

# Parallel-Beam Backprojection: an FPGA Implementation Optimized for Medical Imaging

Miriam Leeser, Srdjan Coric, Eric Miller, Haiqian Yu

Department of Electrical and Computer Engineering  
Northeastern University  
Boston, MA 02115

{mel, scoric, elmiller, hyu}@ece.neu.edu

Marc Trepanier

Mercury Computer Systems, Inc.  
Chelmsford, MA 01824

mtrepanier@mc.com

## ABSTRACT

Medical image processing in general and computerized tomography (CT) in particular can benefit greatly from hardware acceleration. This application domain is marked by computationally intensive algorithms requiring the rapid processing of large amounts of data. To date, reconfigurable hardware has not been applied to the important area of image reconstruction. For efficient implementation and maximum speedup, fixed-point implementations are required. The associated quantization errors must be carefully balanced against the requirements of the medical community. Specifically, care must be taken so that very little error is introduced compared to floating-point implementations and the visual quality of the images is not compromised. In this paper, we present an FPGA implementation of the parallel-beam backprojection algorithm used in CT for which all of these requirements are met. We explore a number of quantization issues arising in backprojection and concentrate on minimizing error while maximizing efficiency. Our implementation shows approximately 100 times speedup over software versions of the same algorithm running on a 1GHz Pentium, and is more flexible than an ASIC implementation. Our FPGA implementation can easily be adapted to both medical sensors with different dynamic ranges as well as tomographic scanners employed in a wider range of application areas including nondestructive evaluation and baggage inspection in airport terminals.

Keywords:

Backprojection, Medical Imaging, Tomography, FPGA, Fixed Point Arithmetic

# 1. INTRODUCTION

Reconfigurable hardware offers significant potential for the efficient implementation of a wide range of computationally intensive signal and image processing algorithms. The advantages of utilizing Field Programmable Gate Arrays (FPGAs) instead of DSPs include reductions in the size, weight, performance and power required to implement the computational platform. FPGA implementations are also preferred over ASIC implementations because FPGAs have more flexibility and lower cost. To date, the full utility of this class of hardware has gone largely unexplored and unexploited for many mainstream applications. In this paper, we consider a detailed implementation and comprehensive analysis of one of the most fundamental tomographic image reconstruction steps, backprojection, on reconfigurable hardware. While we concentrate our analysis on issues arising in the use of backprojection for medical imaging applications, both the implementation and the analysis we provide can be applied directly or easily extended to a wide range of other fields where this task needs to be performed. This includes remote sensing and surveillance using synthetic aperture radar and non-destructive evaluation.

Tomography refers to the process that generates a cross-sectional or volumetric image of an object from a series of projections collected by scanning the object from many different directions [1]. Projection data acquisition can utilize X-rays, magnetic resonance, radioisotopes, or ultrasound. The discussion presented here pertains to the case of two-dimensional X-ray absorption tomography. In this type of tomography, projections are obtained by a number of sensors that measure the intensity of X-rays travelling through a slice of the scanned object. The radiation source and the sensor array rotate around the object in small increments. One projection is taken for each rotational angle. The image reconstruction process uses these projections to calculate the average X-ray attenuation coefficient in cross-sections of a scanned slice. If different structures inside the object induce different levels of X-ray attenuation, they are discernible in the reconstructed image.

The most commonly used approach for image reconstruction from dense projection data (many projections, many samples per projection) is filtered backprojection (FBP). Depending on the type of X-ray source, FBP comes in parallel-beam and fan-beam variations [1]. In this paper, we focus on parallel-beam backprojection, but methods and results presented here can be extended to the fan-beam case with modifications.

FBP is a computationally intensive process. For an image of size  $n \times n$  being reconstructed with  $n$  projections, the complexity of the backprojection algorithm is  $O(n^3)$ . Image reconstruction through backprojection is a highly parallelizable process. Such applications are good candidates for implementation in Field Programmable Gate Array (FPGA) devices since they provide fine-grained parallelism and the ability to be customized to the needs of a particular implementation. We have implemented backprojection by making use of these principles and shown approximately 100 times speedup over a software implementation on a 1GHz Pentium. Our architecture can easily be expanded to newer and larger FPGA devices, further accelerating image generation by extracting more data parallelism.

A difficulty of implementing FBP is that producing high-resolution images with good resemblance to internal characteristics of the scanned object requires that both the density of each projection and their total number be large. This represents a considerable challenge for hardware implementations, which attempt to maximize the parallelism in the implementation. Therefore, it can be beneficial to use fixed-point implementations and to optimize the bit-width of a projection sample to the specific needs of the targeted application domain. We show this for medical imaging, which exhibits distinctive properties in terms of required fixed-point precision.

In addition, medical imaging requires high precision reconstructions since visual quality of images must not be compromised. We have paid special attention to this requirement by carefully analyzing the effects of quantization on the quality of reconstructed images. We have found that a fixed-point implementation with properly chosen bit-widths can give high quality reconstructions and, at the same time, make hardware implementation fast and area efficient. Our quantization analysis investigates algorithm specific and also general

data quantization issues that pertain to input data. Algorithm specific quantization deals with the precision of spatial address generation including the interpolation factor, and also investigates bit reduction of intermediate results for different rounding schemes.

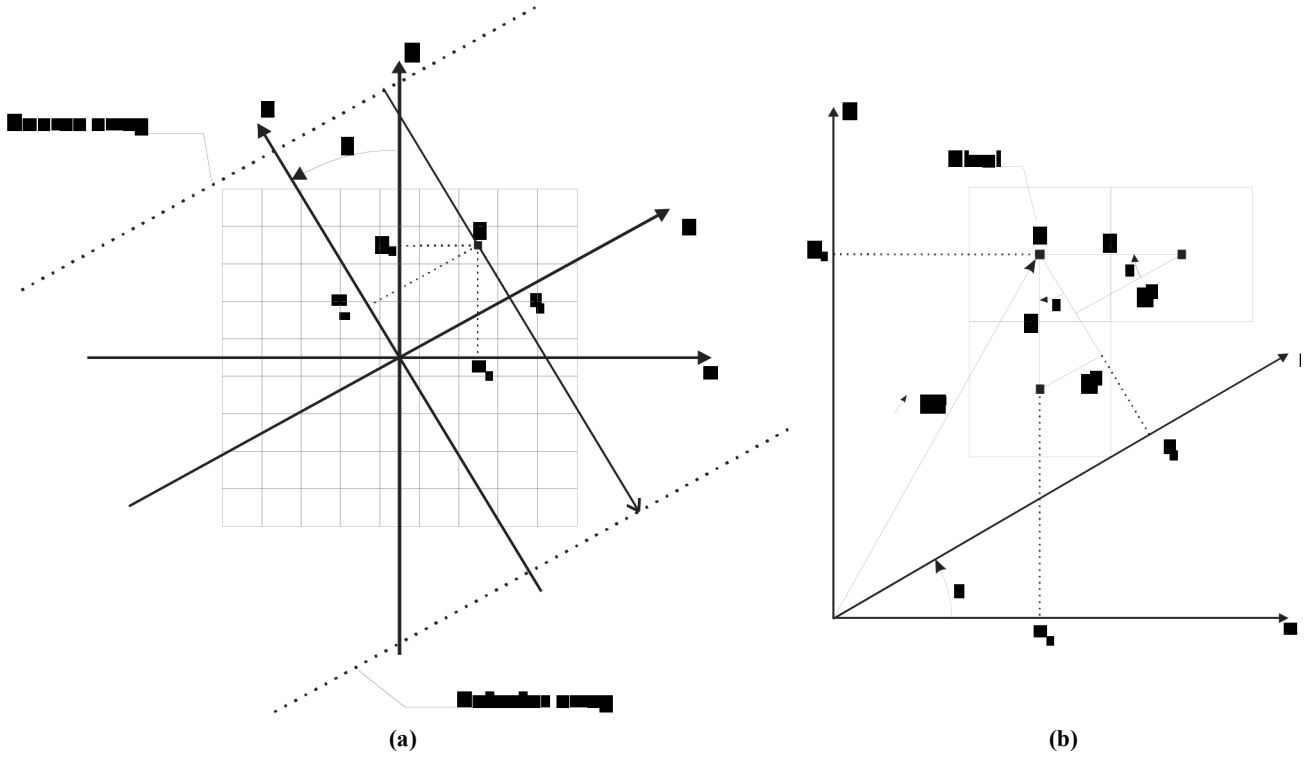
In this paper, we focus on both FPGA implementation performance and medical image quality. In previous work in the area of hardware implementations of tomographic processing algorithms, Wu[2] gives a brief overview of all major subsystems in a computed tomography (CT) scanner and proposes locations where ASICs and FPGAs can be utilized. According to the author, semi-custom digital ASICs were the most appropriate due to the level of sophistication that FPGA technology had in 1991. Agi et al.[3] present the first description of a hardware solution for computerized tomography of which we are aware. It is a unified architecture that implements forward Radon transform, parallel- and fan-beam backprojection in an ASIC based multi-processor system. Our FPGA implementation focuses on backprojection. Agi et al. [4] present a similar investigation of quantization effects; however their results do not demonstrate the suitability of their implementation for medical applications. Although their filtered sinogram data are quantized with 12-bit precision, extensive bit truncation on functional unit outputs and low accuracy of the interpolation factor (absolute error of up to 2) render this implementation significantly less accurate than ours, which is based on 9-bit projections and the maximal interpolation factor absolute error of  $2^{-4}$ . An alternative to using specially designed processors for the implementation of filtered backprojection (FBP) is presented in [5]. In this work, a fast and direct FBP algorithm is implemented using texture-mapping hardware. It can perform parallel-beam backprojection of a 512-by-512-pixel image from 804 projections in 2.1 seconds, while our implementation takes 0.25 seconds from 1024 projections. Luiz et. al.[6] investigated residue number systems (RNS) for the implementation of convolution based backprojection to speedup the processing. Unfortunately, extra binary-to-RNS and RNS-to-binary conversions are introduced. Other approaches to accelerating the backprojection algorithm have been investigated [7, 8]. One approach [7] presents an order  $O(n^2 \log n)$  and merits further study. The suitability to medical image quality and hardware implementation of these approaches[7,8] needs to be demonstrated. There are also a lot of interests in the area of fan-beam and cone-beam reconstruction using hardware implementation. An FPGA-based fan-beam

