image reconstruction," *IEEE Trans. Image Process.*, vol. 8, no. 5, pp. 688–699, May 1999.

[11] J. A. Fessler, "Penalized weighted least-squares image reconstruction for positron emission tomography," *IEEE Trans. Med. Imag.*, vol. 13, no. 2, pp. 290–300, Jun. 1994.

[12] H. M. Hudson and R. S. Larkin, "Accelerated image reconstruction using ordered subsets of projection data," *IEEE Trans. Med. Imag.*, vol. 13, no. 4, pp. 601–609, Dec. 1994.

[13] S. V. Venkatakrishnan, L. F. Drummy, M. De Graef, J. P. Simmons, and C. A. Bouman, "Model based iterative reconstruction for Bright Field electron tomography," *Proc. SPIE*, vol. 8657, pp. 86570A-1–86570A-12, Feb. 2013.

[14] *GE Healthcare Veo Website*, accessed on Apr. 2016. [Online]. Available: http://www3.gehealthcare.com/en/Products/Categories/Computed_Tomography/Discovery_CT750_HD/Veo

[15] K. Evans, T. Sych, and S. Johnson, "New levels of CT image performance and new levels in radiation dose management: Advanced image reconstruction algorithm running on Intel Xeon processors," Intel/GE Healthcare, Santa Clara, CA, USA, White Paper, 2011.

[16] *TEMBIR-HAADF, TEMBIR-BF, and MBIR-Synchrotron Software Available at Website*, accessed on Apr. 2016. [Online]. Available: http://www.openmbir.org

[17] W. J. Palenstijn, K. J. Batenburg, and J. Sijbers, "The ASTRA tomography toolbox," in *Proc. CMMSE*, 2013, pp. 1–7.

[18] J. Fessler. *MATLAB Tomography Toolbox*, accessed on Apr. 2016. [Online]. Available: http://web.eecs.umich.edu/~fessler/code/index.html

[19] J. Zheng, S. Saquib, K. Sauer, and C. A. Bouman, "Parallelizable Bayesian tomography algorithms with rapid, guaranteed convergence," *IEEE Trans. Image Process.*, vol. 9, no. 10, pp. 1745–1759, Oct. 2000.

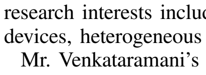
[20] Z. Yu, J. B. Thibault, C. A. Bouman, K. D. Sauer, and J. Hsieh, "Fast model-based X-ray CT reconstruction using spatially nonhomogeneous ICD optimization," *IEEE Trans. Image Process.*, vol. 20, no. 1, pp. 161–175, Jan. 2011.

[21] J.-B. Thibault, K. D. Sauer, C. A. Bouman, and J. Hsieh, "A three-dimensional statistical approach to improved image quality for multislice helical CT," *Med. Phys.*, vol. 34, no. 11, pp. 4526–4544, 2007.

