FPGA Acceleration for 3-D Low-Dose Tomographic Reconstruction

Wentai Zhang, *Graduate Student Member, IEEE*, Linjun Qiao, William Hsu, *Member, IEEE*, Yong Cui, Ming Jiang, *Senior Member, IEEE*, and Guojie Luo, *Member, IEEE*

***Abstract*—X-ray computed tomography (CT) is commonly used to obtain vivo images to characterize diseases but results in radiation exposure to patients. Low-dose CT (LDCT) provides CT images of clinical quality with reduced cumulative radiation dose. Iterative image reconstruction methods with effective regularization are used for LDCT but generally require more computing resources and induce higher computational load than the conventional filtered backprojection (FBP) methods. The high computational demand of the iterative reconstruction (IR) with notably increased reconstruction time precludes its routine clinical application. In this work, we focus on the FPGA acceleration of a compute-intensive full IR (full-IR) algorithm based on the Mumford–Shah regularization. At the algorithmic level, we propose a beam-based asynchronous update algorithm to reduce the computational cost and alleviate the conflicts. At the hardware-level, we first present pipeline-friendly optimization for the original algorithm to increase the computation throughput. We then apply the LDCT-specific tiling strategy to improve the data reuse rate. The experimental results show that our implementation takes 8.5 min to reconstruct a typical physical phantom with the image quality comparable with the vendor's result. The FPGA implementation achieves 11.6× throughput against the state-of-the-art GPU version.**

***Index Terms*—Asynchronous parallelism, field-programmable gate array, iterative image reconstruction, low-dose computed tomography (CT).**

I. Introduction

X-RAY computed tomography (CT) provides high-resolution *in vivo* characterization of disease processes. However, an ongoing concern is the amount of cumulative radiation exposure that occurs, particularly when patients undergo a regular series of CT exams, such as the annual screening for lung cancer. With increased CT examinations conducted worldwide, radiation dose reduction in CT has become an active topic in the last decades [1]–[3]. Low-dose CT (LDCT) protocols have been developed to reduce the radiation dose at the cost of a reduced ability to resolve finer structures and a worse signal-to-noise ratio (SNR). To compensate for the decrease in image quality, advanced image reconstruction methods beyond the conventional filtered back-projection (FBP) method are being explored [1], [3], [4]. However, these methods are currently obstructed by the long reconstruction time caused by the high computational demand. From the computational perspective, the difficulty of LDCT is that the projection data from LDCT is of low SNR with reduced or insufficient sampling rates by such as the increased table pitch of current multislice CT scanners [1], [2], [4], [5]. Hence, the conventional FBP method cannot be applied directly. Therefore, iterative reconstruction (IR) with its theoretical foundation in inverse problems [6], [7] was successfully introduced to clinical CT scanners for LDCT around 2010 [1]. IR iteratively improves image quality by using the mathematical models of scanner physics and geometry together with statistical priors about projection data and image to be reconstructed.

IR can be generally classified into two types: 1) hybrid-IR and 2) full-IR [3]. Hybrid-IR iteratively improves the image quality in the image domain and data quality in the projection data domain, but still uses the FBP for image reconstruction to circumvent the forward and backward projection computation. Currently, hybrid-IR is the typical type of IR used by manufacturers [3] as an expedient solution because it has moderate computational workloads and satisfies the timing constraints. Full-IR requires multiple rounds of the forward and backward projection computation and additional repetitive computation induced by priors. It produces better-quality reconstructed image volumes with higher resolution than both FBP and hybrid-IR. However, the high computational demand of the full-IR with notably increased reconstruction time precludes its routine clinical application and makes it unacceptable in emergency [3], [8]–[10].

An IR method can be formulated mathematically by the Bayesian approach or the regularization approach with a properly chosen statistical prior about projection data and image to be reconstructed [11]. By using the regularization approach,

Manuscript received December 30, 2019; revised March 22, 2020 and May 21, 2020; accepted June 25, 2020. Date of publication July 1, 2020; date of current version March 19, 2021. This work was supported in part by the National Natural Science Foundation of China under Grant 61520106004 and Grant 61631001, in part by the Beijing Academy of Artificial Intelligence, and in part by the Beijing Natural Science Foundation under Grant L172004. This article was recommended by Associate Editor W. Zhang. *(Corresponding author: Guojie Luo.)*

Wentai Zhang and Guojie Luo are with the Center for Energy-Efficient Computing and Applications, Peking University, Beijing 100871, China (e-mail: rchardx@pku.edu.cn; gluo@pku.edu.cn).

Linjun Qiao and Ming Jiang are with the Department of Information Science, School of Mathematical Sciences, Peking University, Beijing 100871, China (e-mail: linjun-qiao@pku.edu.cn; ming-jiang@pku.edu.cn).

William Hsu is with the Department of Radiological Sciences, University of California at Los Angeles, Los Angeles, CA 90095 USA (e-mail: whsu@mednet.ucla.edu).

Yong Cui is with the Department of Radiology, Key Laboratory of Carcinogenesis and Translational Research (Ministry of Education), Peking University Cancer Hospital and Institute, Beijing 100142, China (e-mail: yong.cui@outlook.com).

Digital Object Identifier 10.1109/TCAD.2020.3006183

a penalty functional[1] to model the statistical prior is added to the data fidelity functional that models the forward projection process. There are many choices of statistical priors or penalty functionals. Among them, the Mumford–Shah (MS) functional models not only the image smoothness but also the image edge [11], [12]. Hence, IR with the MS functional can avoid computing derivatives across edges unlike the popular total variation [11], [13], and can obtain both the reconstructed image and its segmentation [11], [14]. The MS functional has been successfully applied for image reconstruction problems by peers [14]–[18]. We also implement various versions on CPU, GPU, and FPGA in our previous works [19]–[21].

In this work, we develop an FPGA-based acceleration of the MS-based full-IR algorithm. Its quality and performance are evaluated on the raw 3-D datasets collected using a helical CT scanner. We also demonstrate that it reaches the performance requirement for routine clinical applications of 3-D LDCT. FPGA accelerators have efficient DSP resources for floating arithmetic and on-chip memory with high bandwidth and customized optimization. First, we propose a beam-based asynchronous update method to reduce the computational cost of backward projection and reduce the communication latency among iterations. Second, we apply pipeline-friendly algorithm optimization to the decomposed beam-based streaming computation flow to achieve optimal pipeline performance. Third, we introduce LDCT-specific tiling optimization to reduce the data exchange and increase the efficiency of the on-chip buffer by improving the data reuse rate. We quantitatively analyze the design space and use an automated method to search for the optimal tiling scheme. Finally, we evaluate the implementation with a real CT phantom. The quality of our results is comparable to the vendor's, especially in the line pair regions. It takes 8.5 min to reconstruct a full helical chest CT image sized $874 \times 874 \times 161$. We achieve $11.6\times$ throughput against the state-of-the-art GPU implementation and $1.56\times$ throughput compared with a state-of-the-art FPGA implementation.

Our contributions in this article are threefold as follows.

1) *Asynchronous Beam-Based Parallelism:* In the original IR algorithm, backward projection dominate the execution time (around 90%). Besides, IR algorithms are usually synchronously parallelized, hence with notable communication latency. We apply beam-based optimization to reduce the backward projection to a light-weight operation. Then, by using the asynchronous update, we guarantee the convergence of the reconstruction with reduced synchronous latency.
2) *Pipeline-Friendly Algorithm Optimization:* To make full use of the hardware performance on the FPGA accelerator, we decompose the computational kernels of the beam-based update into a streaming dataflow. In order to achieve the optimal pipeline performance, we apply pipeline-friendly algorithm optimization on the ray tracing algorithm to remove data dependency and balance the workloads among all stages.

[1]The term "functional" is a noun in mathematical analysis and it refers to a mapping from a space X into the real numbers.

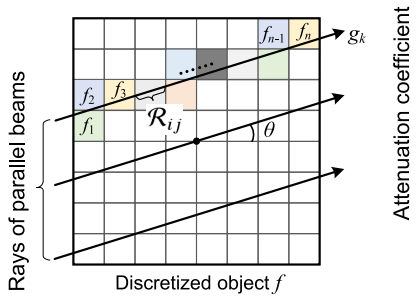


Fig. 1. Discretized X-ray line integral. We illustrate the simple parallel beams to scan a discretized object to generate the projection data g, which is the sum of line integrals of the attenuation coefficient. At any given angle θ, multiple X-rays pass through the object f. Since the parallel beam is used, the number of X-ray beams here is equal to the number of sensors used to receive the actual measurement data. The helical beams have a more complicated projection geometry.

3) *LDCT-Specific Tiling Optimization:* Full-IR 3-D reconstruction of LDCT has large-sized input, output, and intermediate data and is thus memory intensive. We tile the image and projection data to enable a high data reuse rate and ensure the data locality according to the access patterns of full-IR LDCT. We save the memory bandwidth and increase performance by a proper prefetching and buffering strategy.

The outline of this article is as follows. In Section II, we introduce the background of the LDCT. In Section III, we represent our algorithm and software optimization. In Section IV, we describe our hardware implementation and design automation flow. In Section V, we evaluate the implementation with a real CT phantom and compare it with state-of-the-art implementations. Section VI contains the discussions and the comparison to related works. We provide concluding remarks in Section VII.

II. Background and Motivation

A. Background of CT and LDCT

In the following, we will briefly introduce CT's principle, which is necessary to understand our work in this article. Interested readers can refer to [7], [22], and [23] for details. As in Fig. 1, every beam of X-ray penetrates an object as a straight-line and intersects with a number of voxels[2] with X-ray attenuation coefficient $f_1, f_2, \ldots, f_n$. Let the intersection length of the ith beam with voxel j be $\mathcal{R}_{ij}$. The *forward projection* data of the ith beam is then equal to $g_i = \sum_{j=1}^{n} \mathcal{R}_{ij} f_j$, which is the discretization of the Radon transform [24]. For the image reconstruction of CT, projection data of multiple beams from many directions that covers the whole object is needed. There are many kinds of projection geometry, such as parallel beams, fan beams, and cone beams. For IR, the adjoint of R, called the *backward projection* $R^\top$ in literature, is used to transform the measured data in the projection domain to the image domain.

We then introduce the background of 3-D helical scanning used for LDCT in this article. For 3-D clinical imaging, helical

[2]A voxel represents a value on a regular grid in 3-D space.