reconstruction module [9] is proposed and simulated using MAX+PLUS2, version 9.1, but no actual FPGA implementation is mentioned. Moreover, the authors did not explore the potential parallelism for different projections as we do, which is essential for speed-up. More data and computation is needed for 3D cone-beam FBP. Yu's PC based system [10] can reconstruct the  $512^3$  data from  $288 \times 512^2$  projections takes 15.03 minutes, which is not suitable for real-time. The embedded system described in [11] can do 3D reconstruction in 38.7 seconds with the fastest time reported in the literature. However, it is based on a Mercury RACE++ AdapDev 1120 development workstation and need many modifications for a different platform. Bins et. al.[12] have investigated precision vs. error in JPEG compression. The goals of this research are very similar to ours: to implement designs in fixed-point in order to maximize parallelism and area utilization. However, JPEG compression is an application that can tolerate a great deal more error than medical imaging.

In the next section, we present the backprojection algorithm in more detail. In section 3 we present our quantization studies and analysis of error introduced. Section 4 presents the hardware implementation in detail. Finally we present results and discuss future directions. An earlier version of this research was presented [16]. This paper provides a fuller discussion of the project and updated results.

## **2. PARALLEL-BEAM FILTERED BACKPROJECTION**

A parallel-beam CT scanning system uses an array of equally spaced unidirectional sources of focused X-ray beams. Generated radiation not absorbed by the object's internal structure reaches a collinear array of detectors (Figure 1a). Spatial variation of the absorbed energy in the two-dimensional plane through the object is expressed by the attenuation coefficient  $\mu(x, y)$ . The logarithm of the measured radiation intensity is proportional to the integral of the attenuation coefficient along the straight line traversed by the X-ray beam. A set of values given by all detectors in the array comprises a one-dimensional projection of the attenuation coefficient,  $P(t, \theta)$ , where  $t$  is the detector distance from the origin of the array, and  $\theta$  is the angle at which the measurement is taken. A collection of projections for different angles over  $180^\circ$  can be visualized in the form of an image in which one



**Figure 1: a) Illustration of the coordinate system used in parallel-beam backprojection, and b) geometric explanation of the incremental spatial address calculation**

axis is position  $t$  and the other is angle  $\theta$ . This is called a sinogram or Radon transform of the two-dimensional function  $\mu$ , and it contains information needed for the reconstruction of an image  $\mu(x, y)$ . The Radon transform can be formulated as

$$\log_e \frac{I_0}{I_d} = \iint \mu(x, y) \delta(x \cos \theta + y \sin \theta - t) dx dy \equiv P(t, \theta) \quad (1)$$

where  $I_0$  is the source intensity,  $I_d$  is the detected intensity, and  $\delta(\cdot)$  is the Dirac delta function. Equation (1) is actually a line integral along the path of the X-ray beam, which is perpendicular to the  $t$  axis (see Figure 1a) at location  $t = x \cos \theta + y \sin \theta$ . The Radon transform represents an operator that maps an image  $\mu(x, y)$  to a sinogram  $P(t, \theta)$ . Its inverse mapping, the inverse Radon transform, when applied to a sinogram results in an image. The filtered backprojection (FBP) algorithm performs this mapping [1].

FBP begins by high-pass filtering all projections before they are fed to hardware using the Ram-Lak or ramp filter, whose frequency response is  $|f|$ . The discrete formulation of backprojection is

$$\mu(x, y) = \frac{\pi}{K} \sum_{i=1}^K \Pi_{\theta_i}(x \cos \theta_i + y \sin \theta_i), \quad (2)$$

where  $\Pi_{\theta}(t)$  is a filtered projection at angle  $\theta$ , and  $K$  is the number of projections taken during CT scanning at angles  $\theta_i$  over a  $180^\circ$  range. The number of values in  $\Pi_{\theta}(t)$  depends on the image size. In the case of  $n \times n$  pixel images,  $N = \sqrt{2}nD$  detectors are required. The ratio  $D = d/\tau$ , where  $d$  is the distance between adjacent pixels and  $\tau$  is the detector spacing, is a critical factor for the quality of the reconstructed image and it obviously should satisfy  $D > 1$ . In our implementation, we utilize values of  $D \approx 1.4$  and  $N = 1024$ , which are typical for real systems. Higher values do not significantly increase the image quality.

Algorithmically, Eq. (2) is implemented as a triple nested “for” loop. The outermost loop is over projection angle,  $\theta$ . For each  $\theta$ , we update every pixel in the image in raster-scan order: starting in the upper left corner and looping first over columns,  $c$ , and next over rows,  $r$ . Thus, from (2), the pixel at location  $(r, c)$  is incremented by the value of  $\Pi_{\theta}(t)$  where  $t$  is a function of  $r$  and  $c$ . The issue here is that the X-ray going through the currently reconstructed pixel, in general, intersects the detector array between detectors. This is solved by linear interpolation. The point of intersection is calculated as an address corresponding to detectors numbered from 0 to 1023. The fractional part of this address is the interpolation factor. The equation that performs linear interpolation is given by

$$\Pi_{\theta}^{\text{int}}(i) = [\Pi_{\theta}(i+1) - \Pi_{\theta}(i)] \cdot IF + \Pi_{\theta}(i), \quad (3)$$

where  $IF$  denotes the interpolation factor,  $\Pi_{\theta}(t)$  is the 1024 element array containing filtered projection data at angle  $\theta$ , and  $i$  is the integer part of the calculated address. The interpolation can be performed beforehand in software, or it can be a part of the backprojection hardware itself. We implement interpolation in hardware because it substantially reduces the amount of data that must be transmitted to the reconfigurable hardware board.

The key to an efficient implementation of Equation (2) is shown in Figure 1b. It shows how a distance  $d$  between square areas that correspond to adjacent pixels can be converted to a distance  $\Delta t$  between locations where X-ray

beams that go through the centers of these areas hit the detector array. This is also derived from the equation  $t = x \cos \theta + y \sin \theta$ . Assuming that pixels are processed in raster-scan fashion, then  $\Delta t = d \cos \theta$  for two adjacent pixels in the same row ( $x_2 = x_1 + d$ ) and similarly  $\Delta t = d \sin \theta$  for two adjacent pixels in the same column ( $y_2 = y_1 - d$ ). Our implementation is based on pre-computing and storing these deltas in look-up tables (LUTs). Three LUTs are used corresponding to the nested “for” loop structure of the backprojection algorithm. LUT 1 stores the initial address along the detector axis (i.e. along  $t$ ) for a given  $\theta$  required to update the pixel at row 1, column 1. LUT 2 stores the increment in  $t$  required as we increment across a row. LUT 3 stores the increment for columns.

