Hardware Acceleration of Iterative Image Reconstruction for X-Ray Computed Tomography

Jung Kuk Kim, Zhengya Zhang, Jeffrey A. Fessler
EECS Department, University of Michigan, Ann Arbor, MI 48109-2122, USA

ABSTRACT

X-ray computed tomography (CT) images could be improved using iterative image reconstruction if the 3D cone-beam forward- and back-projection computations can be accelerated significantly. We investigated the feasibility of a field-programmable gate array (FPGA) implementation of the separable footprint (SF) forward projector. A 16-bit fixed-point quantization introduces negligible numerical errors without affecting the perceptual image quality. The SF-based 3D cone-beam projector can be efficiently parallelized and its memory bandwidth reduced by exploiting projection geometry and data locality. We demonstrate a fully pipelined, 75-way parallel hardware architecture of the SF forward projector on a Xilinx Virtex-5 FPGA that can complete one forward projection of a 320×320×61 object over 3,625 views in 6.3 seconds.

1. INTRODUCTION

Recently there is increasing interest in using iterative image reconstruction algorithms for X-ray computed tomography (CT) to improve image quality and reduce dose [1]. These algorithms perform forward- and back-projection in a feedback loop to minimize a cost function, increasing computation substantially over the conventional filtered backprojection (FBP) algorithm [2]. In particular, the evaluation of a discrete version of the Radon transform [3] in the forward projection is one of the main compute bottlenecks that impede routine clinical use of iterative algorithms for CT.

The recently proposed separable footprint (SF) projector simplifies the forward projection by approximating the voxel footprint functions as 2D separable functions [4]. The SF projector has high accuracy and favorable speed, but the computational cost is still high: each forward- and back-projection requires on the order of 100 billion floating-point multiply-accumulate (MAC) operations for a practical CT scanner, requiring almost 10 minutes of execution time on a 2.4 GHz quad-core microprocessor.

Graphics processing unit (GPU) has recently been demonstrated to achieve 10 to 100 times speedup over a microprocessor for CT image reconstruction [4]-[6]. These GPU implementations were based on the Feldkamp (FDK) method [2], and challenges still exist in implementing the iterative algorithms.

In this paper, we explore field-programmable gate array (FPGA) as a viable alternative to GPU for CT image reconstruction. FPGA offers efficient integer operations and more flexibility in hardware architecture mapping. We propose a fixed-point quantization of the SF forward projector that substantially reduces the memory usage and computational complexity. We conducted numerical analysis to determine the optimal quantization that satisfies a given error metric. The final fixed-point SF forward projector was mapped to a Xilinx Virtex-5 FPGA [8]. This highly parallel forward projector increases the throughput and its memory bandwidth is reduced by a scheduling algorithm that exploits projection geometry and data locality. The resulting design can accelerate the forward projection by almost an order of magnitude over a quad-core microprocessor.

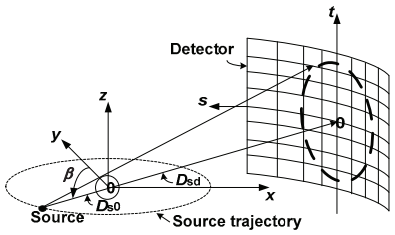


Fig. 1. Axial cone-beam arc-detector geometry for X-ray CT.

2. BACKGROUND

We illustrate the projection geometry of an X-ray CT system in Fig. 1 [4]. We denote (x, y, z) as the coordinate of a voxel of the object being imaged. 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. For each angle β, the source emits X-rays that project the object onto the detector. We denote (s, t) as the coordinate of a detector cell. The transaxial direction s is perpendicular to z and the axial direction t is parallel to z.