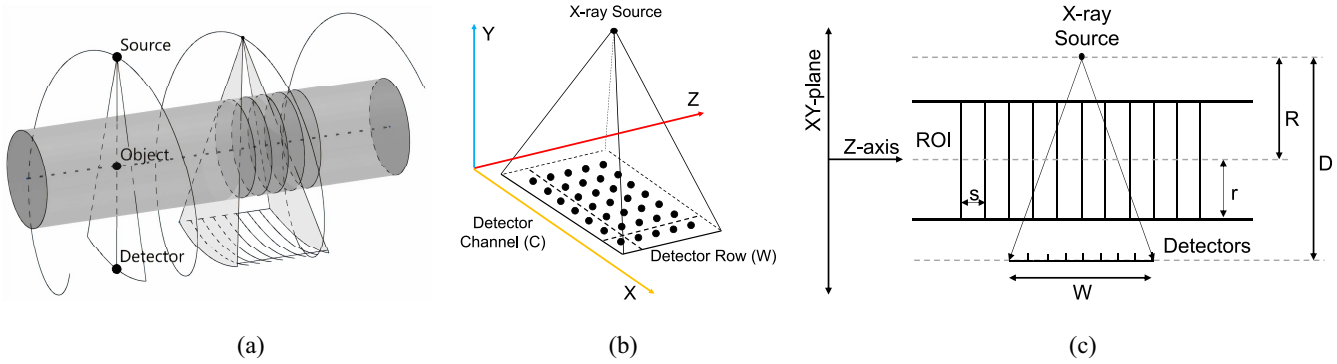


Fig. 2. Helical CT geometry. (a) Diagram of the helical CT geometry, where the trajectory of the X-ray source is a helix relative to an object. (b) Detector rows and channels in the current multirow CT scanners. The X-ray source is moving in the Z direction relative to the object under imaging. The black dots represent the detectors. C is the number of the detector channels, and W is the number of the detector rows. (c) Diagram of the geometry for image reconstruction. ROI means the region of interest. The object under imaging is included in a circle of radius r in each slice. The distance between the source and the center of the rotation R, i.e., the source-to-object distance (SOD). D is the source-to-detector distance (SDD), i.e., the distance between the source and the detectors. R and D are parameters obtained by routine system calibration. s is the slice thickness of the image volume chosen for reconstruction.

CT scanners with multirows of detectors are currently widely used in clinics. A helical CT system consists of one or more X-ray sources and multirows of detectors. The raw projection data are continuously acquired while the patient is moving forward at a constant speed. At the same time, X-ray source and detectors are rotating simultaneously. The scanning mode in helical CT is called a helical scan, because the trajectory of the X-ray source relative to the object is a helix or a spiral, as shown in Fig. 2(a). The pitch of the helix is an important parameter for reducing the dosage for LDCT. The bigger the pitch, the less the dose is imposed. It is possible to reconstruct a CT image with an axial spatial resolution larger than the pitch. A CT image volume consists of many slices of 2-D images, which are the gray circles in Fig. 2(a). Each X-ray beam intersects more than one slices because of the cone-beam geometry. In Fig. 2(b), we illustrate a simplified cone-beam. The detector grids used to receive the attenuation coefficient g have two dimensions: channel C and row W. In Fig. 2(c), we demonstrate important parameters and a slide-view of the helical scanning.

B. Motivation

Typical full-IR reconstruction takes around 30–60 min [3], [8]. The acceleration of full-IR aims to shorten the reconstruction time as much as possible. First, the necessity of the full-IR parallelization originates from the complicated geometry and the enormous total number of X-ray beams in a typical clinical scan. Full-IR needs to process the forward and backward projection in each iteration, which induces high computational cost. Second, the original ray tracing algorithm in the forward and backward projection has data dependency and is not pipeline-friendly. Besides, the unbalanced workloads of all the operations in reconstruction reduce the hardware performance as well. Third, priors or penalty functionals are usually made of stencil memory accesses, which increase the computation load. The stencil access is a mapping where each output depends on the "neighborhood" of inputs, and these inputs are a set of fixed offsets relative to the output position.

We propose three techniques to solve the above problems from three perspectives: 1) parallelism; 2) hardware optimization; and 3) memory. First, we aim to improve the parallelism in full-IR. We propose a beam-based update method to accelerate the computation of the forward and backward projection by reducing them to light-weight versions. Still, parallel processing elements (PEs) conflict with each other when updating the same voxel values in synchronous parallelization. We introduce the asynchronous update and the diminishing step sizes to reduce communication latency. Moreover, we apply pipeline-friendly algorithm optimization to achieve optimal hardware efficiency. We use a state machine for the original ray tracing algorithm to remove the data dependency. Besides, we decompose the computation kernels into a streaming dataflow and balance the workloads among all stages and increase the hardware efficiency. Furthermore, we propose LDCT-specific tiling optimization, because the above two techniques require efficient use of memory. The stencil accesses in them result in multiple voxels accessing at the same time. We tile the image and projection data to increase the data reuse rate. We also propose a prefetching strategy to save the memory bandwidth.

III. FPGA-Friendly Algorithm Design

A. Full-IR With Mumford-Shah Regularization

In this work, we use the MS functional [12] as the regularization term for the full-IR of LDCT aforementioned in Section I. We designate to perform one reconstruction of a certain number of image slices with the projection data of those slices at the CSP, no matter how many channels of detectors there are. The number of slices to be reconstructed at the CSP is determined from the number of detector rows [refer to Fig. 2(c)]. Let the number of detector rows be L. At the CSP, we update the image slices f_l $(l = 1, \ldots, L)$ which are penetrated by X-rays from the CSP. Those image slices f_l $(l = 1, \ldots, L)$ constitute the current subvolume (CSV) $F = [f_1, \ldots, f_L]$ to be updated. Some parts in this subvolume F are not penetrated by X-ray beams at the CSP and will be not updated at this time, but will be updated from other

source positions when we run through the helical trajectory of the X-ray sources. For one CSP, the objective minimization function to update the CSV $F = [f_1, \ldots, f_L]$ and the derived image edges $K = [K_1, \ldots, K_L]$ is as follows:

$$E(F, K) = \sum_{l=1}^{L} \left\{ \|\mathcal{R}f_l - g_l\|^2 + \alpha \int_{\Omega \setminus K_l} |\nabla f_l|^2 \, dx + \beta H(K_l) \right\} \tag{1}$$

where Ω is the image domain; $\|\cdot\|$ is the L^2 norm on the projection data domain, which is $S^1 \times \mathcal{R}$ for 2-D CT or $S^2 \times \mathcal{R}$ for 3-D CT; K_l ($l = 1, \ldots, L$) is the edge set of image f_l; Rf_l is the forward projection of the image slice f_l at the CSP; g_l is the detected projection data of f_l; ∇f_l is the 2-D gradient of f in the plane orthogonal to the scanning direction; and $H(K_l)$ ($l = 1, \ldots, L$) is the 1-D Hausdorff measure of the edge set K_l. The first term in the sum on the right-hand-side of (1) is the fidelity term to force the reconstruction image f_l to have a projection that matches the measured data g_l. The second term is to induce smoothness of the image f_l in everywhere other than the edge set. The third term is to avoid oscillating edges such as fractals and consequently to encourage smooth edges [12]. The second and third terms are together called the MS functional. $\alpha > 0$ and $\beta > 0$ are parameters to control the regularization length of the latter two terms, respectively. In summary, the MS functional models not only image smoothness but also image edge [11], [12]. It also avoids computing derivatives across edges unlike other derivative-based regularization methods, such as the popular total variation [11], [13], and thus can preserve edges during iterative updates.

The edge set is a continuous function and difficult to represent and trace its update in a computer system. In this work, we use the Γ-approximation for the MS functional [25]. The edge set K_l is replaced by a smooth edge indicator function v_l ($l = 1, \ldots, L$) such that $v_l(x) \approx 0$ for $x \in K_l$ and $v_l(x) \approx 1$ for $x \in \Omega \setminus K_l$. We obtain the following Γ-approximation for (1):

$$\Gamma(F, V) = \sum_{l=1}^{L} \left\{ \|\mathcal{R}f_l - g_l\|^2 + \alpha \int_{\Omega} v_l^2 |\nabla f_l|^2 \, dx + \beta \int_{\Omega} \left(\varepsilon |\nabla v_l|^2 + \frac{(1 - v_l)^2}{4\varepsilon} \right) dx \right\} \tag{2}$$

where $V = [v_1, \ldots, v_L]$, for $\varepsilon > 0$. When $\varepsilon \to 0+$, $\Gamma(\cdot, \cdot)$ converges to $E(\cdot, \cdot)$ in the sense of Γ-convergence, and consequently the minimizer (F, V) of $\Gamma(\cdot, \cdot)$ is an approximated minimizer of $E(\cdot, \cdot)$ [14], [25], which provides an update of the full-IR at the CSP. Theoretically, ε is twice of the edge width and is usually chosen by experiments at implementation [14], [17], [19]–[21], [25].

In order to minimize $\Gamma(f, v)$, it is straightforward to use the incremental gradient descent method [26]. They alternatively conduct the gradient descent algorithm for each slice at the CSP and f_l, v_l at each slice, and then iterate over all source positions. At each step, both approaches will be required to compute the gradient of the entire slice f_l and v_l, and also the forward projection operator $\mathcal{R}$ and backward projection operator $\mathcal{R}^\top$, i.e., the transpose of $\mathcal{R}$. However, either approach has poor parallelism and induces high memory and communication expenses because the image and projection data are variables of large sizes, and gradients involved are difficult for parallel optimization.

B. Asynchronous Beam-Based Parallelism

1) Beam-Based Optimization: In this section, we present a decomposition method to reduce the computation expenses aforementioned in the last paragraph of the previous section. Along each X-ray beam, the forward projection $\mathcal{R}$ performs the line integral. For the ith beam at the CSP, let $\mathcal{R}_{(i)}$ be the nonzero entries of $\{\mathcal{R}_{ij}\}$, and $f_{(i)}$ and $v_{(i)}$ be values of the image slice and edge indicator corresponding to $\mathcal{R}_{(i)}$, respectively. Our approach is not only based on incremental gradient decent but also mixed with the beam-based decomposition of F and V. For the ith beam, the gradient $\Gamma(F, V)$ with respect to $f_{(i)}$ and $v_{(i)}$ is calculated by

$$\nabla_{f_{(i)}} \Gamma(f, v) = 2(\mathcal{R}_{(i)} f_{(i)} - g_i)\mathcal{R}_{(i)} - 2\alpha \div (v^2 \nabla f)_{(i)} \tag{3}$$

$$\nabla_{v_{(i)}} \Gamma(f, v) = 2\alpha |\nabla f_{(i)}|^2 v + \frac{\beta}{2\varepsilon}(v_{(i)} - 1) - 2\beta\varepsilon\Delta v_{(i)} \tag{4}$$

where Δ is the 2-D Laplacian. It can be seen from (3) and (4) that the common usage of the backward projection $\mathcal{R}^\top$ for IR methods is reduced to a light-weight operation $\mathcal{R}_{(i)}$ with the beam-based fine-grained decomposition. This reduces the computation cost of forward and backward projection greatly by using separated line integral calculations. In this work, we use the fast ray-tracing technique [27] to compute the forward projection $\mathcal{R}_{(i)} f_{(i)}$, though we develop a pipeline-friendly version in our implementation (see Section IV-B). The forward projection can be written as

$$\mathcal{R}_{(i)} f_{(i)} = \sum_k f_{I_k} w_k \tag{5}$$

where w_k is the intersection length between the ith beam and the voxel; I_k is the index of the kth voxel along the beam; f_{I_k} is the density weights of the voxel (the value of f at index I_K).