### 3. QUANTIZATION

Mapping the algorithm directly to hardware will not produce an efficient implementation. Several modifications must be made to obtain a good hardware realization. The most significant modification is using fixed-point arithmetic. For hardware implementation, narrow bit widths are preferred for more parallelism which translates to higher overall processing speed. However, medical imaging requires high precision which may require wider bit widths. We did extensive analysis to optimize this tradeoff. We quantize all data and all calculations to increase the speed and decrease the resources required for implementation. Determining allowable quantization is based on a software simulation of the tomographic process.



Figure 2: Major simulation steps

Figure 2 shows the major blocks of the simulation. An input image is first fed to the software implementation of the Radon transform, also known as reprojection [13], which generates the sinogram of 1024 projections and 1024 samples per projection. The filtering block convolves sinogram data with the impulse response of the ramp filter generating a filtered sinogram, which is then backprojected to give a reconstructed image.



All values in the backprojection algorithm are real numbers. These can be implemented as either floating-point or fixed-point values. Floating-point representation gives increased dynamic range, but is significantly more expensive to implement in reconfigurable hardware, both in terms of area and speed. For these reasons we have chosen to use fixed-point arithmetic. An important issue, especially in medical imaging, is how much numerical accuracy is sacrificed when fixed-point values are used. Here, we present the methods used to find appropriate bit-widths for maintaining sufficient numerical accuracy. In addition, we investigate possibilities for bit reduction on the outputs of certain functional units in the datapath for different rounding schemes, and what influence that has on the error introduced in reconstructed images. Our analysis shows that medical images display distinctive properties with respect to how different quantization choices affect their reconstruction. We exploit this and customize quantization to best fit medical images. We compute the quantization error by comparing a fixed-point image reconstruction with a floating-point one.

Fixed-point variables in our design use a general slope/bias-encoding, meaning that they are represented as

$$V \approx V_a = S Q + B, \quad (4)$$

where  $V$  is an arbitrary real number,  $V_a$  is its fixed-point approximation,  $Q$  is an integer that encodes  $V$ ,  $S$  is the slope, and  $B$  is the bias. Fixed-point versions of the sinogram and the filtered sinogram use slope/bias scaling where the slope and bias are calculated to give maximal precision. The quantization of these two variables is calculated as:

$$S = \frac{\max(V) - \min(V)}{\max(Q) - \min(Q)} = \frac{\max(V) - \min(V)}{2^{ws} - 1}, \quad (5)$$

$$B = \max(V) - S \cdot \max(Q) \quad \text{or} \quad B = \min(V) - S \cdot \min(Q), \quad (6)$$

$$Q = \text{round} \left( \frac{V - B}{S} \right), \quad (7)$$

where  $ws$  is the word size in bits of integer  $Q$ . Here,  $\max(V)$  and  $\min(V)$  are the maximum and minimum values that  $V$  will take, respectively.  $\max(V)$  was determined based on analysis of data. Since sinogram data are unsigned numbers, in this case  $\min(V) = \min(Q) = B = 0$ . The interpolation factor is an unsigned fractional

number and uses radix point-only scaling. Thus, the quantized interpolation factor is calculated as in Eq. (7), with saturation on overflow, with  $S = 2^{-E}$  where  $E$  is the number of fractional bits, and with  $B = 0$ .

For a given sinogram,  $S$  and  $B$  are constants and they do not show up in the hardware – only the quantized value  $Q$  is part of the hardware implementation. Note that in Eq. (3), two data samples are subtracted from each other before multiplication with the interpolation factor takes place. Thus, in general, the bias  $B$  is eliminated from the multiplication, which makes quantization of filtered sinogram data with maximal precision scaling easily implementable in hardware.

The next important issue is the metric used for evaluating of the error introduced by quantization. Our goal was to find a metric that would accurately describe visual differences between compared images regardless of their dynamic range. If 8-bit and 16-bit versions of a single image are reconstructed so that there is no visible difference between the original and reconstructed images, the proper metric should give a comparable estimate of the error for both bit-widths. The proper metric should also be insensitive to the shift of pixel value range that can emerge for different quantization and rounding schemes. Absolute values of single pixels do not effect visual image quality as long as their relative value is preserved, because pixel values are mapped to a set of grayscale values. The error metric we use that meets these criteria is the Relative Error (RE):

$$RE = \frac{\sum_{i=1}^M [(x_i - \bar{x}) - (y_i^{FP} - \bar{y}^{FP})]^2}{\sum_{i=1}^M (y_i^{FP} - \bar{y}^{FP})^2}, \quad (8)$$

Here,  $M$  is the total number of pixels,  $x_i$  and  $y_i^{FP}$  are the values of the  $i$ -th pixel in the quantized and floating-point reconstructions respectively, and  $\bar{x}$ ,  $\bar{y}^{FP}$  are their means. The mean value is subtracted because we only care about the relative pixel values.

Figure 3 shows some characteristic images from a larger set of 512-by-512-pixel images used as inputs to the simulation process. All images are monochrome 8-bit images, but 16-bit versions are also used in simulations.

Each image was chosen for a certain reason. For example, the Shepp-Logan phantom is well known and widely used in testing the ability of algorithms to accurately reconstruct cross sections of the human head. It is believed that cross-sectional images of the human head are the most sensitive to numerical inaccuracies and the presence of artifacts induced by a reconstruction algorithm [1]. Other medical images were Female, Head, and Heart obtained from the visible human web site[14]. The Random image (a white noise image) should result in the upper bound on bit-widths required for a precise reconstruction. The Artificial image is unique because it contains all values in the 8-bit grayscale range. This image also contains straight edges of rectangles, which induce more artifacts in the reconstructed image. The Artificial image also contains a regular border around the head slice.

