Forward-Projection Architecture for Fast Iterative Image Reconstruction in X-Ray CT

Jung Kuk Kim, *Student Member, IEEE*, Jeffrey A. Fessler, *Fellow, IEEE*, and Zhengya Zhang, *Member, IEEE*

***Abstract*—Iterative image reconstruction can dramatically improve the image quality in X-ray computed tomography (CT), but the computation involves iterative steps of 3D forward- and back-projection, which impedes routine clinical use. To accelerate forward-projection, we analyze the CT geometry to identify the intrinsic parallelism and data access sequence for a highly parallel hardware architecture. To improve the efficiency of this architecture, we propose a water-filling buffer to remove pipeline stalls, and an out-of-order sectored processing to reduce the off-chip memory access by up to three orders of magnitude. We make a floating-point to fixed-point conversion based on numerical simulations and demonstrate comparable image quality at a much lower implementation cost. As a proof of concept, a 5-stage fully pipelined, 55-way parallel separable-footprint forward-projector is prototyped on a Xilinx Virtex-5 FPGA for a throughput of 925.8 million voxel projections/s at 200 MHz clock frequency, 4.6 times higher than an optimized 16-threaded program running on an 8-core 2.8-GHz CPU. A similar architecture can be applied to back-projection for a complete iterative image reconstruction system. The proposed algorithm and architecture can also be applied to hardware platforms such as graphics processing unit and digital signal processor to achieve significant accelerations.**

***Index Terms*—Algorithm and architecture co-optimization, hardware acceleration, iterative image reconstruction, separable footprint projection, X-ray computed tomography.**

I. Introduction

X-RAY computed tomography (CT) is a widely used medical imaging method that produces three-dimensional (3D) images of the inside of a body from many two-dimensional (2D) X-ray images. A 2D X-ray image captures X-ray photons that pass through a body. As different materials attenuate X-ray differently, they can be effectively differentiated by their attenuation coefficients. Using many X-ray images taken around an axis of rotation, the attenuation coefficient of each volume element (voxel) can be reconstructed, providing high-resolution imaging for medical diagnosis.

In current clinical practice, a single CT scan using a state-of-the-art helical CT scanner records up to several thousand X-ray images taken in multiple rotations as the patient's body is moved slowly through the scanner. The projections are captured on an array of detector cells and a dedicated computer is used for image construction. Efficient algorithms, such as filtered back-projection (FBP) [1] and its variants, are in common commercial use to handle large projection data sets and reconstruct images at sufficient throughput. However, being an analytical algorithm, FBP disregards the effects of noise. To improve the image quality and/or reduce X-ray dose, statistical image reconstruction methods have been proposed [2], [3]. These methods are based on accurate projection models and measurement statistics, and formulated as a maximum likelihood (ML) estimation. Iterative algorithms such as conjugate gradient (CG) [4], coordinate descent (CD) [5] and ordered subsets (OS) [6], have been proposed. These algorithms find the minimizer of a cost function by iterative forward- and back-projection. Iterations increase the compute load substantially over FBP and impede routine clinical use.

Recently, a separable footprint (SF) projection algorithm was designed to simplify the forward-projection by approximating the voxel footprints as separable functions [7]. The SF projector has high accuracy and favorable speed, but it is still very computationally intensive: each forward- and back-projection requires on the order of 100 billion floating-point multiply-accumulate (MAC) operations, requiring minutes or longer for each forward- and back-projection on a state-of-the-art multicore microprocessor [8].

High-performance computing platforms have been proposed to accelerate image reconstruction. For example, graphics processing unit (GPU) has recently been demonstrated to achieve 10 to 100 times speedup over a microprocessor for image reconstruction [9], [10]. As a vector processor, GPU can be programmed for efficient parallel processing [11]. Provided with sufficient memory bandwidth, GPU accomplished a 30 times speedup of cone-beam Feldkamp (FDK) back-projection over a system based on 12 2.6-GHz dual-core Xeon processors [9], and a 12 times speedup of algebraic reconstruction [10]. Field-programmable gate array (FPGA) is an another family of hardware platforms that enable more flexibility in mapping parallel computation with an improved efficiency. It was shown to accomplish a 6 times speedup of the cone-beam Feldkamp (FDK) back-projection [9], [12]. However, existing GPU and FPGA implementations are tailored to analytical reconstruction algorithms or algebraic reconstruction methods [9], [10], [12], [13], and challenges still remain in mapping statistical iterative algorithms.

In this paper, we propose architecture and algorithm co-optimization for iterative image reconstruction. We show through numerical simulation that iterative image reconstruction algorithm can be robust to quantization noise. Even with a much shorter word length and coarse quantization, the resulting noise introduced to the reconstructed image is limited, causing no perceptual degradation in image quality. The results provide the

Manuscript received January 19, 2012; revised May 04, 2012; accepted June 23, 2012. Date of publication July 13, 2012; date of current version September 11, 2012. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Tong Zhang. This work was supported in part by a Korea Foundation for Advanced Studies (KFAS) Scholarship and the University of Michigan. The work of J. Fessler is supported by NIH Grant R01-HL-098686.

The authors are with the Department of Electrical and Computer Engineering, University of Michigan, Ann Arbor, MI 48109-2122 USA (e-mail: jungkook@umich.edu; fessler@umich.edu; zhengya@eecs.umich.edu).

Digital Object Identifier 10.1109/TSP.2012.2208636

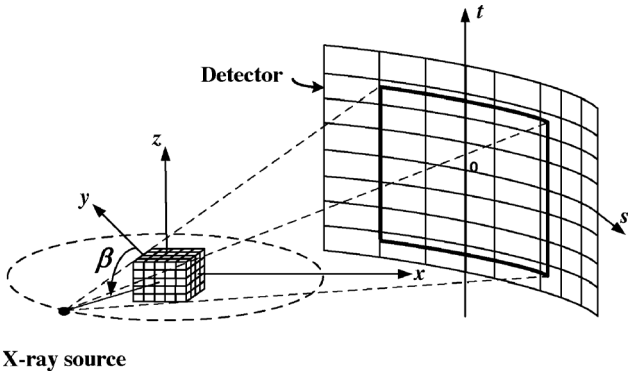


Fig. 1. Axial cone-beam arc-detector geometry for X-ray CT.

basis of a fixed-point quantization that cuts the memory bandwidth and reduces the complexity of arithmetic operations, thus enabling more parallel implementations.

We propose a highly efficient hardware architecture based on a thorough geometry analysis that helps simplify complex control loops, eliminate data dependencies, and maximize temporal and spatial locality of reference. In particular, we present algorithm restructuring to take advantage of loop-level parallelism, water-filling buffer to minimize pipeline stalls, and out-of-order scheduling to compress off-chip memory bandwidth to enable more parallel architectures.

A prototype 55-way parallel SF forward-projector is demonstrated on a Xilinx Virtex-5 FPGA [14] as a proof of concept. The design is capable of completing 925.8 million voxel projections/s. The proposed architecture is also applicable to back-projection and motivates more efficient designs on alternative hardware platforms including GPU and digital signal processors (DSP). The numerical and geometrical insights can be employed in both software and hardware implementations of iterative image reconstruction to achieve significant accelerations.

II. Background

Current generation CT systems have a cone-beam projection geometry, illustrated in Fig. 1 [3], [15]–[17]. The X-ray source rotates on a circle centered at $(x, y) = (0, 0)$ on the $z = 0$ plane. The angle β indexes the projection view measured from positive y-axis to X-ray source. For each angle β, the source emits X-rays that project the volume onto the detector. The transaxial direction s is perpendicular to z and the axial direction t is parallel to z.

A. Statistical Iterative Image Reconstruction

A CT system captures a large series of projections at different view angles, recorded as sinogram. Mathematically, sinogram y can be modeled as $y = Af + \varepsilon$, where f represents the volume being imaged, A is the system matrix, or the forward-projection model, and ε denotes measurement noise. The goal of image reconstruction is to estimate the 3D image f from the measured sinogram y. A statistical image reconstruction method performs the ML estimation of f based on detector measurement statistics. The estimation $\hat{f}$ can be formulated as a solution to a weighted least square (WLS) problem [3], [18].

$$\hat{f} = \arg\min_f \frac{1}{2}\|y - Af\|_W^2, \quad (1)$$

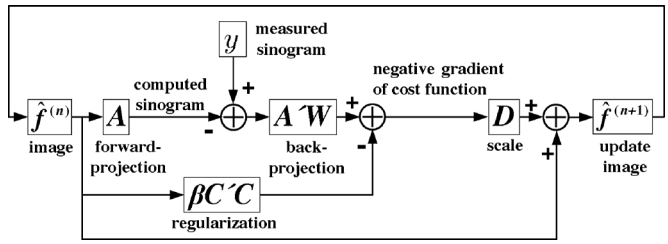


Fig. 2. Block diagram of iterative image reconstruction.

where W is a diagonal matrix with entries based on photon measurement statistics [3]. A solution to (1) satisfies $A'WA\hat{f} = A'Wy$ [18]. If $A'WA$ is invertible, the unique solution to (1) is given by $\hat{f} = (A'WA)^{-1}A'Wy$, where A', the adjoint of the system matrix, represents the back-projection model. This solution can be interpreted as the weighted back-projection of y, followed by a deconvolution filter $(A'WA)^{-1}$. As the deconvolution filter has a high pass characteristic, the deconvolved image is affected by high frequency noise [18]. One approach to control this noise is to add a penalty term to form a penalized weighted least square (PWLS) [3], [18] cost function:

$$\hat{f} = \arg\min_f \Psi(f) = \arg\min_f \frac{1}{2}\|y - Af\|_W^2 + \beta R(f), \quad (2)$$

where $R(f)$ is known as the regularizer and β is a regularization parameter. One example of $R(f)$ is an edge-preserving regularizer [19].