2) Asynchronous Parallelism With Diminishing Step Sizes: In order to encourage the parallelism and reduce communication latency, we further introduce the asynchronous update with diminishing step sizes. "Asynchronous update" means that multiple beams are computed in parallel to update the CSV (F, V) without waiting the complete computation of each other in the inner loop. When those beams intersect and pass through the same voxels, this incurs conflict updates. These conflict updates at the intersections of beams can be resolved by using diminishing relaxation coefficients on step sizes. The diminishing relaxation coefficients (step sizes) guarantee the convergence of the asynchronously parallel algorithm [28], [29], as follows:

$$\sum_k \lambda_k = +\infty, \lim_k \lambda_k = 0, \lambda_k > 0. \tag{6}$$

The relaxation coefficient μ_k satisfy the same diminishing condition. A typical choice of the diminishing condition is

$$\lambda_k = \frac{A}{B + k} \tag{7}$$

Algorithm 1 Asynchronous Parallel Beam-Based Update at the CSP

Input: measured projection data g at the CSP
Output: CSV F, edge indicators V updated at the CSP

$f^0 = 0$
$v^0 = 1$
Initialize step size λ_k and μ_k by (7)
for k = 1 to #iterations **do** ▷ outermost loop
  for p = 1 to #sources **do** ▷ outer loop
    $(f^k, v^k) = (f^{k-1}, v^{k-1})$
    for all processing elements for beam updating **do** ▷ parallelization loop
      for i-th beam of this processing element **do** ▷ inner loop
        Fetch $(f_{(i)}, v_{(i)})$ as a local copy
        $f_{(i)} = f_{(i)} - \lambda_k \nabla_{f_{(i)}} \Gamma(f_{(i)}, v_{(i)})$
        $v_{(i)} = v_{(i)} - \mu_k \nabla_{v_{(i)}} \Gamma(f_{(i)}, v_{(i)})$
        Commit $(f_{(i)}, v_{(i)})$ to (f^k, v^k)
      end for
    end for
  end for
  Reduce step size λ_k and μ_k by (7). ▷ for the convergence
end for

where A and B are positive constants. In Section V-B, we will explain how to choose proper A and B for λ_k and μ_k. Algorithm 1 presents the asynchronous parallel version of our beam-based algorithm.

In summary, at the algorithmic level, our full-IR algorithm is to iteratively execute Algorithm 1 by using each X-ray source along the trajectory as the CSP. The parallelization loop in Algorithm 1 can also be run synchronously with simplified relaxation coefficients update, though it will increase the latency. Furthermore, at the hardware level, we need to optimize the memory and bandwidth usage by carefully scheduling the asynchronous parallel beam-based updates at the inner loop of Algorithm 1 at each CSP, and balance the number of parallel units and hardware resource. The "inner loop" is actually implemented by the PE and will be furthermore introduced in Section IV-B as the PE.

IV. FPGA Accelerator Design

A. Design Flow and Overall Architecture

We use Xilinx SDx Environment [30] to program FPGA devices. We first write the C/C++ version of Algorithm 1 for correctness verification. Second, we synthesize the high-level source code into the hardware design. We use pipelining, streaming, and parallel modules to improve the performance of the computation modules. Finally, we place and route the hardware design to examine the correctness at the physical level. At this step, we will verify the clock frequency and resource utilization. After the implementation of placement and routing, the bitstream file will be generated.

In Fig. 3, we illustrate the overall architecture of the hardware design. The raw data obtained by the CT scanner will

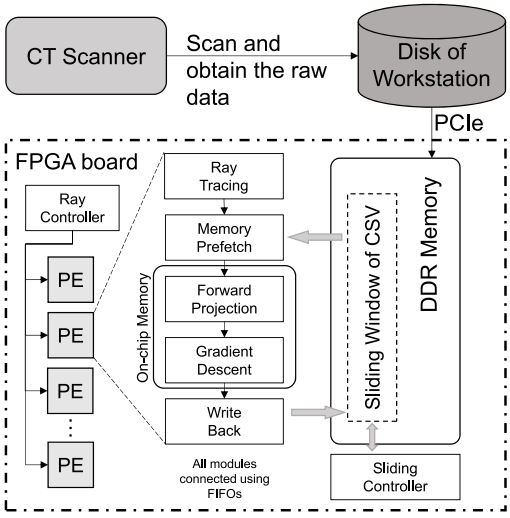


Fig. 3. Overall architecture of the hardware design. The CT scanner obtains the raw data first and sends them to the workstation with the FPGA board. Each PE consists of five stages. Two of the stages work in the on-chip memory (BRAM). PEs access the DDR memory by using the sliding window.

be transfered to FPGA's DDR memory through high-speed links such as PCIe. We use a ray controller to scheduling the execution order and dispatch computational tasks to PEs. Each of PEs has five basic stages: starting with "ray tracing" and ending with "write back." Two of them—forward projection and gradient descent will compute in the on-chip memory. All the stages are connected using FIFOs. We also use a sliding window as the buffering strategy, as described in Section IV-C1.

B. Pipeline-Friendly Algorithm Optimization

We use the straightforward algorithm described at the end of Section III-A, and its computational components are shown in Fig. 4(a). We can see that different modules have different magnitudes in terms of time complexity. For an efficient pipeline, each stage should be expected with the same level of time complexity, or there will be inefficient bubbles and stalls causing performance hazard. This is one reason why we choose the beam-based optimization approach. The computation flow of the inner loop body in Algorithm 1 is shown in Fig. 4(b). There are five parts in detail as follows.

1) *Ray Tracing:* Perform line integral algorithm to generate the voxel indices and weights for related rays $\mathcal{R}_i$.
2) *Memory Prefetch:* Fetch $(f_{(i)}, v_{(i)})$ from the main memory.
3) *Forward Projection:* Perform forward projection $\mathcal{R}_{(i)} f_{(i)}$ and compute $\mathcal{R}_{(i)} f_{(i)} - g_i$.
4) *Gradients Descent:* Compute the gradients $\nabla_{f_{(i)}} \Gamma(f_{(i)}, v_{(i)})$ and $\nabla_{v_{(i)}} \Gamma(f_{(i)}, v_{(i)})$, and update the local copy of $(f_{(i)}, v_{(i)})$.
5) *Write-Back:* Commit local copy back to the main memory without waiting for the completion of updates at other beams.

We can see that both fetch and commit parts are memory-related, while the other three parts are computation-related. The ray tracing part takes the source and destination of the

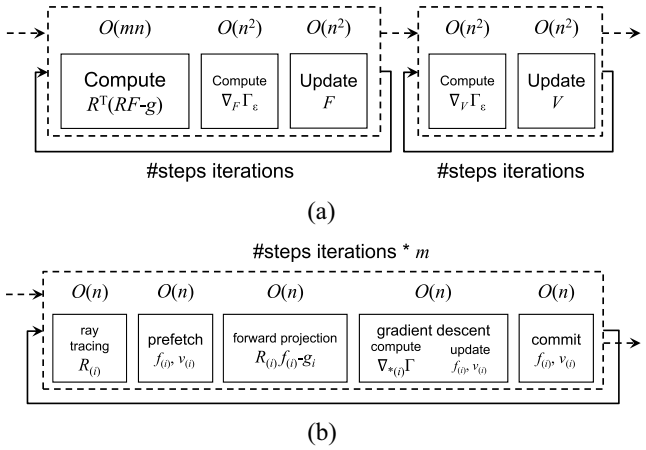


Fig. 4. Raw computation flow and pipelined flow. (a) Original flow of the typical IR algorithm. m is the total number of the projections and n is the average length of the X-ray beam. We can see that the first component has different time complexity from others, and other components have a squared complexity. (b) Pipelined flow after applying beam-based update and asynchronous update. After pipelining optimization, all the components have the same linear time complexity $O(\#\text{steps}\cdot mn)$. This balanced design will increase the hardware occupancy compared to the original flow.

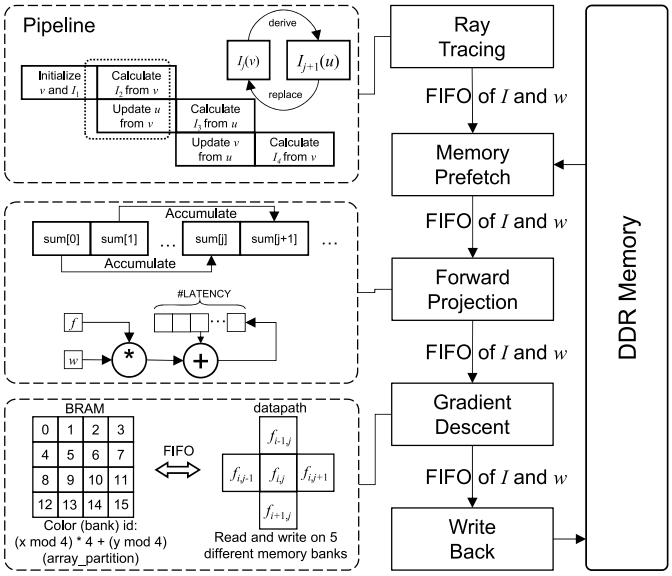


Fig. 5. Details of PEs. All five stages of a PE are connected using FIFOs. We apply pipeline-friendly optimization to the original ray tracing algorithm to remove data dependency. For the forward projection and write-back, we use well-designed techniques to achieve the best performance.

ray beam as inputs and generates the indices and weights. In the line integral algorithm, we used [27], there are many floating-point numbers multiplication and division operations. The forward projection $\mathcal{R}_{(i)}f_{(i)}$ and computation of $\mathcal{R}_{(i)}f_{(i)}-g_i$ is computed as in (5). Equation (5) is a summation of $f_{I_k}w_k$.