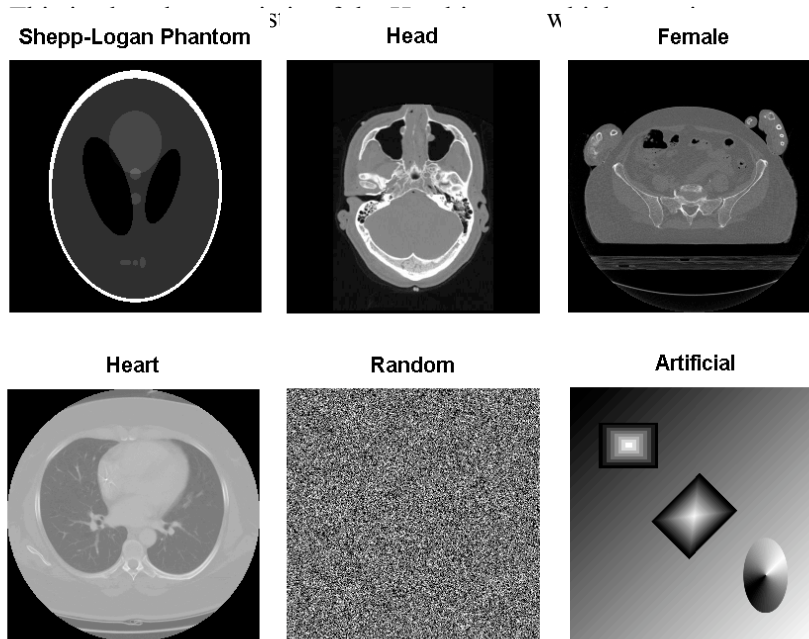


Figure 3: Some of the images used as inputs to the simulation process

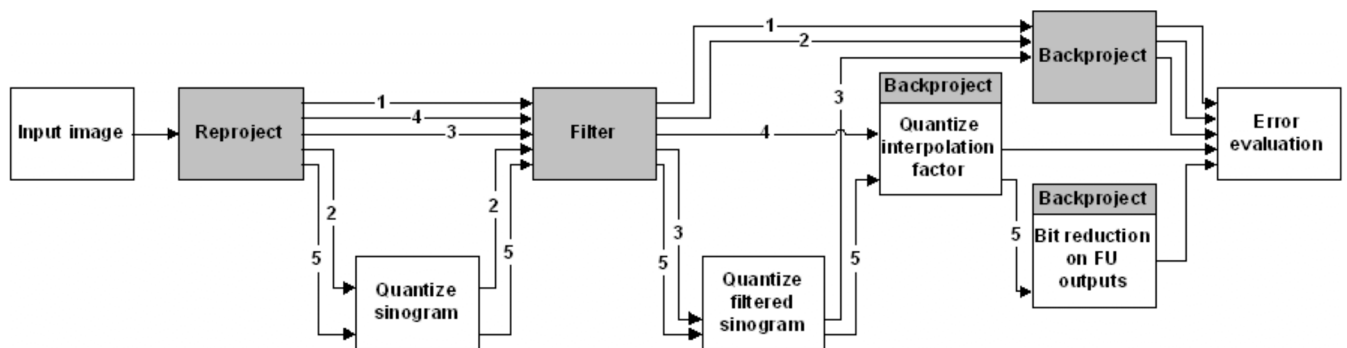


Figure 4: Detailed flowchart of the simulation process

Figure 4 shows the detailed flowchart of the simulated CT process. In addition to the major blocks designated as Reproject, Filter and Backproject, Figure 4 also includes the different quantization steps that we have investigated. Each path in this flowchart represents a separate simulation cycle. Cycle 1 gives a floating-point (FP) reconstruction of an input image. All other cycles perform one or more type of quantization and their resulting images are compared to the corresponding FP reconstruction by computing the Relative Error. The first quantization step converts FP projection data obtained by the reproject step to a fixed-point representation. Simulation cycle 2 is used to determine how different bit-widths for quantized sinogram data affect the quality of a reconstructed image. Our research was based on a prototype system that used 12-bit accurate detectors for the acquisition of sinogram data. Simulations showed that this bit-width is a good choice since worst case introduced error amounts to 0.001%. The second quantization step performs the conversion of filtered sinogram data from FP to fixed-point representation. Simulation cycle 3 is used to find the appropriate bit-width of the words representing a filtered sinogram. Figure 5 shows the results for this cycle. Since we use linear interpolation of projection values corresponding to adjacent detectors, the interpolation factor in Equation (3) also has to be quantized. Figure 6 summarizes results obtained from simulation cycle 4, which is used to evaluate the error induced by this quantization.

Figures 5 and 6 show the Relative Error metric for different word length values and for different simulation cycles for a number of input images. Some input images were used in both 8-bit and 16-bit versions. Figure 5 corresponds to the quantization of filtered sinogram data (path 3 in Figure 4). The conclusion here is that 9-bit quantization is the best choice since it gives considerably smaller error than 8-bit quantization, which for some images induces visible artifacts. At the same time, 10-bit quantization does not give visible improvement. The exceptions are images 2 and 3, which require 13 bits. From Figure 6 (path 4 in Figure 4), we conclude that 3 bits for the interpolation factor (meaning the maximum error for the spatial address is  $2^{-4}$ ) is sufficiently accurate. As expected, image 1 is more sensitive to the precision of the linear interpolation because of its randomness. Figure 7 shows that combining these quantization schemes results in a very small error for image “Head” in Figure 3.

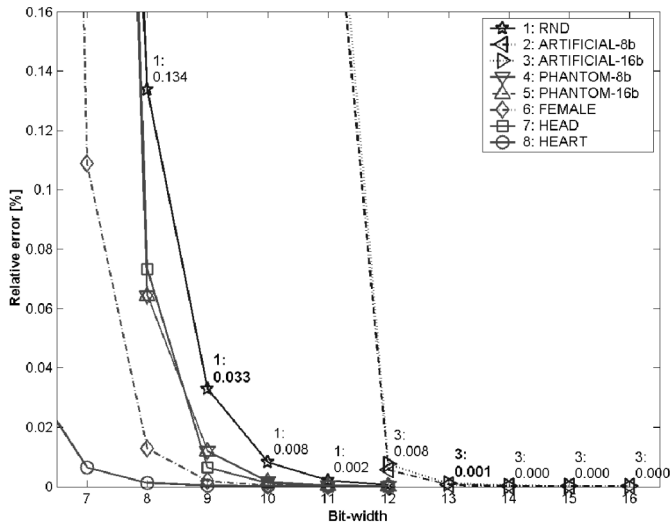


Figure 5: Simulation results for the quantization of filtered sinogram data

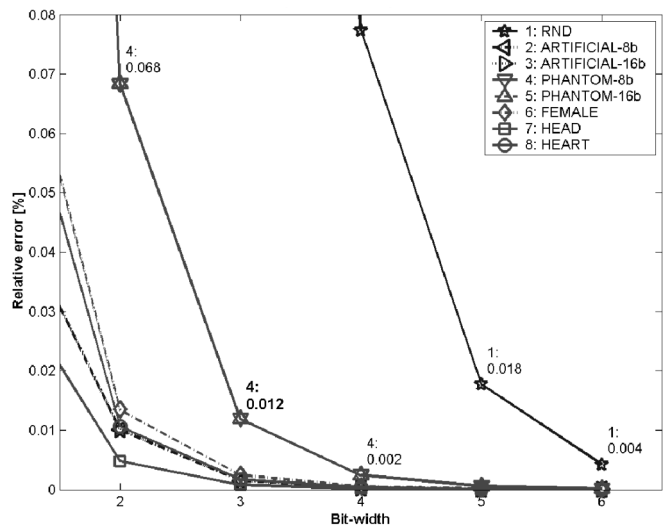


Figure 6: Simulation results for the quantization of the interpolation factor

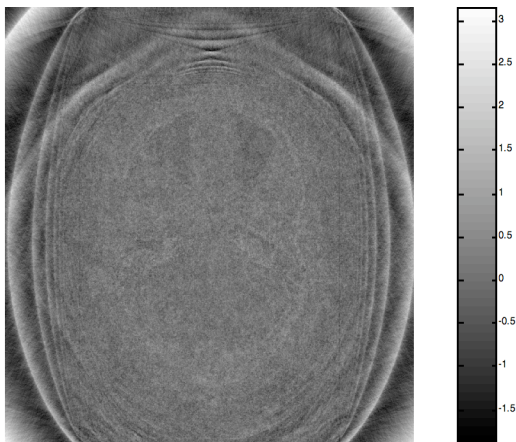


Figure 7: Relative error between fixed-point and floating-point reconstruction

**Before filtering: 12 bits**  
**After filtering: 9 bits**  
**Interpolation factor: 3 bits**  
  
**Relative error: 0.00502%**

We also investigated whether it is feasible to discard some of the least significant bits (LSBs) on outputs of functional units (FUs) in the datapath and still not introduce any visible artifacts. The goal is for the reconstructed pixel values to have the smallest possible bit-widths. This is based on the intuition that bit reduction done further down the datapath will introduce a smaller amount of error in the result. If the same bit-width were obtained by simply quantizing filtered projection data with fewer bits, the error would be magnified by the operations performed in the datapath, especially by the multiplication. Path number 5 in Figure 4 depicts the simulation cycles that investigates bit reduction at the outputs of three of the FUs. These FUs implement subtraction,

multiplication and addition that are all part of the linear interpolation from Equation (3). When some LSBs are discarded, the remaining part of a binary word can be rounded in different ways. We investigate two different rounding schemes, specifically rounding to nearest and truncation (or rounding to floor). Rounding to nearest is expected to introduce the smallest error, but requires additional logic resources. Truncation has no resource requirements, but introduces a negative shift of values representing reconstructed pixels. Bit reduction effectively optimizes bit-widths of FUs that are downstream in the data flow.