Minimizing (2) requires iterative methods [4]–[6]. In this paper we consider a diagonally preconditioned gradient descent method to solve (2) [6], [18]:

$$\begin{aligned}\hat{f}^{(i+1)} &= \hat{f}^{(i)} - D\nabla\Psi\left(\hat{f}^{(i)}\right) \\ &= \hat{f}^{(i)} + D\left[A'W\left(y - A\hat{f}^{(i)}\right) - \beta\nabla R\left(\hat{f}^{(i)}\right)\right]. \quad (3)\end{aligned}$$

The solution is obtained iteratively. In each iteration, a new 3D image estimate $\hat{f}^{(i+1)}$ is obtained by updating the previous image $\hat{f}^{(i)}$ with a chosen step, the negative gradient of the cost function $\Psi(\hat{f})$ scaled by D. Fig. 2 shows a block diagram of this iterative approach. To start, the CT scanner produces the measured sinogram, y and the FBP algorithm is used to estimate the initial image $\hat{f}^{(0)}$, followed by computed forward-projection to obtain the computed sinogram $A\hat{f}^{(0)}$. The error between the computed and measured sinogram $y - A\hat{f}^{(0)}$ is back-projected $A'W(y - A\hat{f}^{(0)})$, then offset by a regularization term. The result is scaled by D, and used to improve the initial image to produce $\hat{f}^{(1)}$. The image $\hat{f}$ is iteratively updated to minimize the cost function.

B. Forward- and Back-Projection

Forward and back-projection are the most computationally intense operations in iterative image reconstruction due to the large size of the system matrix A. It is infeasible to store A, thus the forward-projection $Af^{(i)}$, and back-projection $A'W(y - f^{(i)})$ in (3) are computed on the fly.

The forward-projection is mathematically based on the Radon transform. The Radon transform of a 3D volume $f(x,y,z)$ at view angle β is described by the line integrals [7]:

$$g(s,t;\beta) = \int_{L(s,t,\beta)} f(x,y,z)dl, \tag{4}$$

where $L(s,t,\beta)$ is the line that connects the X-ray source and the detector cell at (s,t). In a practical implementation, a 3D continuous volume $f(x,y,z)$ is discretized to a collection of volume elements, or voxels $f[n_1,n_2,n_3]$, where $[n_1,n_2,n_3]$ is the voxel coordinate. The grid spacings are $\Delta_{\mathrm{x}},\Delta_{\mathrm{y}},\Delta_{\mathrm{z}}$ and dimensions are $N_{\mathrm{x}},N_{\mathrm{y}},N_{\mathrm{z}}$ along the x,y,z directions. Let β_0 be the common voxel basis function, defined as a cubic function, $\beta_0(x,y,z) = \mathrm{rect}(x)\mathrm{rect}(y)\mathrm{rect}(z)$, and $(x_{\mathrm{c}}[n_1], y_{\mathrm{c}}[n_2], z_{\mathrm{c}}[n_3])$ be the location of voxel $[n_1,n_2,n_3]$. We have

$$f(x,y,z) = \sum_{n_1=0}^{N_{\mathrm{x}}-1}\sum_{n_2=0}^{N_{\mathrm{y}}-1}\sum_{n_3=0}^{N_{\mathrm{z}}-1} f[n_1,n_2,n_3] \cdot\beta_0\left(\frac{x-x_{\mathrm{c}}[n_1]}{\Delta_{\mathrm{x}}}, \frac{y-y_{\mathrm{c}}[n_2]}{\Delta_{\mathrm{y}}}, \frac{z-z_{\mathrm{c}}[n_3]}{\Delta_{\mathrm{z}}}\right). \tag{5}$$

To account for the finite detector cell size, the projection is convolved with the detector blur $h(s,t)$. Following a common assumption that the detector blur is shift invariant, independent of the view angle β, and acts only along s and t coordinates, then the ideal noiseless forward-projection on the detector cell $[k,l]$ centered at $(s_{\mathrm{k}},t_{\mathrm{l}})$ is given by

$$y_\beta[k,l] = \sum_{n_1=0}^{N_{\mathrm{x}}-1}\sum_{n_2=0}^{N_{\mathrm{y}}-1}\sum_{n_3=0}^{N_{\mathrm{z}}-1} a_{\mathrm{b}}(s_{\mathrm{k}},t_{\mathrm{l}};\beta;n_1,n_2,n_3) \cdot f[n_1,n_2,n_3], \tag{6}$$

where

$$a_{\mathrm{b}}(s_{\mathrm{k}},t_{\mathrm{l}};\beta;n_1,n_2,n_3) \triangleq \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h(s_{\mathrm{k}}-s,t_{\mathrm{l}}-t) \times a(s,t;\beta;n_1,n_2,n_3)\,ds\,dt, \tag{7}$$

and

$$a(s,t;\beta;n_1,n_2,n_3) \triangleq \int_{L(s,t,\beta)} \beta_0\left(\frac{x-x_{\mathrm{c}}[n_1]}{\Delta_{\mathrm{x}}}, \frac{y-y_{\mathrm{c}}[n_2]}{\Delta_{\mathrm{y}}}, \frac{z-z_{\mathrm{c}}[n_3]}{\Delta_{\mathrm{z}}}\right) dl, \tag{8}$$

where $a(s,t;\beta;n_1,n_2,n_3)$ is the footprint of voxel $[n_1,n_2,n_3]$ and $a_{\mathrm{b}}(s_{\mathrm{k}},t_{\mathrm{l}};\beta;n_1,n_2,n_3)$ is the blurred footprint. For a detailed description of this derivation, see [18]. The separable footprint (SF) method [7] approximates the blurred footprint function as the product of $a_{\mathrm{b1}}(s_{\mathrm{k}},\beta;n_1,n_2)$ and $a_{\mathrm{b2}}(t_{\mathrm{l}},\beta;n_1,n_2,n_3)$, thus (6) is approximated as

$$y_\beta[k,l] \approx \sum_{n_1=0}^{N_{\mathrm{x}}-1}\sum_{n_2=0}^{N_{\mathrm{y}}-1}\sum_{n_3=0}^{N_{\mathrm{z}}-1} a_{\mathrm{b1}}[k,\beta;n_1,n_2] \cdot a_{\mathrm{b2}}[l,\beta;n_1,n_2,n_3] f[n_1,n_2,n_3]. \tag{9}$$

Based on (9), one complete forward-projection involves multiplication and summation over six nested loops: n_1,n_2,n_3,β,k, and l. For a practical object made up of more than 10 million voxels, a SF forward-projection that comprises more than 900 view angles, as in a commercial axial CT scanner [3], requires on the order of 100 billion multiply-accumulate (MAC) operations. In the following sections, we explore architecture and algorithm co-optimization to accelerate the SF forward-projection.

For the sake of completeness, we briefly summarize back-projection. Back-projection is the operation that smears the projection in detector space back into the object space to reconstruct the 3D volume [18]. Back-projection is mathematically described as

$$f_{\mathrm{b}}[n_1,n_2,n_3] = \sum_{n_\beta=0}^{N_\beta-1}\sum_{l=0}^{N_{\mathrm{t}}-1}\sum_{k=0}^{N_{\mathrm{s}}-1} a_{\mathrm{b}}(s_{\mathrm{k}},t_{\mathrm{l}};n_\beta;n_1,n_2,n_3) \cdot g[k,l,n_\beta], \tag{10}$$

where $g[k,l,n_\beta]$ is the weighted difference between measured sinogram and the computed sinogram $y_{\beta(n_\beta)}[k,l]$. Similarly, the SF method approximates back-projection as

$$f_{\mathrm{b}}[n_1,n_2,n_3] \approx \sum_{n_\beta=0}^{N_\beta-1}\sum_{l=0}^{N_{\mathrm{t}}-1}\sum_{k=0}^{N_{\mathrm{s}}-1} a_{\mathrm{b1}}[k,n_\beta;n_1,n_2] \cdot a_{\mathrm{b2}}[l,n_\beta;n_1,n_2,n_3] g[k,l,n_\beta]. \tag{11}$$

Note that the equations governing forward- and back-projection are similar and they also share a common architecture. In this paper, we will focus the discussions on forward-projection, but the results can also be applied to back-projection.