We will apply hardware optimization on this streaming dataflow to achieve optimal FPGA-based design. Memory prefetch and write-back are memory-related steps and they can be pipelined with simple directive clauses "#pragma HLS pipeline." Compared to memory-related stages, the other three computational stages cannot be pipelined efficiently using a straightforward directive clause.

For ray tracing, the original algorithm [27] produces the voxel indices and intersection lengths. The original algorithm cannot be pipelined efficiently mainly because data dependency prevents the optimal initial interval of the loop. In 3-D, the original algorithm generates a list of the passed voxels $I_1, I_2, \ldots, I_K$ and the weights $w_1, w_2, \ldots, w_K$ [as in (5)]. The calculation of I_{j+1} and w_{j+1} depends on the latest results at I_j. Taking I as an example, we formulate the pipeline of the ray tracing in 3-D as a state machine with two state variables: one is for updating I_{j+1} using results at I_j (update state); the other is for replacing I_j with I_{j+1} (replace state). In order to achieve the best initial interval, we forbid simultaneous read and write for a variable in one state in this context. By using two variables for I_j and I_{j+1}, respectively, I_j is written in the replace state and read in the update state; I_{j+1} is written in the update state and read in the replace state. Thus, the variable dependency between I_j and I_{j+1} is removed for the best initial interval of the loop. In a hardware implementation, we use two temporary variables v and u to store the update and replace state, as shown in Fig. 5. In Fig. 5, we also illustrate the optimized pipeline of ray tracing.

The forward projection is a summation process. In hardware design, when the latency of the multiplication and addition is determined, we can use a latency-aware pipelining method to obtain the initial interval of 1, shown as below.

```
for (int i=0; i<T; ++i)
#pragma HLS pipeline
  sum[i+LATENCY]=sum[i]+f[i]*weight[i];
```

In this code, "LATENCY" must be greater than the actual latency of the multiplication "f[i]*weight[i]." T is the total number of voxels in the ray. #pragma HLS pipeline is used to pipeline the loop body. In Xilinx Vivado HLS, the pipelining directive will always achieve the optimal initial interval available, unless one desired initiation interval other than 1 is given. We use LATENCY local accumulators and one adder for this loop. The final execution time will be T cycles and the result is the summation of the last LATENCY elements in the array. In Fig. 5, we illustrate the computation process of forward projection. It can be implemented using a buffer of LATENCY length and an accumulator. This buffer hides the latency of multiplication. A reduction tree is also viable for this accumulation but requires more adders.

The gradient descent part has an initial interval greater than 1 if only simple pragmas are used because of the stencil accesses in the regularization terms. In the computation of (2)–(4), there are calculus operations, such as ∇f, ∇v, and $\div(v^2\nabla f)$. These operations need to access a voxel and its four surrounding neighbors. If the memory has not enough ports for the stencil accesses, the initial interval of the pipelining will be greater than 1. We use coloring to partition the CSV into multiple banks. We partition the BRAM cyclically using 16 colors (16 banks), and for a pixel (x, y) in a 2-D slice, its color is

$$\text{color of } (x, y) \leftarrow (x \bmod 4)\cdot 4 + (y \bmod 4). \tag{8}$$

In Xilinx Vivado SDx and HLS, we can use "#pragma HLS array_partition" to implement the multibank BRAM. In this way, any five voxels of this pattern have different colors and banks. The gradient descent is performed in BRAM and each PE has its own local buffer, as shown in Fig. 5. The data in BRAM are transferred using FIFOs to the data-path of gradient descent. Hence, stencil readings can be done in one single hardware cycle. In Fig. 5, we illustrate the details of the implementation of gradient descent.

C. Memory Optimization

1) Slice-Driven Reconstruction: In previous works [19], [21], [31], either voxel-driven or ray-driven execution order is proposed for IR algorithms. For 3-D reconstruction, voxel-driven algorithms have a bad data reuse rate since the accesses of the projection data are out of order. In addition to this, we do not use voxel-driven reconstruction for the following reason. Many X-ray beams are passing through one voxel, and the voxel-driven iteration is to update the CT volume voxel by voxel. Many forward projections are computed to update only one voxel at each step. All the computation of these forward projections at other voxels than the current voxel is wasted. For the ray-driven algorithms, although they can fully utilize the locality of measured projection data, the memory accesses of the image and edge indicator are out of order and inefficient if this strategy is simply applied to Algorithm 1. Hence, we need to find how to improve the memory usage for Algorithm 1.

We propose a slice-driven method to implement the beam-based asynchronous Algorithm 1. For 3-D CT, the projection data size is enormous. In our experiments, the entire set of the projection data is of 10^8 size. Hence, the data locality of the projection data is critical for the memory performance for 3-D CT. Our solution is to consider the data locality of the projection data first. On the other side, we need to consider the data locality of the image volume. Our slice-driven method is to enumerate the CSPs along the trajectory path. At each CSP, the CSV is determined by the geometric parameters of the CT scanner and the image resolution. The reconstruction is applied slice by slice for each slice in the CSV. Please refer to Section V-B for examples of choosing the CSV. For a segment of consecutive CSPs, their corresponding CSVs also form a segment of consecutive slices. Those CSVs constitute a sliding window covering the segment of consecutive slices, as shown in Fig. 6. In simple words, the sliding window is a FIFO of slices.

2) Tiling: The image volume and measured data can be stored in a multilevel structure as we use the slice-driven method. At the first level, we use BRAM as the high-speed cache for the voxels under current reconstruction. If there are M PEs for beam-based IR algorithms, and the average number of voxels that a ray crosses is Q, the total number of voxels in the first level should be $M \cdot Q$. At the second level, we need to store the sliding window. The total amount of memory usage is determined by the width L of the sliding window. If the image slice size, i.e., the XY-plane size, is $N_x \cdot N_y$ in pixels, and we use a CSV or sliding window of L image slices, the memory consumption is $N_x \cdot N_y \cdot L$.

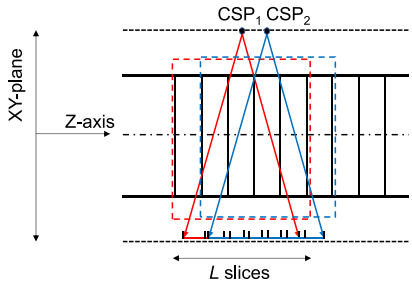


Fig. 6. Demonstration of the sliding window. CSP_1 and CSP_2 correspond to the two different windows (red and blue windows, respectively). Every window needs to cover at least the range of the CSV [Fig. 2(c)]. When the new projection data are introduced in the reconstruction, the window will move along the z-axis.

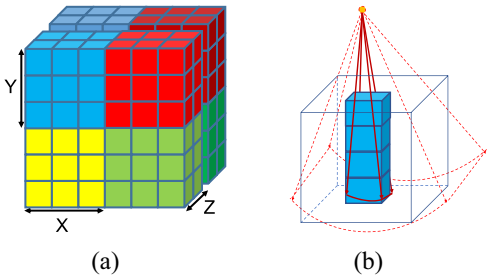


Fig. 7. (a) Demonstration of LDCT-specific tiling. Different colors represent different tiles. Our tiles are of the shape (x, y, z). (b) Block of the measured projection data and its related tiles. Solid red lines represent a group of X-ray beams and the blue tiles are its in-use cache.

One way to improve the memory efficiency is to increase the data reuse rate and decrease the memory I/O requests between DRAM and BRAM. Data reuse rate can be formulated as the size of the CSV divided by the total number of the memory I/O requests for applying Algorithm 1 for this CSV

$$\text{Data reuse rate} = \frac{N_x \cdot N_y \cdot L}{\#\text{Mem I/O}}. \tag{9}$$

The PEs will fetch and buffer the image volume in the on-chip BRAM from DRAM in execution. A higher data reuse rate means that we use less bandwidth of DRAM. We use LDCT-specific tiling optimization to improve the data reuse rate.

We denote (x, y, z) as the shape of the tile, as shown in Fig. 7(a). When the sliding window is $N_x \cdot N_y \cdot L$, the total number of tiles is $\lceil N_x/x \rceil \cdot \lceil N_y/y \rceil \cdot \lceil L/z \rceil$. The shape and number of the tiles affect the data reuse rate and the final memory performance. Our objective is to find out the best (x, y, z) options to optimize the data reuse rate. Because the helical scanning is circular symmetric with respect to the center of each image slice, we set $x = y$ to reduce the dimension of the design space. The search space is then reduced to $1 \le x \le N_x$ and $1 \le z \le L$. We can enumerate each possible value of x, z, and record the total number of memory I/O requests and the maximal local memory used. The maximal local memory used helps prevent the implementation from exceeding the on-chip memory limit.

In Fig. 8, we present our search result for $z = 1$, using the experimental setting in Section V-C, where $N_x = 874$. In this figure, the memory I/O requests and local memory

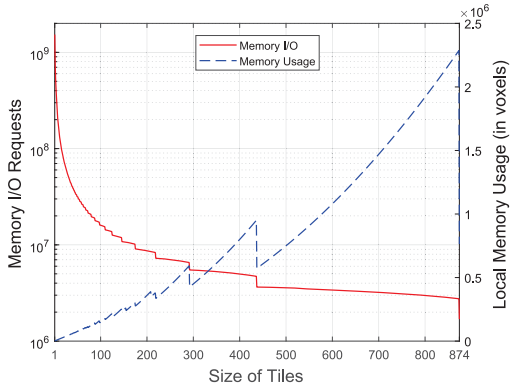


Fig. 8. Searching for the best tiling shape and option. The solid line is the total amount of the data exchange. The dashed line is the memory usage (in voxels).

usage are demonstrated as the solid and dashed lines, respectively. The fluctuations are because N_x is not divided by some x exactly. The memory I/O requests are decreasing when the tile size increases. Hence, an optimal tiling can be found given reconstruction parameters and hardware resources. Because CT scanners are operated with the same scanning parameters by clinical protocols, the optimal tiling we obtained can be reused for routine clinical usage.