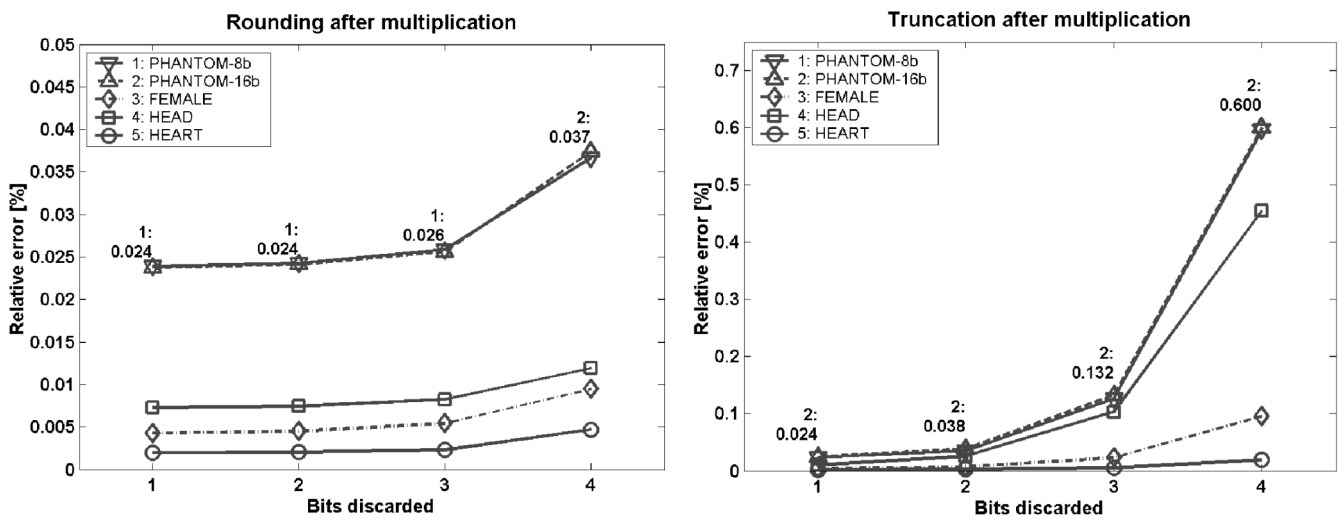
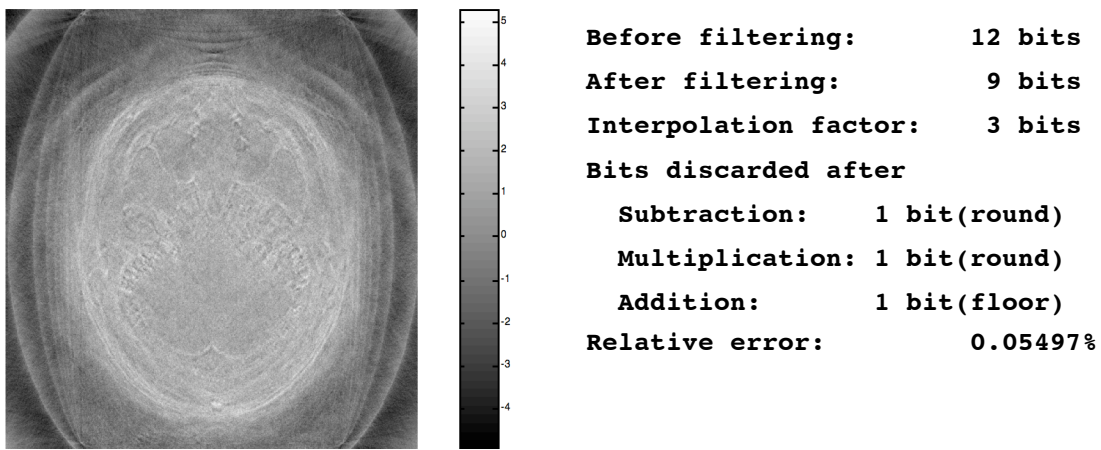


Figure 8: Bit reduction on the output of the interpolation multiplier

Figure 8 shows tradeoffs of bit reduction and the two rounding schemes after multiplication for medical images. It should be noted that sinogram data are quantized to 12 bits, filtered sinogram to 9 bits, and the interpolation factor is quantized to 3 bits ( $2^{-4}$  precision). Similar studies were done for the subtraction and addition operations and on a broader set of images. It was determined that medical images suffer the least amount of error introduced by combining quantizations and bit reduction. For medical images, in case of rounding to nearest, there is very little difference in the introduced error between 1 and 3 discarded bits after multiplication and addition. This difference is higher in the case of bit reduction after addition because the multiplication that follows magnifies the error. For all three FUs, when only medical images are considered, there is a fixed relationship between rounding to nearest and truncation. Two least-significant bits discarded with rounding to nearest introduce an error that is lower than

or close to the error of 1 bit discarded with truncation. Although rounding to nearest requires logic resources, even when only one LSB is discarded with rounding to nearest after each of three FUs, the overall resource consumption is reduced because of savings provided by smaller FUs and pipeline registers (see Figure 11 and 12). Figure 9 shows that discarding LSBs introduces additional error on medical images for this combination of quantizations. In our case there was no need for using bit reduction to achieve smaller resource consumption because the targeted FPGA chip (Xilinx Virtex1000) provided sufficient logic resources.



**Figure 9: Relative error between fixed-point and floating-point reconstruction (With LSB discarding)**

There is one more quantization issue we considered. It pertains to data needed for the generation of the address into a projection array (spatial address *addr*) and to the interpolation factor. As described in the introduction, there are three different sets of data stored in look-up tables (LUTs) that can be quantized. Since pixels are being processed in raster-scan order, the spatial address *addr* is generated by accumulating entries from LUTs 2 and 3 to the corresponding entry in LUT 1. The 10-bit integer part of the address *addr* is the index into the projection array  $\Pi_{\theta}(\cdot)$ , while its fractional part is the interpolation factor. By using radix point-only scaling for the quantization of data in LUTs 1, 2 and 3, the interpolation factor is the lower part of the word that represents the generated address. Thus, it can conveniently be extracted and fed to the multiplier that implements linear interpolation. In software simulations we are using a 3-bit interpolation factor quantized by rounding to nearest. This results in a

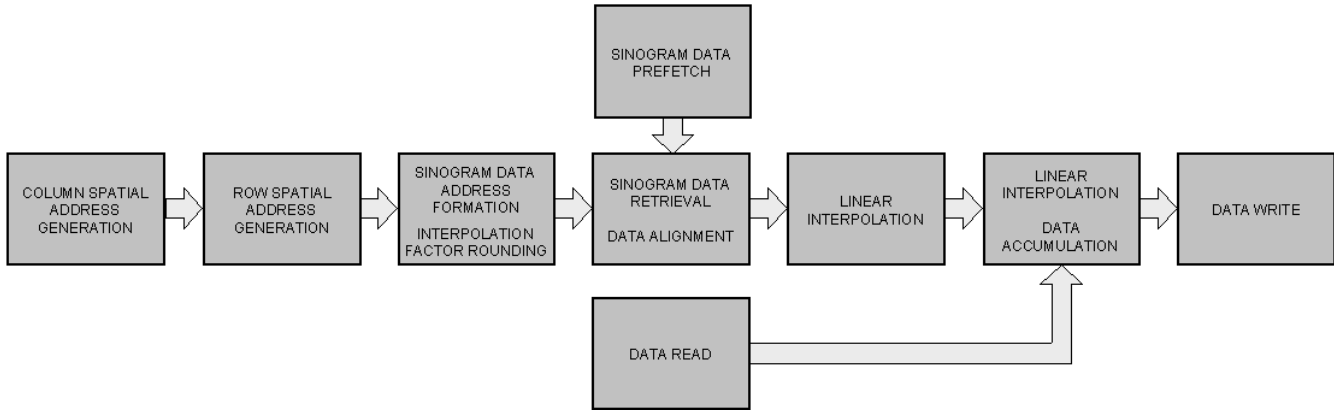
maximum error of  $2^{-4}$ , since the interpolation factor is fractional. The goal is to keep the calculation of the spatial address *addr* in hardware to the same precision. The quantization error of the data from the LUTs is accumulated as pixels are traversed. The critical path for error accumulation is 512 pixels in the vertical and 512 pixels in the horizontal direction, which corresponds to the longest path from one image corner to the opposite one. This results in entries in the first LUT requiring 10 integer and 5 fractional bits, entries in the second LUT having 1 integer bit and 15 fractional bits, and the third LUT having 2 integer and 15 fractional bits. Values stored in LUT 2 are all positive; their representation does not need a sign bit. The MSB of an entry from the third LUT is a sign bit. The address generated will have 15 fractional bits, which can be rounded to 4 bits (the interpolation factor) by rounding to nearest. The total accumulated error in the worst case is  $2^{-6} + 512 \cdot 2^{-16} + 512 \cdot 2^{-16} + 2^{-5}$  which is equal to  $2^{-4}$ . Another option is to discard the 10 least significant fractional bits from the address to get 5 bits. This introduces the same maximal error of  $2^{-5}$  as rounding to 4 bits, but the multiplier and all other FUs in the datapath have to be wider and consume more logic resources. It is thus more area effective to implement rounding to nearest. It is important to note that the worst case quantization error  $2^{-4}$  for the interpolation factor used in simulations was the same for all pixels, while in hardware it linearly increases from the starting pixel and amounts to  $2^{-4}$  for the last pixel, so the hardware generates a more accurate spatial address.