III. Quantization Error Investigation

Iterative CT image reconstruction algorithms are usually implemented in 32-bit single-precision floating-point quantization. Floating-point arithmetic costs more hardware resources and longer latency than integer (or fixed-point) operations. The substantially smaller area and higher speed provide strong incentives for using fixed-point operations. However, fixed-point quantization introduces errors that may degrade image quality. We show in the following that good image quality can be achieved with appropriate quantization choice and sufficient number of iterations.

Our experiment was done using a 61-slice test volume, with each slice made up of 320×320 voxels. Errors are defined in reference to a baseline that is the image reconstructed using 32-bit floating-point quantization after 1 000 iterations. We converted floating-point to fixed-point and varied the word length and quantization of each parameter and operand. Mean absolute error (MAE) and root mean square error (RMSE) of the image update in every iteration were measured compared to the baseline. The errors are expressed in Hounsfield unit (HU), which is a linear transformation of the linear attenuation coefficient (the attenuation coefficient of water at standard pressure and temperature is defined as 0 HU and that of the air is $-1\,000$ HU).

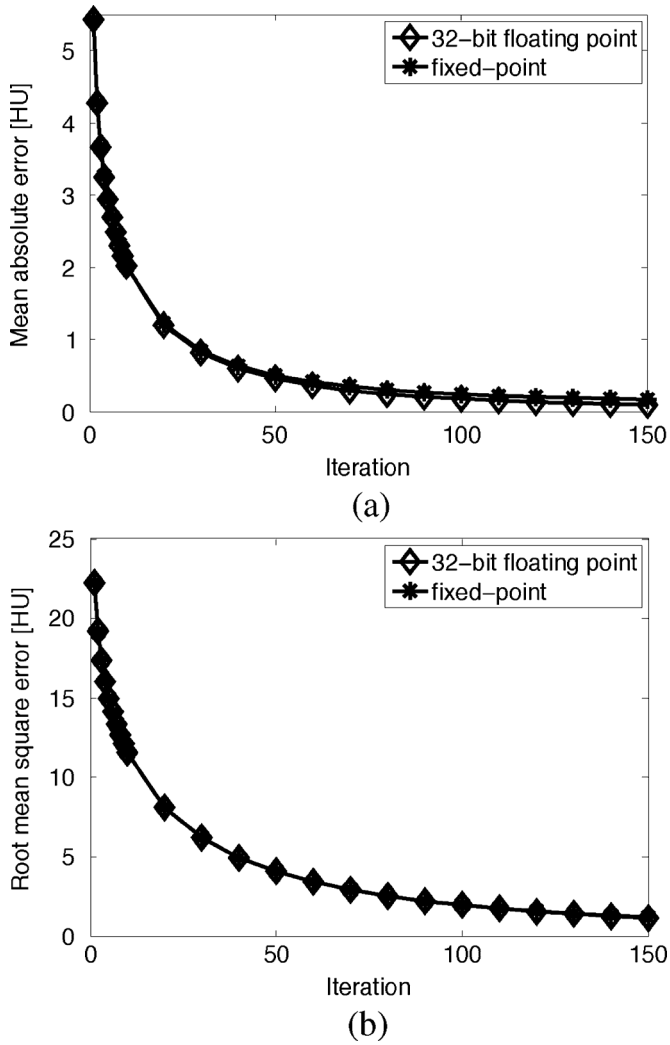


Fig. 3. (a) Mean absolute error and (b) root mean square error of iterative image reconstruction using floating-point and fixed-point quantization.

TABLE I
FIXED-POINT QUANTIZATION OF ITERATIVE IMAGE RECONSTRUCTION

Forward-projection		Back-projection	
Parameter	Quant.	Parameter	Quant.
f	Q13.0	g	Q5.15
$a_{\mathrm{b}2}$	Q1.15	$a_{\mathrm{b}1}$	Q3.17
$a_{\mathrm{b}2}f$	Q13.3	$a_{\mathrm{b}1}g$	Q7.15
$\sum_{n_3} a_{\mathrm{b}2}f$	Q13.3	$\sum_{k} a_{\mathrm{b}1}g$	Q8.15
$a_{\mathrm{b}1}$	Q3.13	$a_{\mathrm{b}2}$	Q1.15
$\sum_{n_3} a_{\mathrm{b}1}a_{\mathrm{b}2}f$	Q15.8	$\sum_{k} a_{\mathrm{b}1}a_{\mathrm{b}2}g$	Q9.15
$\sum_{n_1}\sum_{n_2}\sum_{n_3} a_{\mathrm{b}1}a_{\mathrm{b}2}f$	Q20.8	$\sum_{n_\beta}\sum_{l}\sum_{k} a_{\mathrm{b}1}a_{\mathrm{b}2}g$	Q9.15

We used an OS algorithm [6] with 82 subsets which is a variation of (3) that uses a subset of the projection views for each update. Fig. 3 comprises the 32-bit floating-point quantization and the fixed-point quantization described in Table I. We use the notation $\mathrm{Q}n_{\mathrm{int}} \cdot n_{\mathrm{frac}}$ to denote a fixed-point format with n_{int} before the radix point and n_{frac} after the radix point. The experiment confirms that the fixed-point quantization errors introduced can be limited to fairly low levels. More iterations can help suppress the errors, and the word length can be increased to reduce the errors further if necessary.

Fig. 4 shows the images obtained by iterative image reconstruction as well as the absolute pixel-by-pixel differences between the reconstructed image using 32-bit floating-point quantization and the reconstructed image using fixed-point quantization. Three representative slices in the region of interest are shown from left to right. The vast majority of the pixel errors remain relatively small. We observe no perceptual difference between floating-point and fixed-point reconstructed images. These initial results suggest that the iterative image reconstruction algorithm can be robust to quantization error. The property allows us to simplify the hardware with much more efficient integer arithmetic and smaller memory.

IV. ARCHITECTURE AND ALGORITHM CO-OPTIMIZATION

Forward- and back-projection are the core and most computationally intense building blocks of iterative image reconstruction. A simplistic forward-projection architecture includes image memory on the input and detector memory on the output as in Fig. 5; back-projection exchanges the positions of image and detector memory but its processing architecture is similar. In a state-of-the-art commercial CT scanner, the image and detector datasets are up to 1 GB in size. Such enormous datasets can only be accommodated in off-chip memory, and input and output data are selectively brought to on-chip memory (cache) for processing. The on-chip memory is smaller but much faster and sometimes immediately accessible by the processor, while the larger off-chip memory interface is much slower and costs a longer latency to access. Iterative image reconstruction algorithm in its original form requires moving of large datasets on and off chip constantly, resulting in a low throughput due to limited off-chip memory interface.

Parallelism can be used to improve the throughput, but it further increases memory bandwidth. The architecture can be pipelined, though its throughput is far from ideal due to loop-carried dependencies from geometry processing. In the following we investigate the projection geometry and design algorithms and architectures to reduce the memory bottleneck and improve the efficiency of parallel and pipelined architectures.