Furthermore, we reschedule the execution order of beams for Algorithm 1 to further improve the performance because the order of beams affects the performance of memory and asynchronous parallelism. The original execution order of the X-ray beams is a two-level nested loop with (C, W) as the upper bounds. For a tiled sliding window with our method in the previous section, we aim to schedule the execution order of all the beams involved to improve data reuse rate since the data reuse rate is already improved with our LDCT-specific tiling optimization. Moreover, each X-ray beam should be applied only once because of the massive number of beams in 3-D CT. The data reuse rate here is defined as the ratio of the total number of the in-use X-ray beams divided by the number of the cached tiles. Our scheduling regroups the X-ray beams and each group of X-rays beams shares as many tiles as possible.

We regroup the original loops using execution blocks (C_b, W_b) [in solid red Fig. 7(b)] smaller than the tiled image to improve the data reuse rate, as shown below.

```
for (int cb = 0; cb<C/C_b; ++cb)
  for (int wb = 0; wb<W/W_b; ++wb)
    for (int c = cb*C_b; c<cb*C_b+C_b; ++c)
      for (int w = wb*W_b; w<wb*W_b+W_b; ++w)
        process_beam(c,w);
```

In Fig. 7(b), we represent an illustration of the blocked projection. This block of the X-ray beams will be covered by the tiles marked in the image volume. The only problem here is how to decide the (C_b, W_b) for promising data reuse rate. We maintain the in-use tiles satisfying the local BRAM capacity and maximize the data reuse rate. C_b, W_b depend on the parameters of CT scanners. C_b is related to the spacing between the detector channels. We can find a maximum C_b and ensure this block of X-ray beams not exceed the local cache limitation. W_b is related to the spacing between the detector rows and can be calculated similarly. We can enumerate them for the best data locality because its design space is not large.

3) Design Space Exploration: In previous sections, we have to determine the values of several parameters for our design: x, z, W_b, and C_b. We will find the optimal values of them to generate efficient designs. For each possible combination of (x, z, W_b, C_b), we will examine the data reuse rate and (9) has already defined the calculation of data reuse rate. The input of design space exploration is the CT parameters. The output is the optimal (x, z, W_b, C_b).

We use a two-step searching strategy. We first eumerate the combinations of (x, z) and then (W_b, C_b). In the end, we record the one with the best data reuse rate. x, z, W_b, and C_b are bounded by N_x, L, W, and C, respectively. To calculate the data reuse rate by (9), we have to run one iteration of the reconstruction program with only ray tracing to collect the information. Taking the parameters in Section V-B as the example, there are $L = 24$ and upper bound of C_b is 50 [according to (10)]. Besides, we only need to enumerate the integral point for $\lceil N_x/x \rceil$. When $N_x = 874$, the validate x can be one of $874, 437, 292, 219, 175, 146, \ldots, 30$ (30 possible values) Hence, the search space of (x, z) is $30 \cdot 24 = 720$; the search space of (W_b, C_b) is $50 \cdot 24 = 1200$. One iteration of ray tracing takes around 58 s on a 16-core workstation. Therefore, the total design space exploration on a CPU-based workstation for dataset ACR464 (see Section V-C) will take around $(720 + 1200) \cdot 58 \approx 31$ h. In real practice, we can also prune some suboptimal combinations by intermediately checking the data reuse rate without completing the iteration.

V. Experiments and Results

A. Environment Setup

We select Xilinx Zynq UltraScale+ MPSoC ZCU102 [32] as our heterogeneous platform for our parallel framework of the full-IR reconstruction. ZCU102 has Xilinx XCZU9EG-2FFVB1156 FPGA with 2520 DSP slices. The baseline methods are executed on a multicore server with two Intel Xeon CPU E5-2650 CPUs and 64-GB main memory. Each E5-2650 has ten physical cores running at 2.30 GHz.

We use the parameters from SOMATOM Definition AS (Siemens Healthcare GmbH, Erlangen, and Germany) CT scanner to conduct the physical study. The pitch value is 19.2 mm. The SOD R is 595.0 mm. The SDD D is 1085.6 mm. The spacing between the detector rows is 2.052 394 65 mm. The fan angle is 50°, and then the detector channel spacing is around 1.287 mm. It has 16 detector rows and 736 detector channels. Those parameters are available from the literature [33].

We use the structural similarity index (SSIM) [34] as the image quality criteria to evaluate the quality of reconstructed images. For the physical study, we further use the contrast-to-noise ratio (CNR) [10], [35] to evaluate specifically the image quality of regions of line pairs.

B. Reconstruction Parameters and Resource Allocation

In this section, we will demonstrate the parameter selection in our FPGA implementation. The helical scan mode is described in Section V-A. For image reconstruction, the x-axis and y-axis spacings are both 0.5859 mm; the image slice size is set to 874×874 in pixels; by default, dimensions of the image volume are in pixels/voxels; the slice thickness s is 1.5 mm, to be consistent with the physical experiment in Section V-C to compare our results with those from the vendor. We then determine the design parameters in our FPGA implementation.

First, the width of the sliding window depends on the detector rows. The z-axis distance for a cone-beam is $2.05239465 \times 16 = 32.8$ mm. Since the slice thickness is 1.5 mm, a cone-beam covers $32.8/1.5 = 21.9$ slices. Thus, we choose 24 as the width of the sliding window to cover enough slices for stability with our sliding window approach.

Second, the shape of the tiles relies on the on-chip memory capacity of our FPGA chip. ZCU102 has 1824 BRAM_18 K resources and therefore has 32-Mb BRAM. We will use 16-b fixed point numbers for the image and 8-b fixed point numbers for the edge indicator. We search the design space, as described in Section IV-C2. We choose $z = 6$ and $x = 110$ as the tile shape, which partitions the image slice into 8×8 and consumes at most around 2.91 MB BRAM during runtime.

Third, we decide the values of C_b and W_b. Since $x = 110$ and the fan angle is 50°, W_b should be at max as

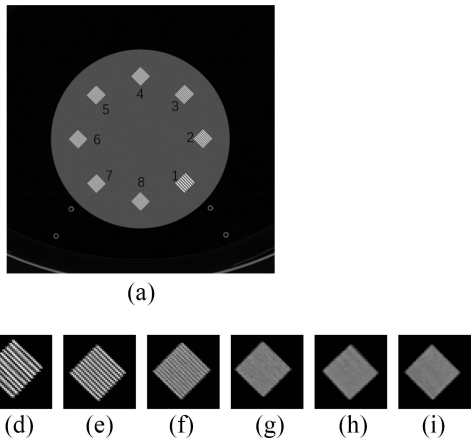
$$110 \times 0.5859/\left(\text{SDD} \cdot \tan\left(\frac{50^\circ}{736}\right)\right) = 110 \times 0.5859/1.287 = 50. \quad (10)$$

C_b can be as large as the width of the sliding window. However, there may be insufficient hardware resources for so many X-ray beams processed at the same time. We will examine the resource usage of one single PE. Table I shows the resource utilization. The first part of the table represents the resource usage of one single PE. DSP and LUT are in shortage if more PEs are implemented. The resource utilization cannot be too high, or probably the design cannot be synthesized, placed, or routed properly. Therefore, we can parallel at most 48 PEs at once on ZCU102, which will consume 91.4% of the DSP resources. We set $W_b = 48$ and $C_b = 1$, and 48 PEs will access among the cached tiles and have efficient data locality. The reason for $W_b = 48$ is that $736/48 = 15.3$ almost equals to the row blocks (16 detector rows and $W_b = 1$). The total resource usage for the 48 PEs is shown in Table I. We can see that BRAM, DSP, and LUT are almost used up.

At the algorithm level, there are also some reconstruction parameters to determine for applications. The first is about the regularization parameters α and β. We can compare the qualities of reconstructed images with different choices of parameters by their SSIM values against the baseline because we have the image from the full-dose scan as the baseline. Hence, the optimal set of parameters for reconstructed images of the best quality could be found by searching the space of regularization parameters with the SSIM as the criterion. The same approach can be applied to determine other parameters, including the step sizes λ_k and μ_k. In this work, we choose $\alpha = 0.1$, $\beta = 0.05$. For step sizes, we choose $A = 1/500, B = 2$ for λ_k and $A = 1/2000, B = 2.5$ for μ_k.

TABLE I
RESOURCE UTILIZATION FOR A SINGLE PE AND 48 PES

	Components	BRAM_18K	DSP48E	FF	LUT
Single PE	ray tracing	0	19	180	782
	computation	4	29	3339	3124
	memory r/w	0	0	54	405
	Total	4 (0.22%)	48 (1.90%)	3573 (0.65%)	4311 (1.57%)
Overall	ray tracing	0	912	8640	37536
	computation	192	1392	160272	149952
	memory r/w	0	0	2592	19440
	local cache	1323	0	28	210
	Total	1515 (83.1%)	2304 (91.4%)	171532 (31.3%)	207138 (75.6%)
	Avaible	1824	2520	548160	274080



(a)

(b) (c) (d) (e) (f) (g) (h) (i)

Fig. 9. (a) Example slice of vendor's reconstructed images; display window [216,3200]; current 215 mA. No. 1–8 represent line pairs of different widths. (b)–(i) Vendor's results of line pairs with different widths. We show the corresponding area of these line pairs in (a).

C. Quality of Results

In this section, we will evaluate our framework's quality of results (QoRs) by using CT ACR 464 phantom (Sun Nuclear Headquarters, Melbourne, FL, USA), built for routine CT scanner evaluation. We compare the vendor result at the full dose level (215 mA) and our result at a reduced dose level (50 mA), i.e., with 23% of the full dose. We reconstructed raw data acquired at different dose levels using CT ACR 464 phantom on the SOMATOM Definition AS. The raw projection data have 14524 samples in total. Every 360°, the CT scanner produces 1152 samples. Therefore, the total number of slices is $14524/1152 \times 19.2/1.5 = 161$. The tube voltage of these projections is fixed at 120 KV and the current is set to 215, 150, 100, or 50 mA for performing LDCT imaging.