Fig. 2 shows the block diagram of an iterative image reconstruction algorithm. An object is first imaged from many view angles to produce the measured sinogram (a set of projections over all view angles) of the object. The FBP algorithm is used to estimate the initial image, followed by the computed forward projection to obtain the computed sinogram. The error between the computed and measured sinogram is back-projected and the result is used to improve the initial image estimate. The image and sinogram are iteratively updated to minimize the error, and regularization is included to control noise [1]. In this paper, we focus on the compute-intense forward projection, mathematically described by [4]:

$$g(s,t,\beta) = \sum_{x,y,z} a(s,t,\beta;x,y,z) f(x,y,z), \qquad (1)$$

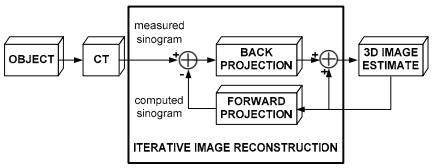


Fig. 2. Block diagram of an iterative image reconstruction algorithm.

where $g(s, t, \beta)$ is the projection, $f(x, y, z)$ is the voxel value at (x, y, z), and $a(s, t, \beta; x, y, z)$ is the footprint function of the voxel centered at (x, y, z). The separable footprint method approximates the footprint function as in [4]:

$$a(s,t,\beta;x,y,z) \approx v_1(s,t,\beta)u_2(\beta;x,y)F_1(s,\beta;x,y)F_2(t,\beta;x,y,z), \quad (2)$$

where $v_1(s, t, \beta)$ and $u_2(\beta; x, y)$ are scaling factors, $F_1(s, \beta; x, y)$ is the footprint function in the s direction with a trapezoidal profile, and $F_2(t, \beta; x, y, z)$ is the footprint function in the t direction with a rectangular profile. The footprints in s and t can be separated and independently evaluated, simplifying (1) to:

$$g(s,t,\beta) \approx v_1(s,t,\beta) \cdot \sum_{x,y} u_2(\beta;x,y)F_1(s,\beta;x,y)\sum_z F_2(t,\beta;x,y,z)f(x,y,z). \quad (3)$$

To compute one forward projection over all view angles, we need to perform multiplication and summation over six layers of nested loops: β, x, y, z, t and s. For a practical object made up of more than 10 million voxels, a forward projection that comprises more than 3,000 view angles, as in a commercial CT scanner, requires on the order of 100 billion multiply-accumulate (MAC) operations. We propose three approaches below to simplify this computation: (1) a fixed-point quantization that reduces the cost of each MAC operation and the memory usage; (2) an efficient parallel hardware architecture that unrolls the loops; and (3) a scheduling algorithm that reduces the memory bandwidth.

3. FIXED-POINT QUANTIZATION

CT image reconstruction algorithms are usually implemented in 32-bit single-precision floating-point quantization. Floating-point operations cost more hardware resources and longer latency than integer operations. For example, on a Xilinx Virtex-5 FPGA, a 32-bit floating-point multiply-accumulate operation costs 5 DSP48E slices and 17 clock cycles to complete, while a 16-bit fixed-point multiply-accumulate operation costs only 1 DSP48E slice and 1 clock cycle [9]. The substantially smaller area and higher speed provide strong incentives for applying the floating-to-fixed-point conversion.

We investigated fixed-point quantization of the SF forward projector using a 61-slice test object, where each slice is composed of 320×320 voxels. The baseline reference is the image reconstructed using 32-bit single-precision floating-point quantization after 1,000 iterations. Fig. 3 shows the error measured in mean absolute error (MAE) and root-mean-square error (RMSE) in Hounsfield units (HU) after

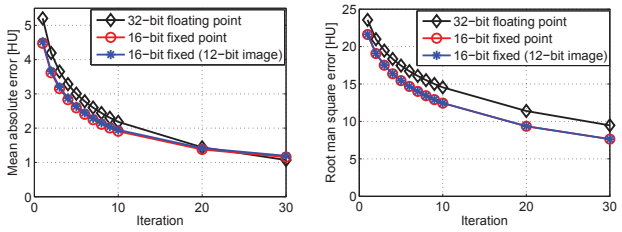


Fig. 3. (a) Mean absolute error and (b) mean square error with each iteration.