#### **4. HARDWARE ORGANIZATION**

Hardware acceleration in reconfigurable hardware comes from parallel processing. There are two basic sources of parallelism in the backprojection algorithm. Pixel parallelism means that the image can be divided into subsections (for example quadrants), which can be reconstructed simultaneously. Projection parallelism calculates the summation from Equation (2) by simultaneously processing individual projections. If the number of pixels reconstructed in parallel is  $n$ , then the memory bandwidth required for the accumulation of reconstructions from different projections is  $n$  times the bandwidth required for one pixel. Thus, utilization of pixel parallelism is limited by the available memory bandwidth. On the other hand, projection parallelism does not require higher memory bandwidth than what is needed for the non-parallel implementation. In addition, as larger FPGAs become



available, more parallelism can easily be extracted. Therefore, we decided to base our architecture primarily on projection parallelism.



**Figure 10: Data flow of the backprojection hardware**

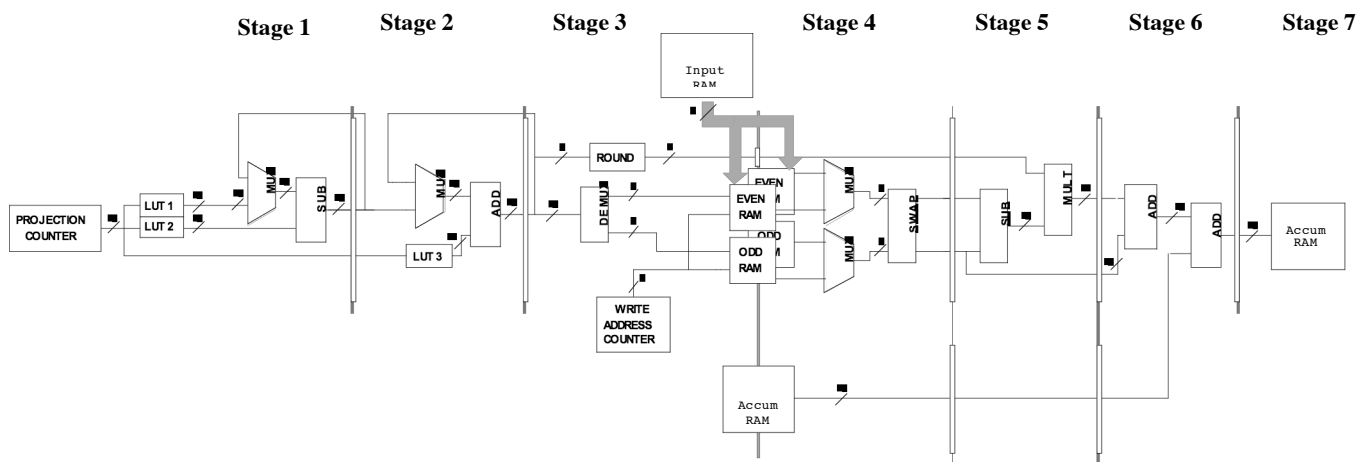
Figure 10 shows a functional view of our hardware implementation. There are two simultaneous flows of data that can be identified from this diagram: the pixel reconstruction flow, and the sinogram data feeding flow.

The reconstruction flow, which is horizontal in Fig. 10, consists of seven pipeline stages and an accumulation stage. The spatial address generation (SAG) is separated into two stages: the first produces a spatial address for each pixel located outside of the image and next to the leftmost image column. This is done incrementally starting from the top pixel. For the spatial address of the first pixel in each row to be obtained, the result of the first SAG stage is passed to the second, which accumulates the necessary horizontal (row) component of the address. Spatial addresses for other pixels in the same row are then generated incrementally in successive clock cycles. In the next stage, the integer part of a spatial address is used to form memory addresses of two adjacent sinogram values needed for linear interpolation, and the fractional part is rounded to give the interpolation factor. The fourth stage retrieves sinogram data from the on-chip memory banks and aligns them to proper inputs of the subtractor in the next stage, which starts performing linear interpolation. Because the pipeline stages are balanced, the resources that implement linear interpolation have been distributed into two stages. Stage six completes the interpolation and accumulates the result to the value for the same pixel reconstructed from previous projections. The

accumulation part has to be synchronized with the data read part so that in each clock cycle in stage six, they correspond to the same pixel. The last stage stores this updated reconstruction into an off-chip memory bank.

The sinogram data feeding flow originates in one of the off-chip memory banks, where projections are stored, and prefetches them into on-chip memory banks. This flow of data always prefetches the next projection to be processed and does that simultaneously with processing the current projection in the pipeline flow.

Figure 11 shows the datapath of the non-parallel backprojection hardware. In this figure, the long narrow rectangles represent pipeline registers; they separate pipeline stages. Here, one pixel is reconstructed from one projection every clock cycle. After reconstructing all pixels from one projection, the next projection has already been loaded and the reconstruction process continues. The first stage of the pipeline stores a new value in the pipeline register once every 512 cycles, i.e. when processing of the next image row commences. During the next cycle, the second stage passes the result of the first SAG stage through its multiplexer to be accumulated with a value from LUT 3 pointed to by the projection number.



**Figure 11: Datapath implementation of the non-parallel backprojection hardware**

A double buffering scheme is used for storing projections; this means processing and data fetch are overlapped in time. In Figure 11, there are two sets of Even RAM and Odd RAM. One set of Even and Odd RAM is called

foreground RAM, and is used for the current projection processing, while the other, background RAM, is loading the next projection data. With every new projection, the buffers switch their roles. Each buffer is comprised of two blocks of on-chip RAM, one storing odd addressed projection data, and the other storing even addressed data. By this means, we can easily implement linear interpolation with one clock cycle data access.

When the reconstruction process is initiated, the first 512 cycles are used to load the first projection that corresponds to the first rotational angle of the source-detector system. Each cycle loads two 9-bit sinograms to Even and Odd RAM simultaneously. After that, processing and fetching are overlapped. The filtered and quantized sinogram is stored in one off-chip memory bank. Each memory word contains two consecutive projection samples.

For the purpose of performing linear interpolation, two projection values with indices  $\lfloor addr \rfloor$  and  $\lceil addr \rceil$ , where  $addr$  is the spatial address, have to be available from the processed projection in a single clock cycle. Having separate odd and even on-chip RAM blocks, which can be read at the same time, makes this possible. If  $\lfloor addr \rfloor$  is an odd number, the projection value that corresponds to this number is located in the odd RAM block at address  $\lfloor addr \rfloor - 1$ , and the next adjacent projection value is in the even block at address  $\lfloor addr \rfloor$ . If  $\lfloor addr \rfloor$  is an even number, the corresponding projection data are at address  $\lfloor addr \rfloor - 1$  in both even and odd memory blocks, except for  $\lfloor addr \rfloor = 0$  when both addresses are also 0. The formation of these addresses and their distribution to memory blocks is implemented in the functional unit designated *demux*. Out of two adjacent projection values that are linearly interpolated, the higher address ( $\lceil addr \rceil$ ) should always be fed to the *in1* input of the subtracter, which performs the operation  $in1 - in2$ . This projection value can come from either the even or the odd memory block. The unit *swap* takes care of this in stage 4.

Stage 5 and the first adder from stage 6 implement linear interpolation. The second adder in stage 6 accumulates the value of the same pixel reconstructed from previously processed projections with the interpolation result. The result is then stored for future accumulation.

Two off-chip memory banks (Accum RAMs in Figure 11) store intermediate results of the accumulation for each pixel. Both are used interchangeably as sources and destinations. Every time a new projection is processed, the accumulation RAM that was a destination for the last addition in stage 6 switches its role to a source, and the one that was a source becomes a destination. Thus, a pixel's current reconstruction can be read and a new one stored at the same address in a single clock cycle.

Since accumulation RAM responds to the initial read request with a delay of several clock cycles, every time processing of a new projection starts, the pipeline has to be stalled until the data and the pixel reconstruction stage are synchronized. The two pipeline registers at the bottom of Figure 11 enable detection of valid data coming from the source accumulation RAM and give enough time for the control unit to stall or reactivate the pipeline.