Fig. 9(a) shows one typical slice from the vendor's result of the CT ACR 464 phantom at the dose of 215 mA, reconstructed using the vendors proprietary algorithm. The phantom contains eight high contrast resolution patterns. Each of them has multiple line pairs per centimeter. From No. 1–8, they have 4, 5, 6, 7, 8, 9, 10, and 12 line pairs per centimeter, respectively. These line pairs are designed to evaluate the spatial and contrast resolution of the reconstructed images. Fig. 9(b)–(i) show the detail of these line pairs with different width. The

TABLE II
IMAGE QUALITY BY SSIM AT DIFFERENT DOSES

Data set	method	mean	max	min	std
215mA	ours	0.991	0.996	0.942	0.0611
	vendor	1.000	1.000	1.000	0.0000
150mA	ours	0.981	0.989	0.965	0.0077
	vendor	0.997	0.998	0.995	0.0010
100mA	ours	0.972	0.990	0.901	0.0258
	vendor	0.997	0.998	0.995	0.0011
50mA	ours	0.976	0.989	0.920	0.0188
	vendor	0.982	0.998	0.995	0.0298

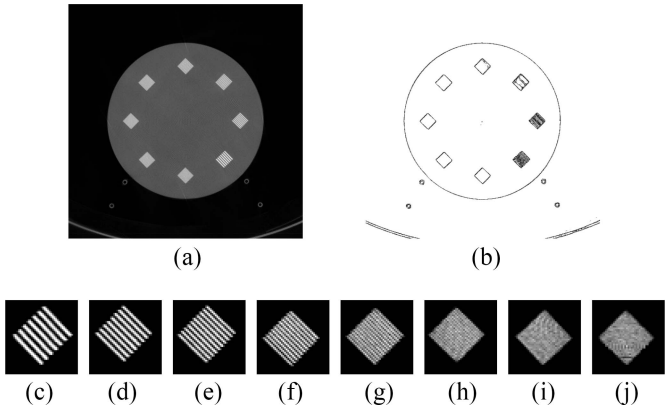


Fig. 10. (a) 29th slice of the reconstructed image. (b) Segmentation of the 29th slice. (c)–(f) Represent line pairs 1–4. (g)–(j) Represent line pairs 5–8. (Display window [216, 3520]. Current 100mA.)

images from the vendor at this high dose of 215 mA are used as the baseline in the following comparison of image quality. We first evaluate the correctness of our implementation with the high dose of 215 mA, and obtain images of the same quality as the vendor's, as shown in the first line of Table II.

Then we perform the experiment with different dose levels with current 215, 150, 100, or 50 mA at fixed voltage 120 KV. Table II shows the image quality comparison in terms of SSIM at different dose levels in ten iterations. It can be seen that our results are comparable to the vendor's at all dose levels in the experiment.

Fig. 10 shows our reconstructed images and the details of the line-pair region of 100 mA. In order to compare the spatial and contrast resolution of our reconstructed images and the vendor's results, we present the line profile of different patterns of line pairs in Fig. 11. From line pairs 1–6, both our and vendor's results are visually the same. For the other two line pairs, both our and vendor's results cannot distinguish the lines.

In order to evaluate the reconstruction quality quantitatively, we use the CNR to evaluate the image quality for regions of line pairs. We use the following CNR $= A/\sigma_N$ [10], [35], where A is the amplitude of the signal and σ_N is the standard deviation of the noise. Fig. 12 shows the CNRs of the regions of line pairs. We can see that our results have higher CNR compared with the vendor results at line pairs 6–8, and these are very thin line pairs. An interesting observation is that in the vendor results, line pairs 1 and 2 have very similar CNR values; line pairs 3–6 have almost the same CNR values; and line pairs 7 and 8 are nearly the same. We guess that the vendor

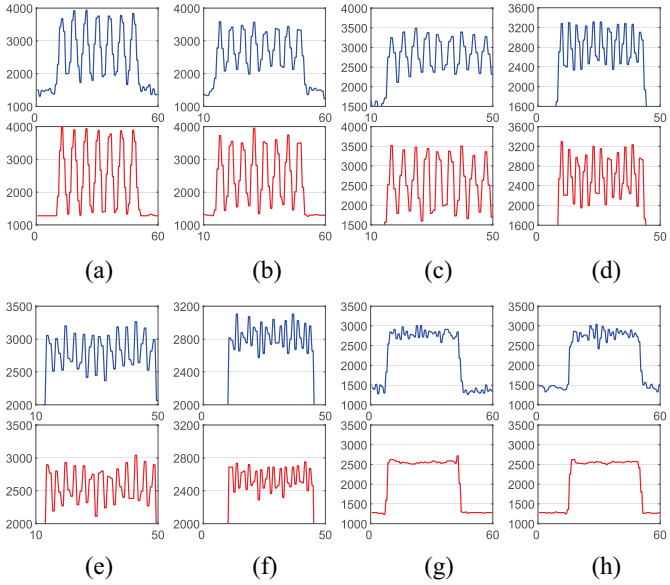


Fig. 11. Line profile of these line pairs shown in Fig. 10. The blue lines represent our results. The red lines represent the vendor's results.

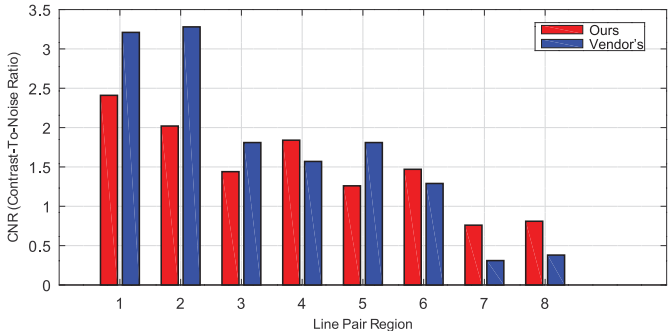


Fig. 12. CNR of the different line pair regions [Fig. 9(a)]. Current 100 mA.

use different filters to enhance the reconstruction quality for certain patterns, especially line pairs 1 and 2.

In summary, we conclude that our framework achieves comparable spatial and contrast resolution compared with the vendor's results, verified by our radiologist co-authors.

D. Performance

In this section, we will report our implementation's performance, and compare with other works. Other works include various platforms such as CPUs, GPUs, and FPGAs.

1) Beam-Based Asynchronous Update Improvement: The baseline in this section is the reconstruction of a 2-D Shepp-Logan phantom with an image size of 512×512 and parallel beams. The algorithm used in the baseline is the synchronous parallel version of Algorithm 1. For Algorithm 1, we evaluate the performance improvement with the same phantom as in the baseline. Table III shows the performance of the baseline version and the asynchronous versions. The "time in seconds" in Table III is the time for one iteration over all sources. Async-X means the asynchronous version with X threads. When using only one thread, our beam-based asynchronous method achieves 2.82× speed-up compared with the baseline

TABLE III
PERFORMANCE IMPROVEMENT THROUGH BEAM-BASED ASYNCHRONOUS UPDATE

Method	baseline	async-1	async-4	async-8	async-16
Time (seconds)	5.6	1.98	0.59	0.35	0.21
Speedup	1.0	2.8	9.5	16.0	26.7

TABLE IV
PERFORMANCE OF DIFFERENT PARTS

Impl.	Parts	Time (seconds)	Percentage
baseline	forward projection	3.4611	64.4%
	backward projection	1.8493	34.4%
	gradient descent	0.0421	0.8%
	memory I/O	0.0199	0.4%
async-2D	ray tracing	0.1814	8.8%
	forward projection	0.0514	2.5%
	gradient descent	1.2625	61.0%
	memory prefetch	0.3520	17.0%
	write-back	0.2215	10.7%

version. When using 16 threads, the beam-based asynchronous method achieves 26.7× speed-up.

To analyze the performance in detail, we report the performance of each part in the streaming flow in Fig. 4 using CPU in Table IV. This "gradient descent" in async-2-D is different from baseline's gradient descent. async-2-D uses the balanced gradient descent shown in Fig. 4(b) and traverses the domain of projection data, while the baseline implementation only computes on the image volume. We split the table into the computation parts and memory parts. It can be seen that the computation of the forward and backward projections is the bottleneck of the baseline. Although the forward projection is invoked many times, its computation cost of async-2-D is significantly reduced because of the beam-based approach. As aforementioned in Section III-B1, the common usage of the backward projection $\mathcal{R}^\top$ for IR methods is reduced to a light-weight operation $\mathcal{R}_{(i)}$ with the beam-based fine-grained decomposition for Algorithm 1.

2) Overall Performance Comparison: After thorough research on some previous works, we compile the necessary results of different implementations in Table V. In Table V, we list all the information of the implementations, including the algorithms used, the dimension of the raw projection data, the image volume, etc. A limitation of the summary is that not all exact details of implementation are available, although we have tried to find details from public literatures. In Table V, "baseline" is our baseline implementation in the previous section, running on the E5-2650 CPU without the beam-based synchronous update; beam-async is the optimized version using the beam-based asynchronous update. Both PSV is a model-based IR (MBIR) method for LDCT [36], [37]. PSV-ICD [36] is a CPU-based parallel structure with super-voxel to improve the cache performance. PSV-GPU [37] is the GPU version of PSV-ICD and optimizes the coalesced memory access. FPGA-Choi [31] is the implementation of the EM algorithm with hybrid tiling to increase the data reuse rate and locality. cuMBIR [21] is the GPU implementation of Algorithm 1 with a conflict-graph solution to reschedule the X-ray beam execution order and mitigates the memory collision and noncoalesced memory accesses. TO3 is the ALERT Task Order 3 dataset [38]. M.12500 [31] is a helical CT chest scan dataset. "Ours" is our FPGA implementation on ZCU102 for the beam-based asynchronous Algorithm 1 in this work.

We first check the image quality (as addressed in Section V-C) for both our FPGA-accelerated version and baseline version. They have the same level of spatial resolution (9 line pairs per centimeter) and the same average CNR of 1.50.

It is difficult to compare fairly the performance of all the implementations in Table V because of various CT geometries, reconstruction algorithms, unknown implementation details, and different image quality metrics. Moreover, no widely recognized criterion has been proposed in previous acceleration works. Because the overall throughput of the implementation is critical to meet the timing requirement of the routine application of LDCT, we define MUPS as the million updates of voxels per second to normalize the throughput of an implementation as

$$\text{MUPS} = \frac{N \cdot L}{T \cdot 10^6} \tag{11}$$

where N is the total number of the X-ray beams of the dataset; L is the average length of the X-ray beams; and T is the execution time. The speed-up of one implementation compared to the baseline is computed by

$$\text{Speed-up} = \frac{\text{MUPS}_i}{\text{MUPS}_b} \tag{12}$$

where MUPS_b is the MUPS of the baseline and MUPS_i is the MUPS of the implementation under comparison. N can be calculated from the CT geometry. L is an inferred value from X, Y. If previous works did not reveal the details of their line integral algorithms, we can only speculate about N and L. We do not use GFLOPS because it does not focus on the throughput of CT reconstruction and is affected by different CT geometries and line integral algorithms.