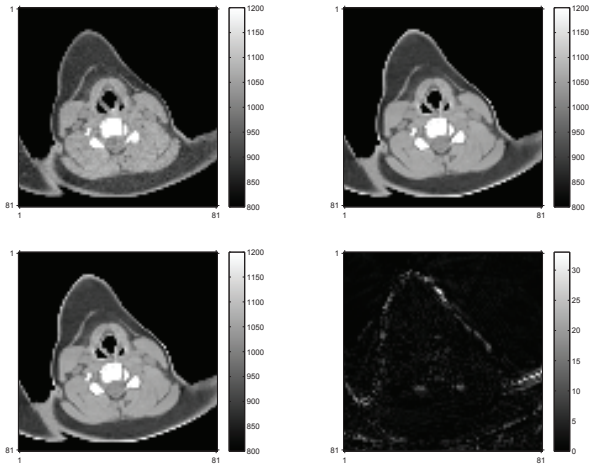


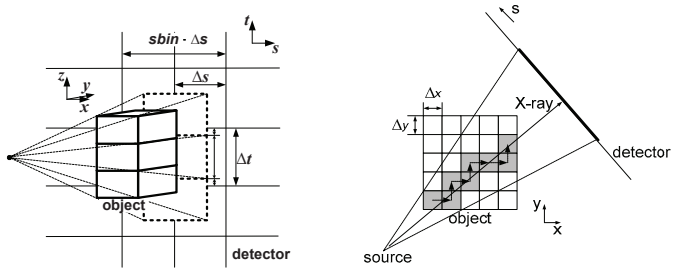
Fig. 4. Original FBP image (top left, only the region of interest is shown), reconstructed image using 32-bit floating-point quantization with 30 iterations (top right), reconstructed image using 16-bit fixed-point quantization with 30 iterations (bottom left), and absolute errors between the floating-point and the fixed-point quantization (bottom right).

each iteration. The performance of the 32-bit floating-point and a much shorter fixed-point quantization (16-bit image, 16-bit arithmetic, and 18-bit projection) are close. Interestingly, the fixed-point quantization allows the algorithm to converge faster initially for this image, but the floating-point quantization eventually improves after 20 iterations. To reach a given error criterion, e.g., an MAE of 1.5 HU, the fixed-point quantization takes only 20 iterations and the final RMSE is below 10 HU. These initial results suggest that the iterative image reconstruction algorithm is robust to quantization noise.

If we continued to reduce the word length, e.g., by using shorter than 12-bit words for image, both MAE and RMSE would increase considerably. So we set the final word length and quantization such that the MAE at 30 iterations was within 1.2 HU. To verify the perceptual quality, Fig. 4 shows the original image obtained by FBP along with the reconstructed images using 32-bit floating-point quantization and the final fixed-point quantization (12-bit image, 16-bit arithmetic, and 18-bit projection), as well as the error map. We observe no perceptual difference between the two.

4. ARCHITECTURAL DESIGN

A simple projector architecture would include image memo-



(a) (b)

Fig. 5. (a) Projection of one column of voxels onto the detector; (b) voxel access order following the projection ray.

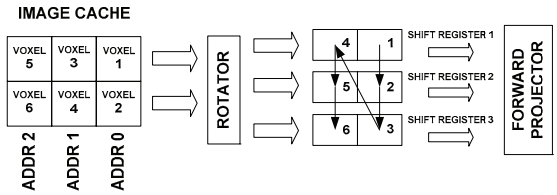


Fig. 6. Input data flow from cache to shift registers.

ry on the input and detector memory on the output. For a commercial CT scanner, the image and detector memory are on the order of 100 MB to 1 GB in size. Such enormous sizes can only be accommodated in off-chip main memory, and inputs are selectively brought to on-chip memory (cache memory) for processing. To improve throughput, the computation needs to be parallelized, but parallel architectures demand high memory bandwidth. We propose a memory-efficient parallel architecture. This architecture is described based on a 320×320×61 test object, and it could be easily adapted to other standard object sizes.

4.1. Parallel and fully-pipelined processing

Each of the six nested loops of the forward projection algorithm can be unrolled for parallel processing, but the impact on the main memory traffic varies. Unrolling the *s* loop, i.e., parallel processing along the neighboring detector cells in one row, takes advantage of data locality, as the neighboring cells are normally cached together. We estimate the width, *sbin* shown in Fig. 5(a), of a voxel's projection in a given detector row as:

$$sbin \le \left\lceil \sqrt{2}\frac{\Delta_x}{\Delta_s}\frac{D_{sd}}{D_{s0}-FOV/2} \right\rceil = 15, \qquad (4)$$

where *FOV*, or field of view, is the largest diameter of an object that can be projected from all view angles, D_{sd} and D_{s0} are the source-to-detector distance and the source-to-rotation-center distance shown in Fig. 1, Δ_x refers to the object grid size, and Δ_s the detector grid size. The maximum *sbin* is 15 for the test object under consideration. Following this derivation, the *s* loop can be efficiently unrolled to 15 parallel computing modules.