A. Projection Geometry

The projection geometry is central to the proposed algorithms and architectures. Fig. 6 illustrates the X-ray projection of a single voxel of dimension $\Delta_{\mathrm{x}} \times \Delta_{\mathrm{y}} \times \Delta_{\mathrm{z}}$ centered at (x, y, z). We define the magnification factor $M_\beta(x, y)$ as the ratio of the source-to-detector distance D_{sd} (which is a constant in cone-beam geometry) over the distance between the source and $(x, y, 0)$. (The magnification factors of all voxels in an axial column are equal.) $M_\beta(x, y)$ is maximized when the voxel is closest to the X-ray source and minimized when the voxel is furthest to the X-ray source, i.e.,

$$\frac{D_{\mathrm{sd}}}{D_{\mathrm{s0}} + FOV/2} \le M_\beta(x, y) \le \frac{D_{\mathrm{sd}}}{D_{\mathrm{s0}} - FOV/2}, \tag{12}$$

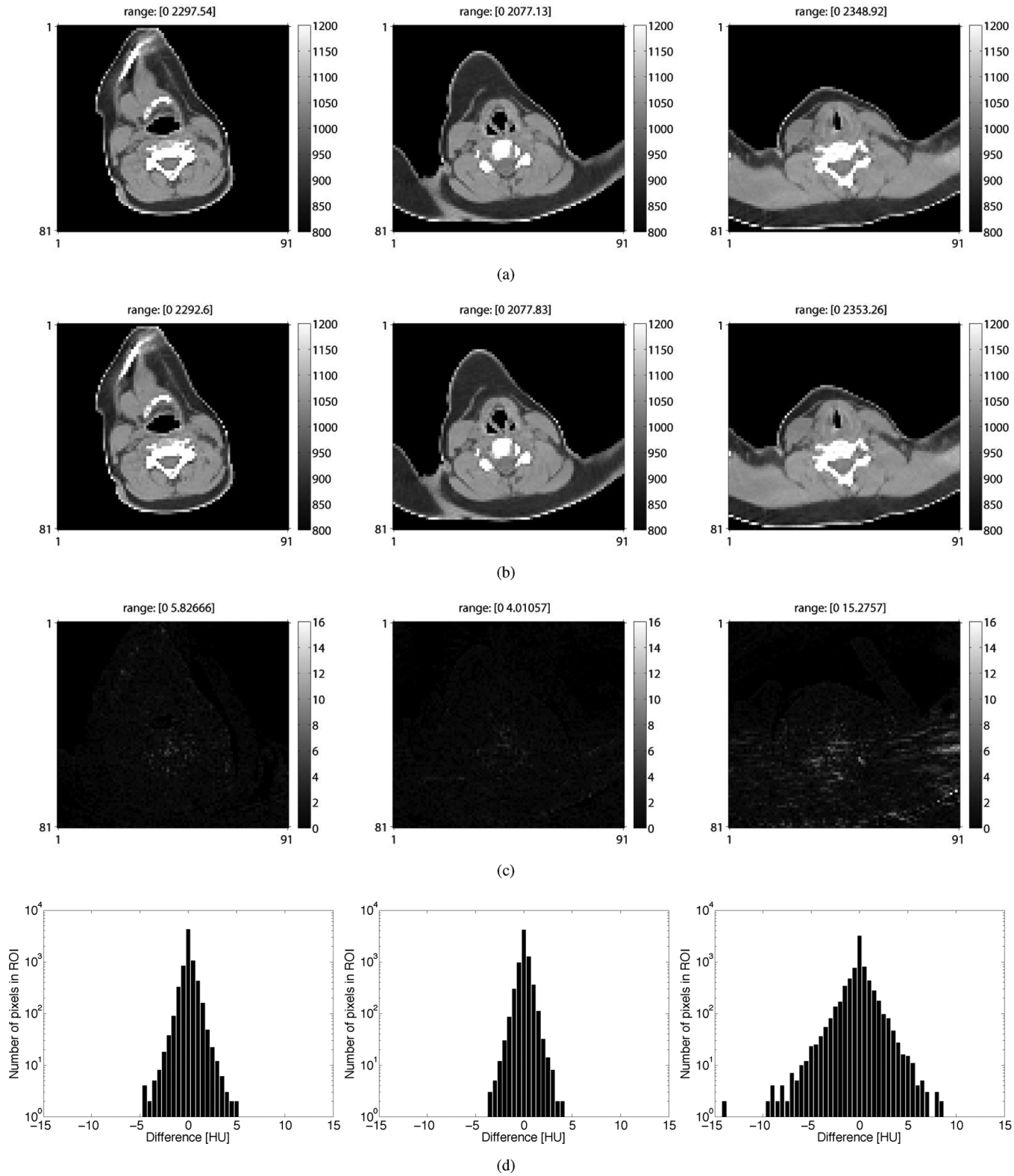


Fig. 4. Reconstructed images using (a) 32-bit floating-point quantization, (b) fixed-point quantization, (c) absolute pixel-by-pixel differences between the floating-point and the fixed-point quantization, and (d) histograms of the differences in logarithm scale. Three slices in the region of interest are shown: slice 17, 31 and 45 from left to right.

where FOV, or field of view, is the diameter of the volume that is reconstructed from all view angles, and D_{s0} is the source-to-rotation-center distance.

Now, consider the position of a voxel relative to the X-ray source—the transaxial width of a voxel's projection is maximized if the transaxial diagonal of the voxel is perpendicular to

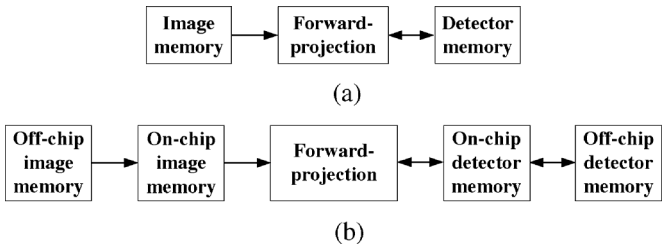


Fig. 5. High-level forward-projection architecture. (a) One-level memory. (b) Two-level memory hierarchy.

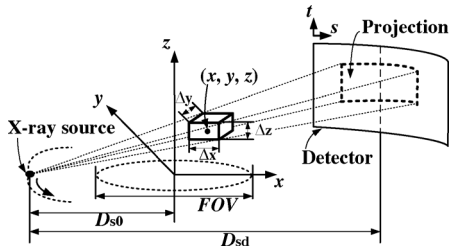


Fig. 6. Forward-projection of a single voxel.

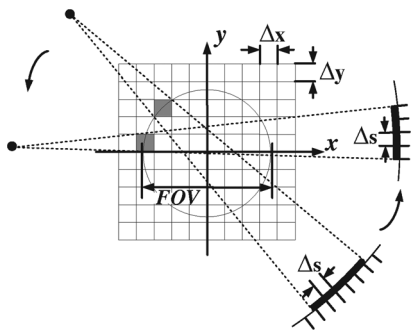


Fig. 7. Top view of the transaxial span of the forward-projection of one voxel.

the line joining the X-ray source and the center of the voxel, illustrated in Fig. 7. Considering both the magnification and the transaxial diagonal of the voxel, the transaxial span of the projection of a voxel, quantized to the axial spacing Δ_{s} of the detector grid, is

$$s_{\mathrm{transaxial}} \leq \left\lceil \frac{\sqrt{\Delta_{\mathrm{x}}^2 + \Delta_{\mathrm{y}}^2}}{\Delta_{\mathrm{s}}} M_\beta(x,y) \right\rceil + 1 \leq \left\lceil \frac{\sqrt{\Delta_{\mathrm{x}}^2 + \Delta_{\mathrm{y}}^2}}{\Delta_{\mathrm{s}}} \frac{D_{\mathrm{sd}}}{D_{\mathrm{s0}} - FOV/2} \right\rceil + 1 = s_{\mathrm{bin}}, \quad (13)$$

where $\lceil\rceil$ denotes ceiling.

The magnification factor in (12) can also be used to derive the axial span. Typically the axial spacing Δ_{t} of the detector grid is designed to match the voxel grid Δ_z by having $\Delta_{\mathrm{t}}/\Delta_z = D_{\mathrm{sd}}/D_{\mathrm{s0}}$. Therefore, on average one voxel maps to one detector cell along the axial direction. However, grid misalignment and geometry cause multiple consecutive voxels in an axial column to project to a single detector cell, as shown in Fig. 8. The axial height of a voxel's projection is minimized if the voxel is located

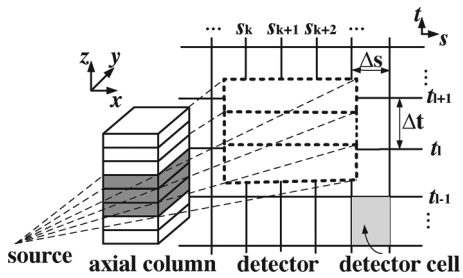


Fig. 8. Forward-projection of one axial column of voxels.

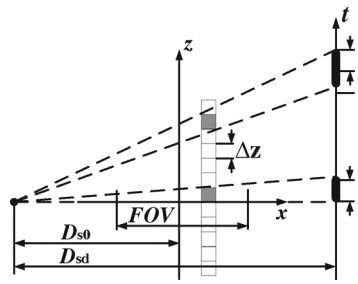


Fig. 9. Side view of the axial span of the forward-projection of one voxel.