In Table V, the baseline, PSV-ICD, and PSV-GPU are all conducted for 2-D CT. 2-D images are too small to stall the memory accesses and have little impact on memory performance. The number of X-ray beams in 2-D CT is much less than that in 3-D CT. Moreover, all three implementations use simple parallel-beam. Because of the geometric simplicity, the parallel beam induces easy computation of the forward and backward projections and is memory-friendly because of the no-intersections of beams from the same view. Although both PSV-ICD and PSV-GPU have demonstrated excellent performance, they are impractical because of the simple 2-D parallel-beam geometry.

cuMBIR [21] is one of the recent works on accelerating CT reconstruction with GPU. Instead of helical scanning in current clinical CT with multirow detectors, they use the circular trajectory of cone-beam with a flat-panel detector. The circular geometry is simpler than the helical and induces less computation of the forward projection. Compared to it, we achieve 11.5× throughput improvement in terms of MUPS.

TABLE V
OVERALL COMPARISON OF OUR IMPLEMENTATION AND OTHER WORKS. (MS IS SHORT FOR MUMFORD–SHAH)

Impl.	baseline	beam-async	PSV-ICD [36]	PSV-GPU [37]	cuMBIR [21]	FPGA-Choi [31]	Ours	
Platform	Xeon CPU 1× E5-2650	Xeon CPU 16× E5-2650	Xeon CPU 20× E5-2660	TITAN X 24 SMMs	V100 84 SMs	Convey HC-1ex 4× Virtex-6 LX760	Xilinx ZCU102 XCZU9EG	
Frequency	2300MHz	2300MHz	2600MHz	1127MHz	1126MHz	100MHZ	200MHz	
Dataset	2D Shepp-Logan	ACR 464	ALERT TO3	ALERT TO3	Shepp-Logan	M.12500	ACR 464	M.12500
CT Geometry	Parallel beam	Helical CT	Parallel beam	Parallel beam	Circular CT	Helical CT	Helical CT	
Image Volume	$512 \cdot 512$ $\times 1$	$874 \cdot 874$ $\times 161$	$512 \cdot 512$ $\times 1$	$512 \cdot 512$ $\times 1$	$256 \cdot 256$ $\times 256$	$512 \cdot 512$ $\times 372$	$874 \cdot 874$ $\times 161$	$512 \cdot 512$ $\times 372$
Projection Data	$180 \cdot 512$	$736 \cdot 16$ $\times 14524$	$720 \cdot 1024$	$720 \cdot 1024$	$256 \cdot 256$ $\times 180$	$672 \cdot 16$ $\times 15022$	$736 \cdot 16$ $\times 14524$	$672 \cdot 16$ $\times 15022$
N	0.092×10^6	171.03×10^6	0.737×10^6	0.737×10^6	11.8×10^6	161.52×10^6	171.03×10^6	161.52×10^6
L	724	1236	724	724	362	724	1236	724
Algorithm	MS Regularization	MS Regularization	MBIR (details unknown)	MBIR (details unknown)	MS Regularization	EM without TV Regularization	MS Regularization	MS Regularization
#Iterations	10	10	4.8	5.9	10	50	10	10
Exec. Time (seconds)	56	6506	1.97	0.41	120	445	513.3	283.9
Throughput (MUPS)	1.27	34.46	288.55	1386.45	37.76	279.85	436.82	438.65
Speed-up	1×	27×	227×	1092×	30×	220×	344×	345×

TABLE VI
RESOURCE UTILIZATION COMPARISON

Impl.	BRAM	DSP	FF	LUT
FPGA-Choi [31]	3690 (64%)	1408 (40%)	1263716 (33%)	1142380 (60%)
Ours	1515 (83%)	2304 (91%)	171532 (31%)	207138 (76%)

FPGA-Choi [31] also accelerates the 3-D LDCT reconstruction with the FPGA platform. The implementation is a mixed parallelism of beam-driven and voxel-driven reconstruction. FPGA-Choi uses the EM algorithm without TV regularization, which has the same computational cost with our beam-based optimized algorithm without the MS regularization. When the MS regularization is applied, it induces more computation load because of more regularization terms than the TV regularization. Our algorithm has more workloads than FPGA-Choi's even with algorithmic optimization because we use a more advanced regularization model. Hence, the improvement comes from the architectural optimization majorly. We achieve a 1.56× throughput compared with FPGA-Choi. We compare the resource utilization between FPGA-Choi and our implementation in Table VI. FPGA-Choi uses 4 Virtex-6 LX760 FPGAs and we use one ZCU102 (XCZU9EG). From Table VI we can see that FPGA-Choi has used more BRAM, FF, and LUT resource units than ours. However, FPGA-Choi uses much fewer DSP slices. In our opinion, it is because FPGA-Choi has not used regularization terms and hence involves much less floating-point arithmetic. In terms of accuracy, we measured SSIM [34], SNR, root-mean-squared error (RMSE), universal quality index (UQI), and correlation coefficient (CC). Our method has SSIM = 0.967, SNR = 21.2, RMSE = 0.02, UQI = 0.963, and CC = 0.997, while FPGA-Choi has SNR = 27.5, RMSE = 3.97, UQI = 0.975, and CC = 0.975. Hence, the accuracy of our results is very close to FPGA-Choi's for the M.12500 chest scans.

E. Summary

Previously in Sections I and II-B, we have concluded our threefold contributions. We analyze their effects as follows.

1) *Asynchronous Beam-Based Parallelism:* Section V-D1 demonstrates that a beam-based asynchronous version has 2.8× speed-up in execution time than the baseline version. When the number of cores increases to 16, the speed-up is 26.7×—remaining 60% efficiency.
2) *Pipeline-Friendly Algorithm Optimization:* To achieve the best pipeline performance, we propose a state machine to remove the data dependency in the original algorithm. Afterward, all the parts in a single PE complete one pixel/voxel per cycle. Besides, we save the hardware resources, as shown in Table VI.
3) *LDCT-Specific Tiling Optimization:* We achieve a 106.9× data reuse rate by LDCT-specific tiling optimization, which means the same reduction rate of external memory accesses. Compared to memory optimization implemented in FPGA-Choi [31], our method is 1.93× better (FPGA-Choi's data reuse rate is 55.3), because we use a quantitative model to search the design space and reschedule the X-rays for the sliding windows (See Section IV-C).

VI. DISCUSSION WITH RELATED WORKS

There have been already many works accelerating the CT reconstruction. In 1994, the first GPU implementation used texture mapping hardware to improve the memory performance [39]. Later, GPU was used to accelerate the forward and backward projections [40]. The MBIR method PSV (PSV-ICD in Table V) was first implemented in CPU to speed up the reconstruction using 20 cores combined with dividing an image into several super voxels [36]. PSV-ICD [36] is a CPU-based parallel structure with supervoxel to improve the cache performance. However, the parallelism was limited by the number of CPU cores and not parallel-friendly main memory. Afterward, a GPU framework (PSV-GPU in Table V) was reported to speed up the reconstruction by data layout transformations, which optimized the coalesced memory access [37]. In order to make full use of GPU resources, their algorithm exploited three levels of parallelism, included intravoxel parallelism, intra-SV parallelism,

and inter-SV parallelism. Another recent GPU implementation is cuMBIR in Table V. It is the GPU implementation of Algorithm 1 with a conflict-graph solution to reschedule the X-ray beam execution order and mitigates the memory collision and noncoalesced memory accesses. Their framework also introduced mixed-precision computing to accelerate reconstruction. Another GPU solution was to process sets of independent voxels parallelly without introducing additional computations [41]. Their algorithm used a multivoxel update scheme within the ICD framework.

The first wave of the FPGA-based CT accelerators focused on the forward and backward projection [42], using the ping-pong buffering technique. Then, FBP was implemented in a ray-by-ray manner using Impulse C tools [43]. Later, a hybrid solution, consisting of a CPU, GPU, and FPGA, was proposed to implement a 3-D image reconstruction with the compressive sensing technique [44]. An FPGA is used to speed up the major computation kernel—forward and backward projection, and a GPU is used to accelerate the TV operations. The EM algorithm was implemented with 4 FPGAs to reconstruct the 3-D images and they proposed a hybrid driven method to improve the data locality [31]. They applied the tiling technique to increase the data reuse rate.

Our framework has several advantages and differences compared with previous works. First, our framework uses sophisticate prior models such as the MS regularization for LDCT. At the same time, previous approaches only focus on the forward and backward projection or simple regularization methods such as total variation. Second, we propose a beam-based asynchronous parallel Algorithm 1 to reduce the overhead of computing the forward and backward projection and communication latency, and increase parallelism. Third, our framework is based on a quantitative analysis of the design space and LDCT-specific tiling optimization to improve the data reuse rate.

VII. Conclusion

In this work, we accelerate the full-IR with the MS functional by FPGA device to reach the required performance for routine clinical applications of 3-D LDCT. In general, three major contributions are made to achieve the desired performance. First, we proposed the asynchronous beam-based update to exploit the parallelism with reduced latency and the backward projection computation cost. Second, after analyzing every step in the streaming dataflow, we find they have the same linear time complexity because of the beam-based approach. We apply pipeline-friendly algorithm optimization to all compute components in the streaming dataflow to utilize the hardware resources with balanced workloads fully. Third, we apply LDCT-specific tiling optimization to save the memory bandwidth and increase the memory performance through prefetching and buffering. The experimental results show that our implementation takes 8.5 min to reconstruct a typical physical phantom with the image quality comparable with the vendor's result. We achieve 11.6× throughput against the state-of-the-art GPU implementation and 1.56× throughput compared with a state-of-the-art FPGA implementation.

Acknowledgment

The authors would like to express great appreciation to John M. Hoffman and Michael McNitt-Gray of UCLA Department of Radiological Sciences for performing the LDCT scans for the CT ACR 464 phantom on the SOMATOM Definition AS at the Department of Radiological Sciences, UCLA, and providing us the projection data and the scanner parameters.