A higher-degree parallelism can be achieved by unrolling the *t* loop and *z* loop. In fact, these two loops can be merged. The projection geometry dictates that the maximum number of voxels in a column that project to a single detector row is $L_{max} = 3$ for the test object, as shown in (5) and illustrated in Fig. 5(a), while the average number of voxels in a column that project to a single detector row is $L_{avg} = 2$.

$$L_{max} = 1 + \left\lceil \frac{D_{s0}+FOV}{D_{s0}} \right\rceil = 3. \qquad (5)$$

We exploit this geometry to merge the *z* and *t* loops. Since $L_{avg} = 2$, the input voxels are stored in cache in pairs as shown in Fig. 6. The input pairs of voxels are sliced and then distributed by a rotator to two of the three shift registers for data alignment. The three shift registers accommodate $L_{max} = 3$ for one *t* computation. A few cycles of prefetch are needed to start filling the shift registers. After completing one *t* computation, the voxels that are no longer needed are shifted out from the registers and replenished from the cache. The structure shown in Fig. 6 bridges the geometric mismatch between the image and detector grids and enables a fully-pipelined processing.

4.2. Memory-efficient scheduling

We could further parallelize the forward projection by unrolling the *x* and *y* loops, but it would cause potential memory contention because voxels in different (*x*, *y*) locations can project to the same detector cell. To avoid potential access conflicts, we have parallel projectors operate on (*x*, *y*) locations that are further apart. This scheduling enables a stream architecture with very simple memory controllers.

The contention-free scheduling inevitably increases the output to main memory traffic. To reduce the traffic, we take advantage of the overlap between the projections of neighboring voxels. The (*x*, *y*) access order for each projector is decided such that the projections of voxels at consecutive (*x*, *y*) locations share the maximum overlap to reduce the cache size. Fig. 5(b) illustrates one efficient (*x*, *y*) access order that follows the direction of the X-ray. The span of the series of projections over *s* is given by:

$$sbin \le \left\lceil \sqrt{5}\frac{\Delta_x}{\Delta_s}\frac{D_{sd}}{D_{s0}-FOV/2} \right\rceil = 23. \qquad (6)$$

This span of detector memory will be accessed in processing the strip of all the voxels at (*x*, *y*) locations along the X-ray, as highlighted in Fig. 5(b). The average number of (*x*, *y*) locations along a strip is on the order of several hundred for practical objects. For the test image, this scheduling algorithm can reduce the output traffic to the main memory by approximately four hundred times.

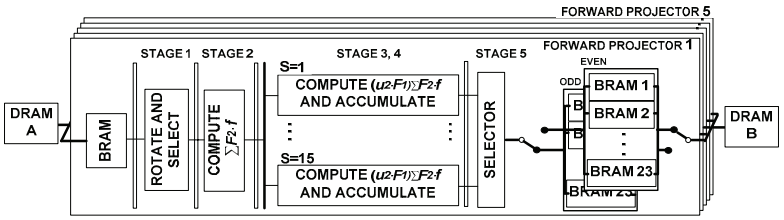


Fig. 7. 5-stage fully-pipelined architecture for SF forward projector.

TABLE I FPGA RESOURCE UTILIZATION BASED ON XILINX VIRTEX-5 XC5VLX155T DEVICE

	Usage	Utilization ratio
Slice register	37,939	39%
Slice LUT	35,994	37%
BRAM	192	93%
DSP48E	93	72%

5. IMPLEMENTATION RESULTS

We prototyped the proposed architecture and memory scheduling algorithms on a Xilinx Virtex-5 XC5VLX155T FPGA device. The device provides 128 DSP48E slices and its 7-Mb block RAMs (BRAM) can be used as the on-chip cache. Two DDR400 64-bit channels of the device provide the access to 4 GB of off-chip main memory.