TABLE II
SAMPLE HELICAL CONE-BEAM CT GEOMETRY PARAMETERS

Parameter	Value	Parameter	Value
N_1	320	Δ_{x}	2.1911 $[mm]$
N_2	320	Δ_{y}	2.1911 $[mm]$
N_3	61	Δ_{z}	0.625 $[mm]$
N_s	888	Δ_{s}	1.023 $[mm]$
N_t	32	Δ_{t}	1.096 $[mm]$
N_{views}	3,625	D_{s0}	541.0 $[mm]$
Views per rotation	984	D_{sd}	949.075 $[mm]$
Pitch	0.513	FOV	500 $[mm]$

on the $z = 0$ plane, illustrated in Fig. 9. It follows that the number of voxels in an axial column that project to a single detector cell is

$$z_{\mathrm{axial}} \leq \left\lceil \frac{\Delta_{\mathrm{t}}}{\Delta_{\mathrm{z}}} \frac{1}{M_\beta(x,y)} \right\rceil + 1 \leq \left\lceil \frac{FOV/2}{D_{\mathrm{s0}}} \right\rceil + 2 = z_{\mathrm{vx}}. \quad (14)$$

For a numerical example, substituting sample helical cone-beam geometry parameters given in Table II, we get $s_{\mathrm{bin}} = 11$ and $z_{\mathrm{vx}} = 3$, i.e., one voxel's projection spans at most 11 detector cells along the transaxial direction, and at most 3 consecutive voxels in an axial column project to one detector cell.

B. Loop-Level Parallelism and Water-Filling

The SF forward-projection algorithm contains six layers of nested loops (9): β (view angle), n_1 (x index), n_2 (y index), n_3 (z index), l (t index) and k (s index) for each forward-projection. The innermost k loop computes the transaxial projection of a voxel. As discussed in the previous section, one voxel projects to a row of up to s_{bin} detector cells, each of which can be evaluated independently. Thus we exploit loop-level parallelism by allocating s_{bin} multiply-accumulate (MAC) units and detector memory banks for the transaxial projection, as shown in Fig. 10.

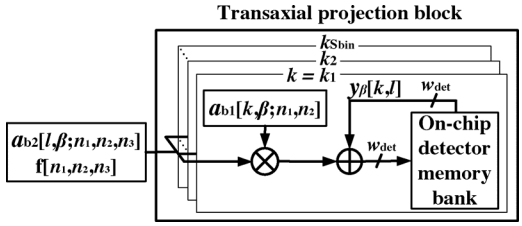


Fig. 10. Parallel transaxial projection.

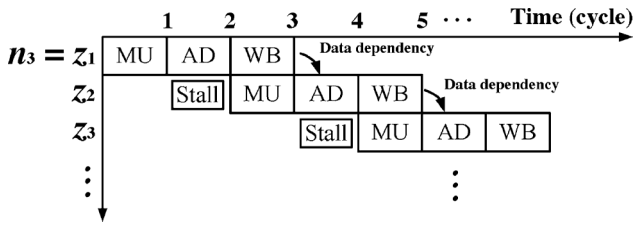


Fig. 11. Pipeline bubbles inserted to resolve data dependencies in axial projections.

The quantization study showed that the transaxial projection can be carried out in a 16-bit × 16-bit fixed-point multiply followed by a 28-bit accumulate. To operate at a high clock frequency, e.g., 200 MHz on a Xilinx Virtex-5 FPGA, we pipeline the MAC unit to 3 stages: multiply (MU), add (AD), and write back (WB). Let $w_{\rm det}$ be the wordlength of $y_\beta[k,l]$ that is stored in the detector memory and $f_{\rm clk}$ be the clock frequency. Then, the required read and write bandwidth to the on-chip detector memory is $2w_{\rm det}f_{\rm clk}$ b/s. Since one complete transaxial projection block uses $s_{\rm bin}$ MAC units, the total on-chip detector memory bandwidth is $2s_{\rm bin}w_{\rm det}f_{\rm clk}$ b/s.

The outer l loop can be easily pipelined, but it is complicated by loop-carried dependencies: multiple voxels in an axial column can project to a single detector cell, as illustrated in Fig. 8, so the pipeline would have to be stalled, waiting for write back to complete before next add. The 3-stage pipeline chart in Fig. 11 shows that one pipeline bubble is necessary to resolve data dependency. A deeper pipeline will result in more stalls.

The mismatch between the voxel grid and detector grid requires the joint consideration between the n_3 loop and the l loop. To eliminate loop-carried dependencies, we propose an algorithm transformation to merge the two loops. In the transformed algorithm, for each l-th detector cell, we identify the group of contiguous voxels along the axial column that project to the cell and sum up the contributions. In particular, we allocate $z_{\rm vx}$ shift registers, each providing one candidate voxel (because up to $z_{\rm vx}$ voxels in an axial column project to a single detector cell), as in Fig. 12. Each candidate voxel is multiplied by its axial footprint and the contributions are summed, which is equivalent to a partial unrolling of the n_3 loop.

An example is shown in Fig. 13 using 2-stage shift registers and input prefetching. Initially, $l = l_1$, voxels z_1 and z_2 project to detector cell l_1. A controller sets $a_{\rm b2,1} = a_{\rm b2}[l_1,\beta;n_1,n_2,z_1]$, $a_{\rm b2,2} = a_{\rm b2}[l_1,\beta;n_1,n_2,z_2]$ and $a_{\rm b2,3} = 0$, respectively. The contributions by voxels z_1 and z_2 to the axial projection are summed, followed by transaxial projection. Next, $l = l_2$, voxels z_2 and z_3 project to detector cell l_2. The controller sets $en_1 = 1, en_2 = 0, en_3 = 0$ to pop z_1 and keep z_2 and z_3. Now the water level in $SR1$ has dropped

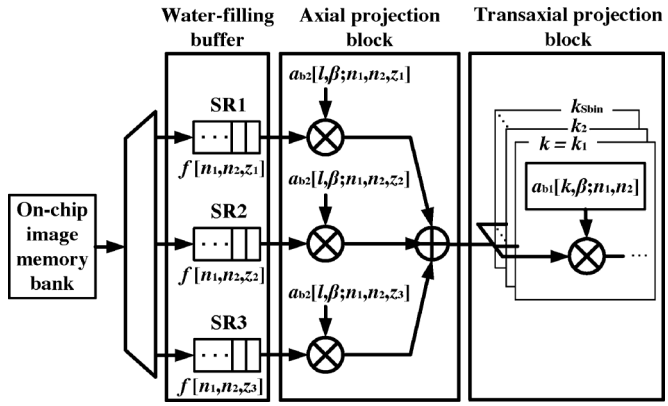


Fig. 12. Water-filling buffer and partially-unrolled axial projection.

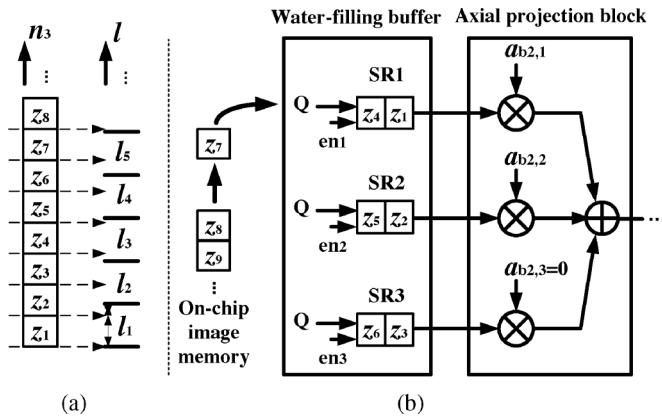


Fig. 13. Example showing (a) n_3 and l grid mismatch, and (b) the corresponding water-filling buffering scheme.

TABLE III
PIPELINE STALL RATE VERSUS SHIFT REGISTER LENGTH OF THE WATER-FILLING BUFFER

Shift register length	Stall rate (%)
1	9.70
2	7.42
3	5.65
4	4.36
5	3.48

and the input multiplexer will direct the new voxel input z_7 to $SR1$.

Note that in the above example, one new voxel is brought in the water-filling buffer every cycle to support the average input consumption rate. The average consumption rate is one input per clock cycle because $\Delta_{\rm z}$ and $\Delta_{\rm t}$ are designed to be matched as previously described. However, the actual input consumption varies every cycle and prefetching is needed to avoid stalling the pipeline. A longer shift register and prefetching guarantee a lower stall rate, but increase latency and resource usage. We experimentally verified the stall rate versus shift register length, and the results are listed in Table III. We choose a 2-stage shift registers in our prototype design for a stall rate $P_{\rm stall} = 7.42\%$. A lower stall rate is possible with longer shift registers.

The new water-filling architecture can be implemented using 3 MAC units that are pipelined in two stages: read (RE) and sum (SU), which augment the 3-stage pipeline in Fig. 11 to 5 stages as in Fig. 14. Pipeline bubbles due to loop-carried

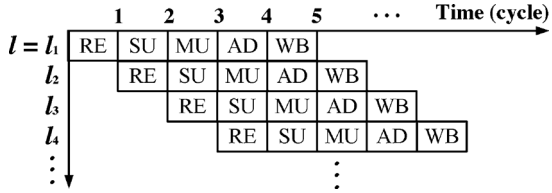


Fig. 14. Pipeline chart for the complete forward-projection module.