References

[1] M. J. Willemink *et al.*, "Iterative reconstruction techniques for computed tomography—Part 1: Technical principles," *Eur. Radiol.*, vol. 23, no. 6, pp. 1623–1631, 2013.
[2] W. A. Kalender, "Dose in X-ray computed tomography," *Phys. Med. Biol.*, vol. 59, no. 3, pp. R129–R150, 2014.
[3] T. Kubo, "Vendor free basics of radiation dose reduction techniques for CT," *Eur. J. Radiol.*, vol. 110, pp. 14–21, Jan. 2019.
[4] D. Zinsser *et al.*, "Dose reduction and dose management in computed tomography—State of the art," *Rofo-Fortschritte Auf Dem Gebiet Der Rontgenstrahlen Und Der Bildgebenden Verfahren*, vol. 190, no. 6, pp. 531–540, 2018.
[5] J. B. Moser *et al.*, "Radiation dose-reduction strategies in thoracic CT," *Clin. Radiol.*, vol. 72, no. 5, pp. 407–420, 2017.
[6] F. Natterer and F. Wübbeling, *Mathematical Methods in Image Reconstruction*. Philadelphia, PA, USA: SIAM, 2001.
[7] G. T. Herman, *Fundamentals of Computerized Tomography: Image Reconstruction From Projections*, 2nd ed. London, U.K.: Springer, 2009.
[8] W. Stiller, "Basics of iterative reconstruction methods in computed tomography: A vendor-independent overview," *Eur. J. Radiol.*, vol. 109, pp. 147–154, Oct. 2018.
[9] F. Moloney *et al.*, "A phantom study of the performance of model-based iterative reconstruction in low-dose chest and abdominal CT: When are benefits maximized?" *Radiography*, vol. 24, no. 4, pp. 345–351, 2018.
[10] M. Afadzi, E. K. Lysvik, H. K. Andersen, and A. C. T. Martinsen, "Ultra-low dose chest computed tomography: Effect of iterative reconstruction levels on image quality," *Eur. J. Radiol.*, vol. 114, pp. 62–68, Feb. 2019.
[11] O. Scherzer, *Handbook of Mathematical Methods in Imaging*, 2nd ed. New York, NY, USA: Springer, 2015.
[12] D. K. Mumford and J. Shah, "Optimal approximations by piecewise smooth functions and associated variational problems," *Commun. Pure Appl. Math.*, vol. 42, no. 5, pp. 577–685, 1989.
[13] L. I. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Physica D Nonlinear Phenom.*, vol. 60, nos. 1–4, pp. 259–268, 1992.
[14] M. Jiang, P. Maaß, and T. Page, "Regularizing properties of the Mumford–Shah functional for imaging applications," *Inverse Probl.*, vol. 30, no. 3, p. 17, 2014.
[15] L. Rondi and F. Santosa, "Enhanced electrical impedance tomography via the Mumford–Shah functional," *ESAIM Control Optim. Calc. Variations*, vol. 6, pp. 517–538, Jan. 2001.
[16] R. Ramlau and W. Ring, "A Mumford–Shah level-set approach for the inversion and segmentation of X-ray tomography data," *J. Comput. Phys.*, vol. 221, no. 2, pp. 539–557, 2007.
[17] K. Hohm, M. Storath, and A. Weinmann, "An algorithmic framework for Mumford–Shah regularization of inverse problems in imaging," *Inverse Probl.*, vol. 31, no. 11, 2015, Art. no. 115011.
[18] E. Klann, R. Ramlau, and P. Sun, "A Mumford–Shah-type approach to simultaneous reconstruction and segmentation for emission tomography problems with poisson statistics," *J. Inverse Ill Posed Probl.*, vol. 25, no. 4, pp. 521–542, 2017.
[19] W. Zhang *et al.*, "FPGA acceleration by asynchronous parallelization for simultaneous image reconstruction and segmentation based on the Mumford–Shah regularization," in *Proc. SPIE*, 2015, Art. no. 96000H.
[20] L Shen, E. T. Quinto, S. Wang, and M. Jiang, "Simultaneous reconstruction and segmentation with the Mumford–Shah functional for electron tomography," *Inverse Probl. Imag.*, vol. 12, no. 6, pp. 1343–1364, 2018.

[21] X. Li *et al.*, "cuMBIR: An efficient framework for low-dose X-ray CT image reconstruction on GPUs," in *Proc. Int. Conf. Supercomput. (ICS)*, 2018, pp. 184–194.

[22] F. Natterer, *The Mathematics of Computerized Tomography*. Philadelphia, PA, USA: SIAM, 2001.

[23] J. Hsieh, *Computed Tomography: Principles, Design, Artifacts, and Recent Advances*, 2nd ed. Hoboken, NJ, USA: Wiley, 2009, p. 13.

[24] J. Radon, "On the determination of functions from their integral values along certain manifolds," *IEEE Trans. Med. Imag.*, vol. MI-5, no. 4, pp. 170–176, Dec. 1986.

[25] L. Ambrosio and V. M. Tortorelli, "On the approximation of free discontinuity problems," *Bollettino dell'Unione Matematica Italiana*, vol. 6B, no. 1, pp. 105–123, 1992.

[26] K. C. Kiwiel, "Convergence of approximate and incremental subgradient methods for convex optimization," *SIAM J. Optim.*, vol. 14, no. 3, pp. 807–840, 2004.

[27] H. Zhao and A. Reader, "Fast ray-tracing technique to calculate line integral paths in Voxel arrays," in *Proc. IEEE Nucl. Sci. Symp. Conf. Rec.*, vol. 4, 2003, pp. 2808–2812.

[28] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1989.

[29] A. Nedić, D. P. Bertsekas, and V. S. Borkar, *Distributed Asynchronous Incremental Subgradient Methods*, vol. 8. Amsterdam, The Netherlands: Elsevier, 2001, pp. 381–407.

[30] Xilinx. *SDSoC Development Environment*. Accessed: Jul. 14, 2020. [Online]. Available: https://www.xilinx.com/products/design-tools/software-zone/sdsoc.html

[31] Y.-K. Choi and J. Cong, "Acceleration of EM-based 3D CT reconstruction using FPGA," *IEEE Trans. Biomed. Circuits Syst.*, vol. 10, no. 3, pp. 754–767, Jun. 2016.

[32] Xilinx. *Xilinx Zynq UltraScale+ MPSoC ZCU102 Evaluation Kit*. Accessed: Jul. 14, 2020. [Online]. Available: https://www.xilinx.com/products/boards-and-kits/ek-u1-zcu102-g.html

[33] J. Hoffman, S. Young, F. Noo, and M. McNitt-Gray, "Technical note: FreeCT_wFBP: A robust, efficient, open-source implementation of weighted filtered backprojection for helical, fan-beam CT," *Med. Phys.*, vol. 43, no. 3, pp. 1411–1420, 2016.

[34] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.

[35] M. Welvaert and Y. Rosseel, "On the definition of signal-to-noise ratio and contrast-to-noise ratio for fMRI data," *PLoS ONE*, vol. 8, no. 11, pp. 1–10, Nov. 2013.

[36] X. Wang, A. Sabne, S. J. Kisner, A. Raghunathan, C. A. Bouman, and S. P. Midkiff, "High performance model based image reconstruction," in *Proc. 21st ACM SIGPLAN Symp. Principles Pract. Parallel Program. (PPoPP)*, 2016, pp. 1–12.

[37] A. Sabne, X. Wang, S. J. Kisner, C. A. Bouman, A. Raghunathan, and S. P. Midkiff, "Model-based iterative CT image reconstruction on GPUs," in *Proc. 22nd ACM SIGPLAN Symp. Principles Pract. Parallel Program. (PPoPP)*, 2017, pp. 207–220.

[38] DHS/ALERT. *Research and Development of Reconstruction Advances in CT-Based Object Detection Systems*. Accessed: Jul. 14, 2020. [Online]. Available: http://www.neu.edu/alert/assets/taskorders/taskorder03/TO3_FinalReport.pdf

[39] B. Cabral, N. Cam, and J. Foran, "Accelerated volume rendering and tomographic reconstruction using texture mapping hardware," in *Proc. Symp. Vol. Visual.*, 1994, pp. 91–98.

[40] M. Knaup, S. Steckmann, and M. Kachelriess, "GPU-based parallel-beam and cone-beam forward-and backprojection using CUDA," in *Proc. IEEE Nucl. Sci. Symp. Conf. Rec.*, 2008, pp. 5153–5157.

[41] S. Ha and K. Mueller, "A GPU-accelerated multivoxel update scheme for iterative coordinate descent (ICD) optimization in statistical iterative ct reconstruction (SIR)," *IEEE Trans. Comput. Imag.*, vol. 4, no. 3, pp. 355–365, May 2018.

[42] M. Leeser, S. Coric, E. L. Miller, H. Yu, and M. Trepanier, "Parallel-beam backprojection: An FPGA implementation optimized for medical imaging," *J. VLSI Signal Process. Syst. Signal Image Video Technol.*, vol. 39, no. 3, pp. 295–311, 2005.

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

[44] 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.

Wentai Zhang (Graduate Student Member, IEEE) received the B.S. degree in computer science from Peking University, Beijing, China, in 2014, where he is currently pursuing the Ph.D. degree with the Center for Energy-Efficient Computing and Applications.

His current research interests include electronic design automation, heterogeneous accelerators, and medical imaging analytics.

Linjun Qiao received the B.S. degree in computer science from Xidian University, Xi'an, China, in 2015. He is currently pursuing the graduation degree with the Department of Information Science, School of Mathematical Science, Peking University, Beijing, China.

His current research interests include image reconstruction and medical image processing.

William Hsu (Member, IEEE) received the B.S. degree in biomedical engineering from Johns Hopkins University, Baltimore, MD, USA, in 2004, and the Ph.D. degree in biomedical engineering with an emphasis in medical imaging informatics from the University of California at Los Angeles, Los Angeles, CA, USA, in 2009.

He is an Associate Professor with the Department of Radiological Sciences and a Member of the Medical Imaging and Informatics Group, University of California at Los Angeles. His research interests include data integration, predictive modeling, population health management, and imaging informatics.

Yong Cui received the M.D. degree in medical imaging from Peking University, Beijing, China, in 2008.

He has been joining the Department of Radiology, Peking University Cancer Hospital and Institute, Beijing, since 2008 and became the Vice-Chairman since 2016. His current research interests include application of low dose imaging in body cancer and radiomics analysis in body cancer diagnosis.

Ming Jiang (Senior Member, IEEE) received the B.Sc. and Ph.D. degrees in mathematics from Peking University, Beijing, China, in 1984 and 1989, respectively.

He has been a Professor with the Department of Information Science, School of Mathematical Science, Peking University since 2002. His research interests are mathematical and technical innovations in biomedical imaging and image processing, with X-ray computed tomography, optical molecular tomography, and multimodality biomedical imaging as the main focus.

Guojie Luo (Member, IEEE) received the B.S. degree (Hons.) in computer science from Peking University, Beijing, China, in 2005, and the M.S. and Ph.D. degrees in computer science from the University of California at Los Angeles, Los Angeles, CA, USA, in 2008 and 2011, respectively.

He joined the Faculty of Center for Energy-Efficient Computing and Applications, Peking University in August 2011. His current research interests include electronic design automation, customized computing with FPGAs and emerging devices, and medical imaging analytic.