We designed a 5-stage pipelined architecture for the SF forward projection as shown in Fig. 7. The design was synthesized to operate at a maximum clock frequency of 200 MHz. At this frequency, each DRAM channel is able to transfer 128 bits of data per cycle. Based on the fixed-point quantization of 12 bits per voxel, one DRAM channel is capable of feeding 5 parallel forward projectors, each of which updates a maximum of 15 detector cells in parallel.

DRAM channel A provides the image inputs that are loaded to the cache memory before processing. In the first pipeline stage, the input voxels are fetched from the cache to the shift registers, as described in Fig. 6. In the second stage, the weighted sum $\sum_z F_2(t, \beta; x, y, z) f(x, y, z)$ is computed. In the third and fourth stages, up to 15 parallel computing modules are activated in parallel to compute $u_2(\beta; x, y) F_1(s, \beta; x, y) \sum_z F_2(t, \beta; x, y, z) f(x, y, z)$ for the neighboring s values, followed by the accumulation to update $g(s, t, \beta)$. Each projection accesses one of the 23 BRAMs. After completing the projection along one strip shown in Fig. 5(b), the current bank (odd bank) of 23 BRAMs is disconnected from the projector and connected to the DRAM channel B for write back and loading. At the same time, the other bank (even bank) of 23 BRAMs is connected to the projector for the next set of projections. The odd and even banks are accessed in a ping-pong manner. A memory controller coordinates the input and output access orders.

The FPGA resource utilization is listed in Table I. The DSP48E and BRAM utilization are high to support the highly parallel processing and the necessary memory bandwidth. At a 200 MHz clock frequency, this design can complete one forward projection of a 320×320×61 test object over 3,625 views in 6.31 seconds. The same task requires 52.5 seconds of execution time on a 2.4 GHz quad-core microprocessor.

6. CONCLUSION

We demonstrated the feasibility of accelerating the SF forward projection on FPGA. A fixed-point quantization improves the computational efficiency while maintaining the image quality. The hardware architecture can be efficiently parallelized without hitting the memory bottleneck. A prototype 75-way parallel design on FPGA increases the throughput of forward projection by almost an order of magnitude over a quad-core microprocessor.

ACKLOWEDGEMENT

This work was supported in part by a Korea Foundation for Advanced Studies (KFAS) Scholarship and NIH grant R01 HL 098686. The authors would like to thank Yong Long for helpful discussions and acknowledge the equipment donation from BEEcube and Xilinx.

REFERENCES

[1] J. A. Fessler, "Statistical image reconstruction methods for transmission tomography," *Handbook of Medical Imaging, Volume 2. Medical Image Processing and Analysis*, Bellingham: SPIE, 2000, pp. 1-70.
[2] L. A. Feldkamp *et al.*, "Practical cone beam algorithm," *J. Opt. Soc. Am. A*, vol. 1, no. 6, pp. 612-619, Jun. 1984.
[3] S. R. Deans, *The Radon Transform and Some of Its Applications*. New York: Wiley, 1983.
[4] Y. Long *et al.*, "3D forward and back-projection for X-ray CT using separable footprints," *IEEE Trans. Med. Imag.*, vol. 29, no. 11, pp. 1839-50, Nov. 2010.
[5] W. Bi *et al.*, "Accelerate helical cone-beam CT with graphics hardware," *Proc. SPIE* 6913, 69132T (2008).
[6] P. B. Noel *et al.*, "Clinical evaluation of GPU-based cone beam computed tomography," *Proc. HP-MICCAI*, 2008.
[7] F. Xu and K. Mueller, "Real-time 3D computed tomographic reconstruction using commodity graphics hardware," *Phy. Med. Biol.*, vol. 52, pp. 3405-3419, 2007.
[8] *Virtex-5 FPGA Family* [Online]. Available: http://www.xilinx.com/products/virtex5/index.htm
[9] *Floating-Point Operator* [Online]. Available: http://www.xilinx.com/products/ipcenter/FLOATING_PT.htm