TABLE IV
FPGA RESOURCE UTILIZATION OF A FORWARD-PROJECTION MODULE BASED ON XILINX VIRTEX-5 XC5VLX155T DEVICE

	Usage	Utilization ratio
FPGA slice register	10,419	10%
FPGA slice LUT	9,124	9%
Occupied FPGA slice	5,119	21%
BRAM	37	17%
DSP48E	17	13%

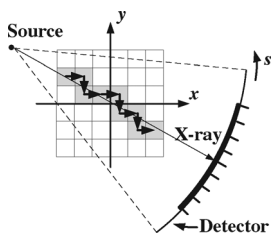


Fig. 15. Top view of the forward-projection following an X-ray.

dependencies have been eliminated to achieve an average throughput of $f_{\rm clk}(1-P_{\rm stall})$ voxel projections/s. The required on-chip image memory bandwidth is $w_{\rm img}f_{\rm clk}$ b/s with $w_{\rm img}$ as the voxel wordlength. Substituting parameters from Table II, $P_{\rm stall}=7.42\%$, and $f_{\rm clk}=200$ MHz that is typical of an FPGA platform, the proposed projection module completes 185.2 million voxel projections/s and requires an on-chip image memory bandwidth of 2.6 Gb/s and detector memory bandwidth of 123.2 Gb/s. In the following section, we propose out-of-order scheduling to reduce the detector memory bandwidth.

A complete forward-projection module consisting of the water-filling axial projection and parallel transaxial projection has been synthesized on a Xilinx Virtex-5 XC5VLX155T FPGA and the device usage is listed in Table IV.

C. Out-of-Order Scheduling

We could further parallelize the n_1 and n_2 loops, but it would increase the memory bandwidth. Absent of any temporal locality of reference, the off-chip memory bandwidth will be easily saturated as we continue to parallelize. To circumvent the difficulty, we compress the off-chip memory bandwidth by an out-of-order access schedule that maximizes the temporal locality of reference. To explain the approach, note that the voxels along a line cast projections onto the same block of detector cells, thus the on-chip memory can be reused without resorting to off-chip access, as shown in Fig. 15. Based on this observation, we design an out-of-order scheduling algorithm as follows: (1) divide the detector into sectors as in Fig. 16(a); (2) draw the upper and lower edge of each sector by connecting the X-ray source and the upper and lower end of the sector; (3) determine the set of voxels whose projections lie entirely

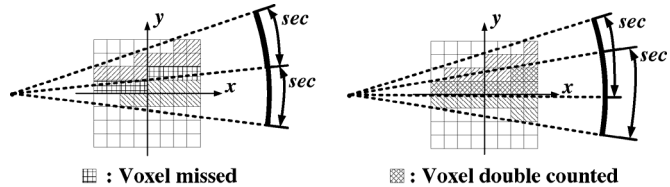


Fig. 16. Illustrations showing (a) non-overlapping sectors, and (b) overlapping sectors.

TABLE V
MOVING DIRECTIONS FOR RUN-LENGTH ENCODING

View : β (rad)	Direction
$\frac{\pi}{4}\le\beta\le\frac{3\pi}{4}$	$-n_2$
$\frac{3\pi}{4}\le\beta\le\frac{5\pi}{4}$	$+n_1$
$\frac{5\pi}{4}\le\beta\le\frac{7\pi}{4}$	$+n_2$
$0\le\beta\le\frac{\pi}{4}$ or $\frac{7\pi}{4}\le\beta\le 2\pi$	$-n_1$

in each sector. Assign the set of voxels to a projection module for processing to maximize the detector memory's locality of reference.

If we choose the sectors to be non-overlapping as in Fig. 16(a), some voxels will be missed as their projections do not completely lie in any sector. Adjacent sectors will have to overlap by an amount $(s_{\rm bin}-1)\Delta_{\rm s}$ to ensure all voxels are accounted for. (Recall that $s_{\rm bin}$ is the maximum transaxial span of a voxel's projection. An overlap of $s_{\rm bin}\Delta_{\rm s}$ or more is not necessary.) For simplicity of implementation, we choose a fixed overlap of $(s_{\rm bin}-1)\Delta_{\rm s}$ in making sectors. Now another problem arises with the choice of a fixed overlap, as some voxels will be counted twice in adjacent sectors, as shown in Fig. 16(b). To avoid double counting, we keep track of the upper and lower edge of each sector.

The out-of-order schedule can be computed in design time and stored in memory. The required memory is $w_{\rm coord}N_{\rm x}N_{\rm y}N_{\rm views}$, where $w_{\rm coord}$ is the wordlength to store the (x,y) coordinate pair. Using the sample geometry in Table II, the out-of-order schedule memory takes 796.5 MB. If we take into account the multiple rotations in a CT scan that repeat view angles and only voxels inside the *FOV*, the out-of-order schedule memory size is reduced to 86.3 MB, which is still significant.

To further shrink the out-of-order schedule memory, we design a run-length encoding to compress the schedule. The encoding scheme is illustrated in Fig. 17: we store the voxel coordinates along $edge_1$ of Sec_1, and encode and store $edge_2$ of Sec_1 as the run length from $edge_1$. $edge_2$ of Sec_1 becomes the $edge_1$ of Sec_2 and the $edge_2$ encoding follows a similar fashion. The direction to count run length depends on the view angle β, as described in Table V. For a numerical example, if we choose a sector size of $sec=20$, the out-of-order schedule memory can be compressed by an order of magnitude to 8 MB.

Table VI lists a few more example sector sizes based on the geometry in Table II. If we choose sector size $sec=20$, with a fixed sector-sector overlap of $s_{\rm bin}-1=10$, the detector is divided into 89 sectors. A sector covers an average of $N_{\rm vx}=456$ voxel columns. Sectors are processed sequentially. After finishing one sector, we move forward by a stride of $\Delta_{\rm ns}=sec-(s_{\rm bin}-1)=10$ to the new sector. The external memory access is reduced to only $\Delta_{\rm ns}$ banks every sector. When $sec=20$,

TABLE VI
SECTOR CHOICE FOR OUT-OF-ORDER SCHEDULING

sec	N_{sec}	Δ_{ns}	$N_{\text{vx,min}}$	$N_{\text{vx,max}}$	N_{vx}	On-chip memory (Kb)	$\Delta_{\text{ns}}/s_{\text{bin}}/N_{\text{vx}}$	Off-chip bandwidth (Mb/s)	Schedule memory (Mb)
14	222	4	41	341	183	15.75	0.00199	245.17	98.85
16	148	6	61	457	274	19.25	0.00199	245.17	67.05
18	111	8	96	572	365	22.75	0.00199	245.17	75.00
20	89	10	112	681	456	26.25	0.00199	245.17	60.82
30	45	20	189	1245	903	43.75	0.00201	247.63	42.12
40	30	30	330	1855	1355	61.25	0.00201	247.63	29.23
50	23	40	170	2401	1767	78.75	0.00206	253.79	28.15

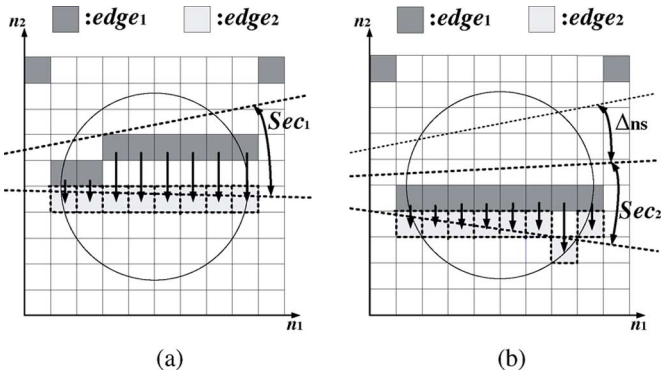


Fig. 17. Illustration of run-length encoding: (a) first sector, and (b) second sector.

the off-chip detector memory bandwidth of the proposed projection module described in the previous section is reduced to 245.2 Mb/s. As we increase the sector size, both the stride Δ_{ns} and sector coverage N_{vx} increase, resulting in an almost constant off-chip memory bandwidth. A larger sector size requires a larger on-chip memory but a smaller out-of-order schedule memory.

The out-of-order scheduling requires sectored processing. The number of on-chip detector memory banks has to be increased from s_{bin} to sec. Since a projection covers only an s_{bin} segment of the sector, a rotator and an inverse rotator are needed to select the detector memory banks. The rotator-based architecture can be implemented using multiplexers and it incurs a high routing overhead. An alternative selector-based architecture allocates sec transaxial projection blocks, and each block can be enabled or disabled by the write enable to the corresponding memory bank. The comparison between the rotator-based and the selector-based architecture is illustrated in Fig. 18 with FPGA synthesis results listed in Table VII. A selector-based architecture uses fewer logic units or FPGA slices, but more MAC units or DSP48E slices. In both architectures, a small sector size results in more efficient use of hardware.