[22] *DE5 Development and Education Board*, accessed on Apr. 2016. [Online]. Available: http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&No=526
[23] K. Park *et al.*, "Growth mechanism of gold nanorods," *Chem. Mater.*, vol. 25, no. 4, pp. 555–563, 2013.
[24] J. Li, C. Papachristou, and R. Shekhar, "An FPGA-based computing platform for real-time 3D medical imaging and its application to cone-beam CT reconstruction," *J. Imag. Sci. Technol.*, vol. 49, pp. 237–245, May 2005.
[25] J. Chen, J. Cong, M. Yan, and Y. Zou, "FPGA-accelerated 3D reconstruction using compressive sensing," in *Proc. ACM/SIGDA Int. Symp. Field Program. Gate Arrays (FPGA)*, New York, NY, USA, 2012, pp. 163–166.
[26] J. Chen, J. Cong, L. A. Vese, J. Villasenor, M. Yan, and Y. Zou, "A hybrid architecture for compressive sensing 3-D CT reconstruction," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 2, no. 3, pp. 616–625, Sep. 2012.
[27] J. K. Kim, J. A. Fessler, and Z. Zhang, "Forward-projection architecture for fast iterative image reconstruction in X-ray CT," *IEEE Trans. Signal Process.*, vol. 60, no. 10, pp. 5508–5518, Oct. 2012.
[28] J. K. Kim, Z. Zhang, and J. A. Fessler, "Hardware acceleration of iterative image reconstruction for X-ray computed tomography," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2011, pp. 1697–1700.
[29] B. De Man *et al.*, "A study of four minimization approaches for iterative reconstruction in X-ray CT," in *Proc. IEEE Nucl. Sci. Symp. Conf. Rec.*, vol. 5. Oct. 2005, pp. 2708–2710.
[30] G. Dasika, A. Sethia, V. Robby, T. Mudge, and S. Mahlke, "MEDICS: Ultra-portable processing for medical image reconstruction," in *Proc. 19th Int. Conf. Parallel Archit. Compilation Techn. (PACT)*, New York, NY, USA, 2010, pp. 181–192.
[31] R. Sampson, M. Yang, S. Wei, C. Chakrabarti, and T. F. Wenisch, "Sonic Millip3De: A massively parallel 3D-stacked accelerator for 3D ultrasound," in *Proc. IEEE 19th Int. Symp. High Perform. Comput. Archit. (HPCA)*, Washington, DC, USA, Feb. 2013, pp. 318–329.
[32] M. Birk, M. Zapf, M. Balzer, N. Ruiter, and J. Becker, "A comprehensive comparison of GPU- and FPGA-based acceleration of reflection image reconstruction for 3D ultrasound computer tomography," *J. Real-Time Image Process.*, vol. 9, no. 1, pp. 159–170, Mar. 2014.
[33] G. Dasika, K. Fan, and S. Mahlke, "Power-efficient medical image processing using PUMA," in *Proc. IEEE 7th Symp. Appl. Specific Processors (SASP)*, Jul. 2009, pp. 29–34.

Junshi Liu received the B.S. degree in electronics engineering from the College of Electronic Engineering, Zhejiang University, Hangzhou, China, in 2010, where he is currently pursuing the Ph.D. degree with the Institute of VLSI Design.

His current research interests include hardware acceleration, image process, embedded system, approximate computing, and 3-D imaging.



Swagath Venkataramani (S'13) received the bachelor's degree in electrical and electronics engineering from the College of Engineering, Anna University, Guindy, India, in 2010, as the University Gold Medalist. He is currently pursuing the Ph.D. degree with the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, USA.

He has been with the Exa-Scale Computing Group, Intel, Hillsboro, OR, USA, as part of the U.S. DOE's FastForward Program, and with the Sensing and Energy Research Group, Microsoft Research, Seattle, WA, USA. His current research interests include approximate computing, computing with spintronic devices, heterogeneous parallel architectures, and computational imaging.

Mr. Venkataramani's dissertation research received the Intel Ph.D. Fellowship in computing leadership and the Purdue Bilsland Dissertation Fellowship, and has been featured in *MIT Technology Review*, Slashdot, Physics Today, and NSF News From the Field.

Singanallur V. Venkatakrishnan (S'12) received the B.Tech. degree in electronics and communication engineering from the National Institute of Technology Tiruchirappalli, Tiruchirappalli, India, in 2007, and the M.S. and Ph.D. degrees in electrical and computer engineering from Purdue University, West Lafayette, IN, USA, in 2009 and 2014, respectively.

He was a Research and Development Engineer with Baker Hughes Inc., Houston, TX, USA, from 2009 to 2010, where he was involved in logging while drilling-imaging magnetometers. He is currently a Post-Doctoral Fellow with Advanced Light Source, Lawrence Berkeley National Laboratory, Berkeley, CA, USA. His current research interests include statistical information processing, inverse problems, computational imaging, and machine learning.

Yun Pan (M'11) received the B.S. degree from the Department of Information Science and Electronic Engineering, Zhejiang University, Hangzhou, China, in 2002, and the Ph.D. degree from the Department of Electronic Engineering, Tsinghua University, Beijing, China, in 2008.