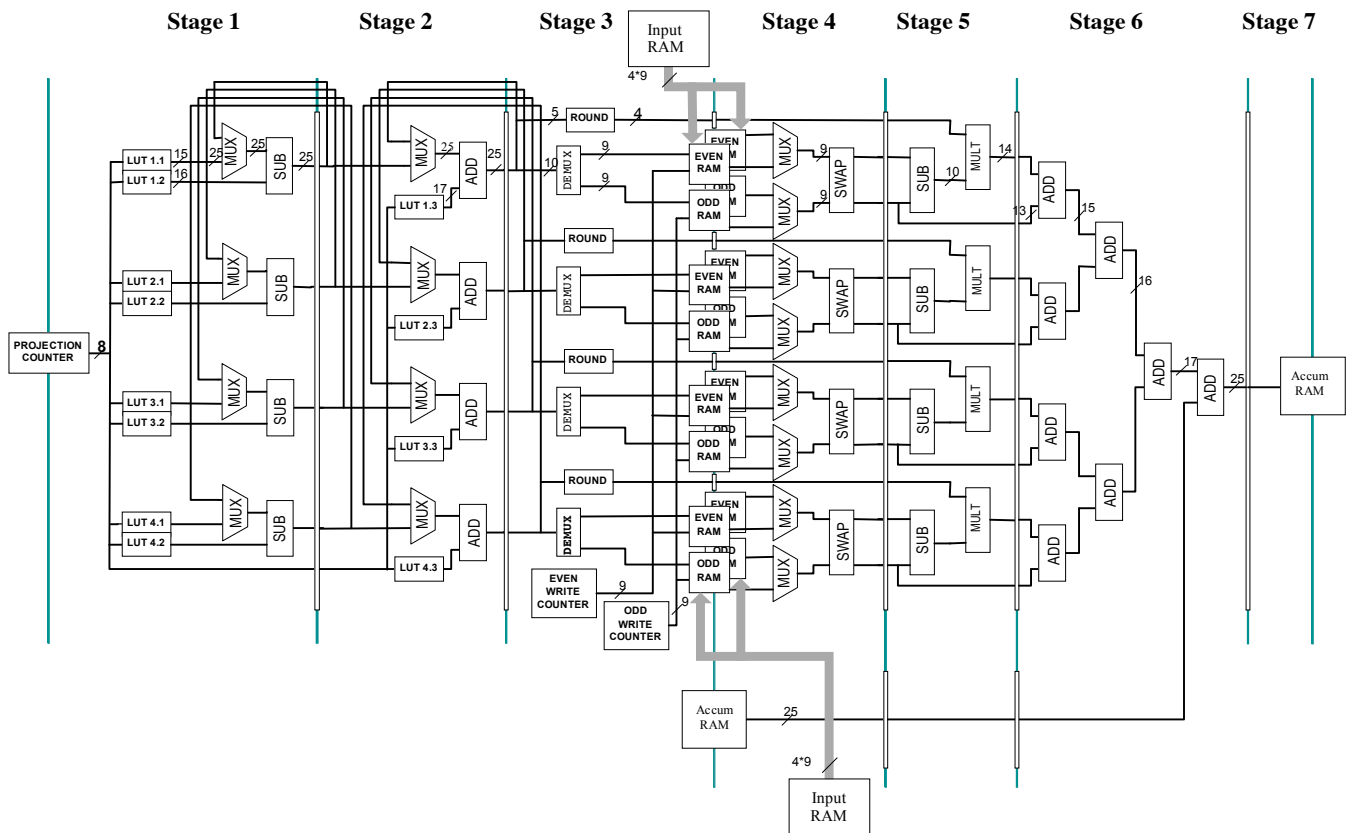


Figure 12: Datapath of the implemented 4-way parallel backprojection hardware

The architecture described can be extended to n-way parallel processing. The main difference is that n projections are processed simultaneously in the parallel structure. Figure 12 gives the datapath for a 4-way parallel architecture, which is used as an example to explain the basic idea. Parallel processing requires replication of most pipeline resources. In this case, the pipeline flow is quadrupled. The LUTs are reorganized into 4 smaller units that together store the same data. The addition tree in stage 6 sums the reconstructions from each of 4 projections and the last adder in this stage accumulates this value with previous results for the same pixel. Since 4 projections are being loaded in parallel for 512 cycles, two input RAM banks are required to store sinogram data to achieve the required bandwidth. From each bank, 4 projection samples are loaded separately into even and odd on-chip buffer banks. This makes the bandwidth of the sinogram data flow 4 times higher than in the non-parallel case. However, if instead of loading all 4 projections in 512 cycles, this task is extended to last 2048 cycles, the required bandwidth would not need to be increased. The processing of one set of projections takes  $512^2$  cycles, which gives as many cycles for the prefetch of the next set to be completed. Only the loading of the first set of projections, which is not overlapped with processing, would affect the total execution time.

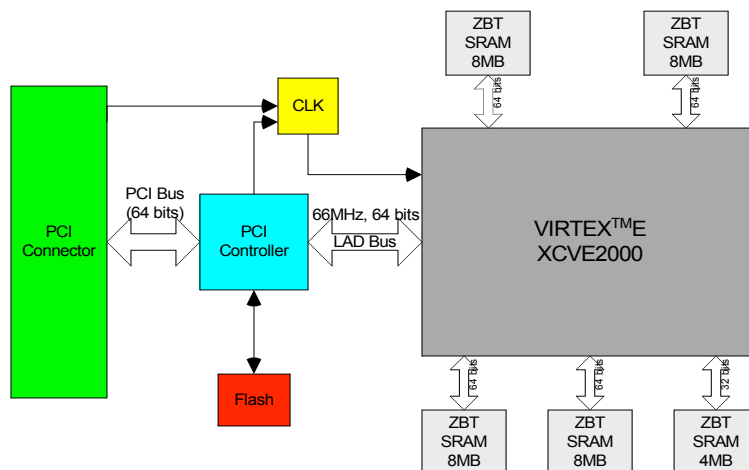


Figure 13: Firebird Diagram

More projection parallelism can be extracted by processing more than four projections at the same time as long as we have enough hardware resource such as on-chip RAM, logic resources, on-board memory etc. We have

implemented parallel-beam backprojection on an Annapolis Micro Systems Firebird board using a Xilinx Virtex 2000E FPGA chip. The block diagram of Firebird board is showed in Figure 13. It has five independent on-board memory banks, containing 36 Mbytes of synchronous ZBT SRAM. Two SRAM banks are used for sinogram data and two more are used for reconstructed image data. The fifth SRAM, which has smaller bandwidth to the FGPA than the others, is not used in our design.

Non-parallel processing, 8-way and 16-way parallel processing have been implemented using this board. Compared to 4-way, two modifications are needed for 16-way processing. First, further replication of functional units and two more level of accumulation is needed. The addition tree does not fit in stage 6, so we put the last two level of accumulation adders into stage 7 which is almost empty (see Figure 12). Second, more clock cycles are needed to prefetch the sinogram data. Due to the memory bandwidth limitation, we can not load all 16 projections in 512 clock cycles, instead, we use 2048 clock cycles. As mentioned before, the processing of one set of projections takes  $512 \times 512$  cycles, thus we have enough time to load the projections for the next round of processing. The main limiting factor for extracting further parallelism is the amount of on-chip RAM. Our implementation uses 90% of the RAM available on the targeted FPGA chip (Xilinx Virtex2000E), and 29% of logic resources. By migrating to newer FPGAs, such as the Xilinx VC2V6000 with larger on-chip RAM capacity, more than 16-way parallel implementations can be accommodated.

## 5. RESULTS AND PERFORMANCE

We have implemented parallel-beam backprojection on an Annapolis Micro Systems Firebird board using one Xilinx Virtex2000E FPGA chip.