The detector memory is dual-port to support one read and one write per cycle for the read-accumulate-write operation. To enable loading and unloading from off-chip memory without stalling the computation, we increase the number of detector memory banks from sec to $sec + \Delta_{\text{ns}}$. While sec memory banks are accessed for the projection of the current sector, the remaining Δ_{ns} banks are being unloaded/loaded to/from off-chip memory. To avoid stalling the pipeline, the loading and unloading time by the Δ_{ns} memory banks should be no greater than the time spent on the projection computation. This condition can be easily met in the proposed sectored processing.

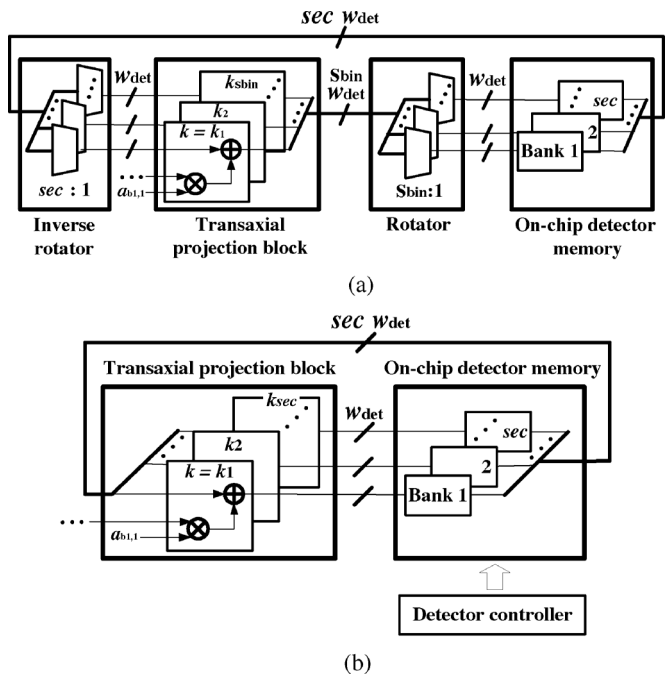


Fig. 18. Architectures supporting sectored processing: (a) rotator-based architecture, and (b) selector-based architecture.

TABLE VII
FPGA RESOURCE UTILIZATION OF A FORWARD-PROJECTION MODULE SUPPORTING SECTORED PROCESSING BASED ON XC5VLX155T DEVICE

	$sec = 14$		$sec = 20$	
	Rotator	Selector	Rotator	Selector
FPGA slice register	11,120	11,487	11,494	12,250
FPGA slice LUT	12,335	11,060	15,028	11,809
Occupied FPGA slice	6,347	6,166	7,132	6,522
BRAM	39	39	45	45
DSP48E	17	20	17	26

V. FPGA IMPLEMENTATION

A complete forward-projection module is shown in Fig. 19. Inputs are read from the image memory, held by the water-filling buffer before being processed by the partially-unrolled axial projection block. Transaxial projections are performed in parallel and the results are accumulated in the detector memory. A selector-based architecture orchestrates sectored processing following an out-of-order schedule. A summary of the architecture metrics is listed in Table VIII.

The projection module has been mapped to a Xilinx Virtex-5 XC5VLX155T FPGA [14] and the device utilization is listed in Table IX. We followed the sample geometry in Table II and chose a small sector size $sec = 14$ with $\Delta_{\text{ns}} = 4$. The projection

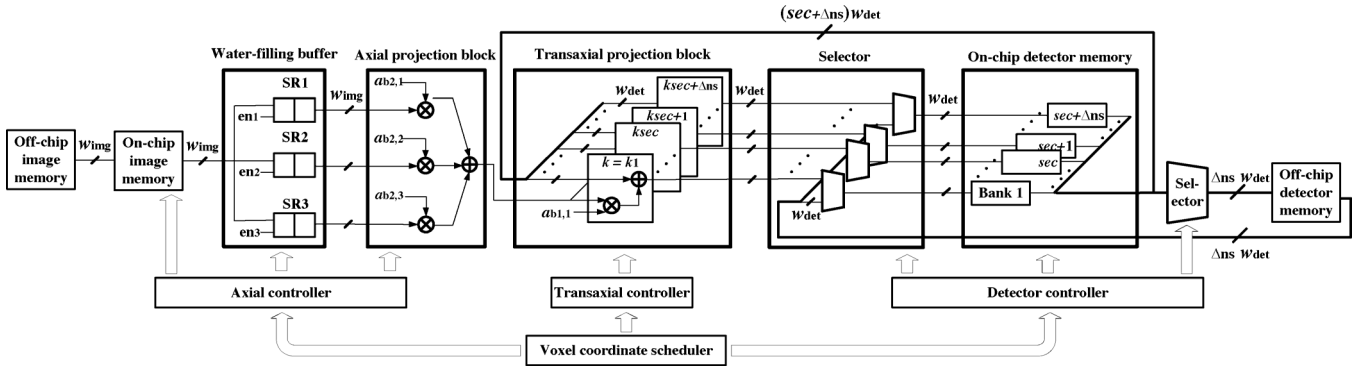


Fig. 19. Complete selector-based forward-projection module supporting sectored processing.

TABLE VIII
ARCHITECTURE METRICS OF A FORWARD-PROJECTION MODULE SUPPORTING SECTORED PROCESSING

On-chip image memory bandwidth	$w_{\mathrm{img}} f_{\mathrm{clk}}$ [b/s]
Off-chip image memory bandwidth	$w_{\mathrm{img}} f_{\mathrm{clk}}$ [b/s]
On-chip detector memory bandwidth	$2 s_{\mathrm{bin}} w_{\mathrm{det}} f_{\mathrm{clk}}$ [b/s]
Off-chip detector memory bandwidth	$2\Delta_{\mathrm{ns}} w_{\mathrm{det}} f_{\mathrm{clk}} / N_{\mathrm{vx}}$ [b/s]
On-chip image memory banks	1
On-chip detector memory banks	$sec + \Delta_{\mathrm{ns}}$
MAC units	$sec + \Delta_{\mathrm{ns}} + z_{\mathrm{vx}}$
Throughput	$f_{\mathrm{clk}}(1 - P_{\mathrm{stall}})$ [voxel projs/s]

TABLE IX
FPGA RESOURCE UTILIZATION OF COMPLETE FORWARD-PROJECTION MODULES BASED ON XILINX VIRTEX-5 XC5VLX155T DEVICE

	single module		5× parallel modules	
	Usage	Utilization ratio	Usage	Utilization ratio
FPGA slice register	12,077	12%	30,323	31%
FPGA slice LUT	11,939	12%	31,874	32%
Occupied FPGA slice	6,328	26%	14,243	58%
BRAM	43	20%	117	55%
DSP48E	24	18%	108	84%

module uses 24 DSP48E slices as MAC units, 43 block RAMs as on-chip memory banks, and occupies 6,328 FPGA slices. Note that the resource usage includes a fixed overhead created to handle interfaces to the FPGA and controls. At a 200 MHz clock frequency, the off-chip input image memory bandwidth is 2.6 Gb/s and the off-chip output detector memory bandwidth is compressed to 245.2 Mb/s. Additional memory access is needed to load the out-of-order schedule, but the bandwidth is very low as only one pair of coordinates is read per column of voxels and the coordinates have been compressed using run-length encoding. The projection module is fully pipelined and capable of completing up to $s_{\mathrm{bin}} = 11$ projections per clock cycle for an average throughput of 185.2 million voxel projections/s at $f_{\mathrm{clk}} = 200$ MHz.

The substantially reduced off-chip memory bandwidth allows us to parallelize the design further by multiple projection modules. The Xilinx Virtex-5 XC5VLX155T FPGA can accommodate 5 parallel projection modules, and the device utilization is shown in Table IX. The parallel projection modules will be assigned to non-adjacent sectors, so they will be able to operate independently for a 55-way parallel computation towards a combined average throughput of 925.8 million voxel projections/s at $f_{\mathrm{clk}} = 200$ MHz. The 55-way parallel forward-projector is integrated with two DDR400 64-bit DRAM channels that each provides up to 25.6 Gb/s off-chip memory interface. One DRAM channel is used as the off-chip image memory and the other as the off-chip detector memory. This 55-way parallel design completes one forward-projection of a $320 \times 320 \times 61$ test object over 3,625 views in 6.31 seconds. The same task implemented in C requires 31.1 seconds of execution time on an 8-core 2.8-GHz Intel processor for a throughput of 203.0 million voxel projections/s. The C program uses 16 threads, and is optimized based on the projection geometry.