He held a post-doctoral position with the Institute of VLSI Design, College of Electrical Engineering, Zhejiang University, in 2008. He is currently an Associate Professor with the Department of Information Science and Electronic Engineering, Zhejiang University. He has authored over 40 academic papers, co-authored two books and one chapter, and holds nine Chinese patents in his research areas. His current research interests include on-chip communication, mobile computing, mobile healthcare, and smart camera systems.

Prof. Pan is also a member of the China Computer Federation.

Charles A. Bouman (F'01) received the B.S.E.E. degree from the University of Pennsylvania, Philadelphia, PA, USA, in 1981, the M.S. degree from the University of California at Berkeley, Berkeley, CA, USA, in 1982, and the Ph.D. degree in electrical engineering from Princeton University, Princeton, NJ, USA, in 1989.

He was a Full Staff Member with the MIT Lincoln Laboratory, Cambridge, MA, USA, from 1982 to 1985. He joined as a Faculty Member with Purdue University, West Lafayette, IN, USA, in 1989, where he is currently the Showalter Professor of Electrical and Computer Engineering and Biomedical Engineering. He is also a Founding Co-Director of Purdue's Magnetic Resonance Imaging Facility located in Purdue Research Park. His current research interests include use of statistical image models, multiscale techniques, and fast algorithms in applications, including tomographic reconstruction, medical imaging, and document rendering and acquisition.

Prof. Bouman is a fellow of the American Institute for Medical and Biological Engineering, the Society for Imaging Science and Technology (IS&T), and the SPIE Professional Society. He is a recipient of the IS&T's Raymond C. Bowman Award for outstanding contributions to digital imaging education and research. He received the College of Engineering Engagement/Service Award and the Team Award, and the Electronic Imaging Scientist of the Year Award in 2014. He has been a Purdue University Faculty Scholar. He was the Editor-in-Chief of the IEEE TRANSACTIONS ON IMAGE PROCESSING and a Distinguished Lecturer of the IEEE Signal Processing Society. He is the Vice President of Technical Activities for the IEEE Signal Processing Society. He has been an Associate Editor of the IEEE TRANSACTIONS ON IMAGE PROCESSING and the IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE. He was the Co-Chair of the 2006 SPIE/IS&T Symposium on Electronic Imaging and the 2000 SPIE/IS&T Conference on Visual Communications and Image Processing. He has also been a Vice President of Publications and a member of the Board of Directors for the IS&T Society, and he is the Founder and Co-Chair of the SPIE/IS&T Conference on Computational Imaging.

Anand Raghunathan (F'12) received the B.Tech. degree from IIT Madras, Chennai, India, and the M.A. and Ph.D. degrees from Princeton University, Princeton, NJ, USA.

He was a Senior Research Staff Member with Nippon Electric Company Laboratories America, Princeton, where he led projects on system-on-chip architecture and design methodology. He has held the Gopalakrishnan Visiting Chair with the Department of Computer Science and Engineering, IIT Madras. He is currently a Professor of Electrical and Computer Engineering and the Chair of the VLSI Area with Purdue University, West Lafayette, IN, USA, where he directs research with the Integrated Systems Laboratory. He has co-authored a book, eight book chapters, and over 200 refereed journal and conference papers, and holds 21 U.S. patents. His current research interests include domain-specific architecture, system-on-chip design, computing with post-CMOS devices, and heterogeneous parallel computing.

Prof. Raghunathan has also been a member of the technical program and organizing committees of several leading conferences and workshops, chaired premier IEEE/ACM conferences, such as International Conference on Compilers, Architecture and Synthesis for Embedded Systems, ISLPED, Very Large Scale Integrated Circuits Test Symposium, and VLSI Design, and served on the Editorial Boards of various IEEE and ACM journals in his areas of interest. He is a Golden Core Member of the IEEE Computer Society. He received a Patent of the Year Award and two Technology Commercialization Awards from NEC, and was chosen among the MIT TR35 (top 35 innovators under 35 years across various disciplines of science and technology) in 2006. His publications received eight best paper awards and five best paper nominations. He also received the IEEE Meritorious Service Award and the Outstanding Service Award.