A. Software – 1 GHz Pentium (Fixed Point):	28s
B. Hardware (non-parallel) – 75MHz:	3.6s
C. Hardware (8-way parallel) – 65MHz:	0.5s
D. Hardware (16-way parallel) – 65MHz:	0.25s

1024\*1024 projection values; 512\*512 pixel image

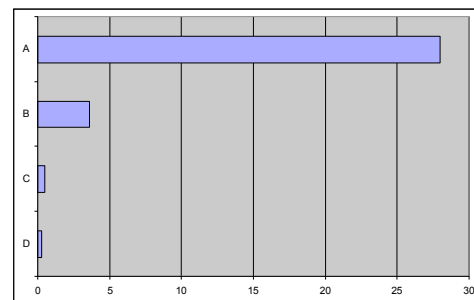
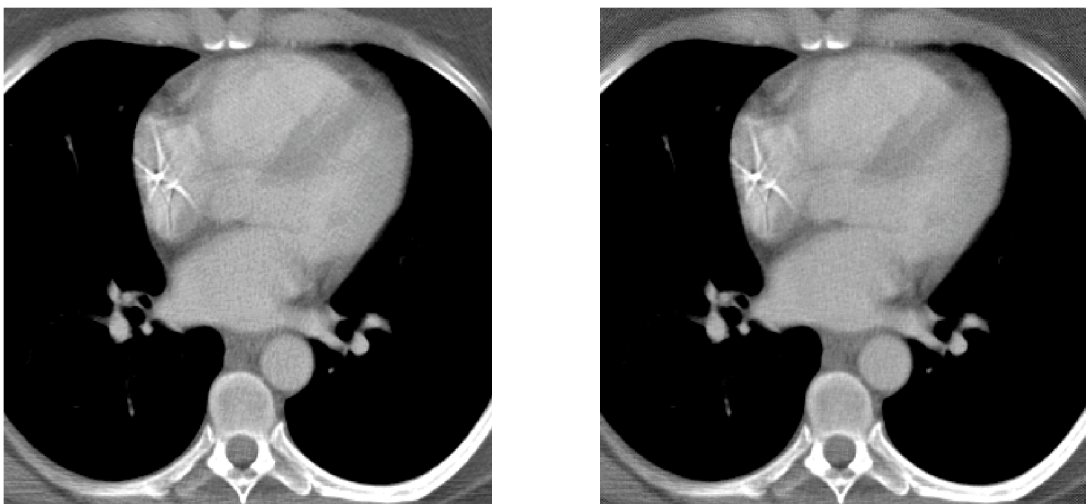


Figure 14: Performance results – Software vs. Hardware

Figure 14 summarizes our performance results by comparing backprojection execution times in seconds for software and hardware implementations. The software is run on a 1GHz Pentium PC with 256KB of cache. The software implementation performs all calculations on integer values obtained after the quantizations have been performed (to be similar to the hardware implementation). The computation numbers are the time to compute the reconstruction of an image from sinogram data. In hardware, this includes the time to compute and store the reconstructed image on the FPGA hardware. The time to transmit sinogram data and image data between the host PC and the FPGA board are not included. The hardware implementation labeled as test case D is the 16-way parallel version (extended structure of Figure 12), while B denotes the non-parallel version shown in Figure 11. Note that, despite its relatively slow clock speed, the FPGA implementation significantly outperforms the software implementation. Application clock speeds of 60 MHz or more are fast for FPGA implementations. The performance improvement over the software implementation is due mainly to parallelism in the implementation as well as other factors [15].

Figure 15 shows a section from the test image Heart on the left-hand side and the same section from its hardware reconstruction on the right hand side. The mapping between the grayscale range and the pixel value range was manually adjusted to show the image in more detail. This illustrates the high quality medical image reconstructed by our approach.



**Figure 15: Performance results – Software vs. Hardware**

## 6. CONCLUSION

We have presented an FPGA implementation of the parallel-beam backprojection algorithm optimized for medical imaging. We have based our implementation on the analysis of quantization effects caused by finite bit-widths, and paid special attention not to compromise the high precision requirements of medical imaging. Our quantization analysis results were used by Mercury Computer Systems, Inc. for their cone-beam reconstruction [11]. Our solution shows a 100 times speed-up over a similar software implementation. The combined effect of our quantizations results in a worst case relative error of 0.015% compared to a floating-point implementation. Our approach was developed with future expansions in mind. Real-time image reconstruction is easily attainable by exploiting the inherent parallelism of our solution to utilize the resources of larger FPGA devices. The hardware architecture presented can easily be modified to different bit-widths in order to accommodate different sensors and applications. In the future we plan to investigate the application of this approach to fan-beam reconstruction. We also plan to investigate the applicability of our hardware implementation to SAR image formation and to cone beam reconstruction.

## ACKNOWLEDGEMENTS

We would like to thank Mercury Computer Systems who contributed to the funding of this research. This work was affiliated with CenSSIS, the Center for Subsurface Sensing and Imaging Systems, under the Engineering Research Centers Program of the National Science Foundation (award number EEC-9986821).

## REFERENCES

- [1] Kak, A.C., and Slaney, M., *Principles of Computerized Tomographic Imaging*, New York, IEEE Press, 1988.
- [2] Wu, M.A., "ASIC Applications in Computed Tomography Systems," *Proceedings of Fourth Annual IEEE International ASIC Conference and Exhibit*, Rochester, NY, USA 1991, pp.P1-3/1-4.



- [3] Agi, I., Hurst, P.J., and Current, K.W., "An image processing IC for backprojection and spatial histogramming in a pipelined array," *IEEE Journal of Solid-state Circuits*, vol. 28, no. 3, 1993, pp. 210-221.
- [4] Agi, I., Hurst, P.J., and Current, K.W., "A VLSI architecture for high-speed image reconstruction: considerations for a fixed-point architecture," *Proceedings of SPIE, Parallel Architectures for Image Processing*, vol. 1246, 1990, pp. 11-24.
- [5] Stephen G. Azevedo, Brian K. Cabral and J. Foran, "Tomographic image reconstruction and rendering with texture-mapping hardware," *Proceedings of Mathematical Methods in Medical Imaging III, SPIE*, vol. 2299, 1994, pp. 280-289
- [6] Luiz Maltar C.B., Felipe M.G. Franca, Vladimir C. Alves, Claudio L. Amorim, "Reconfigurable Hardware for Tomographic Processing," *Proceedings of the XI Brazilian Symposium on Integrated Circuit Design, IEEE Computer Society Press, Rio de Janeiro/RJ*, 1998, pp. 19-24.
- [7] Basu, S., and Bresler, Y., " $O(N^2 \log_2 N)$  filtered backprojection reconstruction algorithm for tomography," *IEEE Transactions on Image Processing*, vol. 9, no. 10, Oct 2000, pp. 1760-1773.
- [8] Chen, Chung-Ming, Cho, Zang-Hee, and Wang, Cheng-Yi, "A Fast Implementation of the Incremental Backprojection Algorithms for Parallel Beam Geometries," *IEEE Transactions on Nuclear Science*, vol. 43, no. 6, Dec 1996, pp. 3328-3334.
- [9] Luiz Maltar C.B., Felipe M.G. Franca, Vladimir C. Alves, Claudio L. Amorim, "An FPGA-Based Fan Beam Image Reconstruction Module", *Proceedings of the Seventh Annual IEEE Symposium on Field-Programmable Custom Computing Machines*, Napa, CA, USA, April 1999, pp. 331-332.
- [10] R. Yu, R. Ning, B. Chen, "High Speed Cone Beam Reconstruction on PC", *SPIE Medical Imaging 2001*, San Diego, CA, Feb. 17-22, 2001, pp. 964-973.
- [11] Iain Goddard, Marc Trepanier, "High-Speed Cone-Beam Reconstruction: an Embedded Systems Approach", *SPIE Medical Imaging 2002*, San Diego, CA, Feb 24-26, 2002, pp.483-491.
- [12] Bins, J., Draper, B., Bohm, W., and Najjar, W., "Precision vs. Error in JPEG Compression," *Parallel and Distributed Methods for Image Processing III (SPIE)*, Denver CO, Jul 22, 1999, pp. 76-87.
- [13] Joseph, P.M. "An improved algorithm for reprojecting rays through pixel images," *IEEE Transactions on Medical Imaging*, vol. MI-1, no. 3, Nov 1982, pp.192-196.
- [14] [http://www.nlm.nih.gov/research/visible/fresh\\_ct.html](http://www.nlm.nih.gov/research/visible/fresh_ct.html), last accessed Nov 14, 2002
- [15] Guo, Z. Najjar, W., Vahid, F., Vissers, K. "A Quantitative Analysis of the Speedup Factors of FPGAs over processors," *Twelfth ACM International Symposium on Field-Programmable Gate Arrays (FPGA04)*, February, 2004, pp. 162-170.
- [16] Srdjan Coric, Miriam Leeser, Eric Miller, and Marc Trepanier, "Parallel-Beam Backprojection: an FPGA Implementation Optimized for Medical Imaging" *Tenth ACM International Symposium on Field-Programmable Gate Arrays (FPGA02)*, February, 2002, pp. 217-226.