VI. CONCLUSION

We present algorithm and architecture techniques to construct a highly efficient hardware-based forward-projection for iterative image reconstruction. The solutions are based on a study of the projection geometry which uncovers loop-level parallelism, locality of reference, as well as geometric mismatch between the object grid and the projection grid. We exploit loop-level parallelism and spatial locality of reference to unroll inner loops for a high throughput. However, geometric mismatches and off-chip memory access bottleneck limit the achievable throughput. A water-filling buffer is thus created to bridge the geometric mismatch and remove the pipeline stalls, and an out-of-order schedule is designed to compress the off-chip memory access. The cost of implementing these schemes is kept low by judicious considerations of buffer length used in the water-filling buffer, sector size and architecture used in the sectored processing, as well as run-length encoding designed to compress the out-of-order schedule memory.

The resulting architecture is fully pipelined and can be parallelized for a very high throughput. We demonstrate the design in a 5-stage pipelined, 55-way parallel forward-projector implemented on a Xilinx Virtex-5 XC5VLX155T FPGA that achieves an average throughput of 925.8 million voxel projections/s at a clock frequency of 200 MHz. Note that the throughput is limited by the number of MAC units available on this device, as a Virtex-5 XC5VLX155T FPGA contains only 128 DSP48E slices. The latest Xilinx Virtex-7 devices offer up to 3,600 DSP slices [20], which will allow for a much higher throughput potential.

The proposed architecture can be adopted for back-projection for a complete iterative image reconstruction system, which is part of our future work. Testing fixed-point quantization of

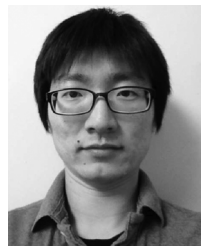
higher-resolution images also remains our future work. The proposed algorithm and architecture techniques also apply to designs that are built on alternative hardware platforms, such as GPU and DSP to achieve significant accelerations.

ACKNOWLEDGMENT

The authors would like to thank Donghwan Kim and Yong Long for helpful discussions and acknowledge the equipment donation from BEEcube, Xilinx and Intel.

REFERENCES

[1] L. A. Feldkamp, L. C. Davis, and J. W. Kress, "Practical cone-beam algorithm," *J. Opt. Soc. Amer. A*, vol. 1, no. 6, pp. 612–619, 1984.
[2] J. A. Fessler, "Statistical image reconstruction methods for transmission tomography," in *Handbook of Medical Imaging*. Bellingham, WA: SPIE, 2000, vol. 2, Medical Image Processing and Analysis, pp. 1–70.
[3] T. M. Buzug, *Computed Tomography: From Photon Statistics to Modern Cone-Beam CT*. New York: Springer-Verlag, 2009.
[4] S. Kawata and O. Nalcioglu, "Constrained iterative reconstruction by the conjugate gradient method," *IEEE Trans. Med. Imag.*, vol. 4, no. 2, pp. 65–71, 1985.
[5] Z. Q. Luo and P. Tseng, "On the convergence of the coordinate descent method for convex differentiable minimization," *J. Optim. Theory Appl.*, vol. 72, no. 1, pp. 7–35, 1992.
[6] H. Erdogen and J. A. Fessler, "Ordered subsets algorithms for transmission tomography," *Phys. Med. Biol.*, vol. 44, pp. 2835–2851, 1999.
[7] Y. Long, J. A. Fessler, and J. M. Balter, "3D forward and back-projection for X-ray CT using separable footprints," *IEEE Trans. Med. Imag.*, vol. 29, no. 11, pp. 1839–1850, 2010.
[8] J. Kim, Z. Zhang, and J. A. Fessler, "Hardware acceleration of iterative image reconstruction for X-ray computed tomography," in *Proc. IEEE Conf. Acoust., Speech, Signal Process.*, May 2011, pp. 1697–1700.
[9] F. Xu and K. Mueller, "Real-time 3D computed tomographic reconstruction using commodity graphics hardware," *Phy. Med. Biol.*, vol. 52, pp. 3405–3419, 2007.
[10] F. Xu and K. Mueller, "Accelerating popular tomographic reconstruction algorithms on commodity PC graphics hardware," *IEEE Trans. Nucl. Sci.*, vol. 52, no. 3, pp. 654–663, 2005.
[11] J. Sanders and E. Kandrot, *CUDA by Example: An Introduction to General-Purpose GPU Programming*. Reading, MA: Addison-Wesley Professional, 2010.
[12] I. Goddard and M. Trepanier, "High-speed cone-beam reconstruction: An embedded systems approach," in *Proc. SPIE*, Feb. 2002, vol. 4681, pp. 483–491.
[13] J. Xu, N. Subramanian, A. Alessio, and S. Hauck, "Impulse C vs. VHDL for accelerating tomographic reconstruction," in *Proc. IEEE Symp. Field-Programmable Custom Comput. Mach.*, 2010, pp. 171–174.
[14] Xilinx Corporation, Virtex-5 FPGA Family [Online]. Available: http://www.xilinx.com/products/virtex5/index.htm
[15] E. Seeram, *Computed Tomography: Physical Principles, Clinical Applications, and Quality Control*. Philadelphia, PA: Saunders Elsevier, 2009.
[16] J. H. Siewerdsen and D. A. Jaffray, "Cone-beam computed tomography with a flat-panel imager: Effects of image lag," *Med. Phys.*, vol. 26, pp. 1624–1641, 1999.
[17] D. A. Jaffray, J. H. Siewerdsen, J. W. Wong, and A. A. Martinez, "Flat-panel cone-beam computed tomography for image-guided radiation therapy," *Int. J. Radiat. Oncol. Biol. Phys.*, vol. 53, pp. 1337–1349, 2002.
[18] J. A. Fessler, *Image Reconstruction: Algorithms and Analysis,* book in preparation.
[19] 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, pp. 4526–4544, 2007.
[20] Xilinx Corporation, Virtex-7 FPGA Family [Online]. Available: http://www.xilinx.com/products/silicon-devices/fpga/virtex-7/index.htm



Jung Kuk Kim (S'10) received the B.S. degree in electrical engineering from the Korea Advanced Institute of Science and Technology, Daejeon, South Korea, in 2009 and the M.S. degree in electrical engineering from the University of Michigan, Ann arbor, in 2011.

He is currently working towards the Ph.D. degree in electrical engineering at the University of Michigan, Ann Arbor. His research interests are in algorithm and architecture co-optimization, and image reconstruction algorithms for X-ray CT.

Jeffrey A. Fessler (F'06) received the B.S.E.E. degree from Purdue University, West Lafayette, IN, in 1985, the M.S.E.E. degree and the M.S. degree in statistics both from Stanford University, Stanford, CA, in 1986 and 1989, respectively. From 1985 to 1988, he was a National Science Foundation Graduate Fellow at Stanford, where he received the Ph.D. degree in electrical engineering in 1990.

He has worked at the University of Michigan since then. From 1991 to 1992, he was a Department of Energy Alexander Hollaender Post-Doctoral Fellow in the Division of Nuclear Medicine. From 1993 to 1995, he was an Assistant Professor in the Nuclear Medicine and the Bioengineering Program. He is currently a Professor in the Departments of Electrical Engineering and Computer Science, Radiology, and Biomedical Engineering. He became a Fellow of the IEEE in 2006, for contributions to the theory and practice of image reconstruction. His research interests are in statistical aspects of imaging problems, and he has supervised doctoral research in PET, SPECT, X-ray CT, MRI, and optical imaging problems.

Dr. Fessler received the Francois Erbsmann award for his IPMI93 presentation. He serves as an Associate Editor for the IEEE TRANSACTIONS ON MEDICAL IMAGING and is a former Associate Editor for the IEEE TRANSACTIONS ON IMAGE PROCESSING and the IEEE SIGNAL PROCESSING LETTERS. He was Co-Chair of the 1997 SPIE Conference on Image Reconstruction and Restoration, Technical Program Co-Chair of the 2002 IEEE International Symposium on Biomedical Imaging (ISBI), and General Chair of ISBI 2007.

Zhengya Zhang (S'02–M'09) received the B.A.Sc. degree in computer engineering from the University of Waterloo, Canada, in 2003 and the M.S. and Ph.D. degrees in electrical engineering from the University of California, Berkeley (UC Berkeley), in 2005 and 2009, respectively.

Since September 2009, he has been on the faculty of the University of Michigan, Ann Arbor, where he is an Assistant Professor of electrical engineering and computer science.

Dr. Zhang received the NSF CAREER Award in 2011, the David J. Sakrison Memorial Prize from UC Berkeley Electrical Engineering and Computer Science Department in 2009, the Best Student Paper Award at the Symposium on VLSI Circuits in 2009, an Analog Devices Outstanding Student Designer Award, and a Vodafone U.S. Foundation Fellowship.