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

Srdjan Coric, Miriam Leeser, Eric Miller
Department of Electrical and Computer Engineering
Northeastern University
Boston, MA 02115
{scoric, mel, elmiller}@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 this important area. 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 significant speedup over software versions of the same algorithm, 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.

1. INTRODUCTION
This paper presents the implementation of parallel beam backprojection on reconfigurable hardware. This represents the first study applying reconfigurable hardware techniques to medical image processing applications. 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 [6]. It is applied in diagnostic medicine, baggage inspection, industry, astronomy, and geology. 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 [6]. In this paper, we focus on parallel-beam backprojection, but methods and results presented here can be extended to the fan-beam case.

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 $O(n^3)$. There is another algorithm whose complexity is on the order of $n^2\log_2 n$ [3]; however, its suitability for hardware implementation has not yet been investigated. Image reconstruction through backprojection is a highly parallelizable process. Such applications are good candidates for implementation in 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 20 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.

Another 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, especially in terms of required data transfer rates. Therefore, it can be beneficial for fixed-point implementations 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.

Finally, 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 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 related work.

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, $P(t, q)$, where t is the detector distance from the origin of the array, and q is the angle at which the measurement is taken. A collection of projections for different angles over 180° can be visualized in the form of an image in which one axis is position t and the other is angle q. This is called a sinogram or Radon transform of the two-dimensional function μ, and it contains information needed for the reconstruction of an image $\mu(x, y)$. The Radon transform can be formulated as

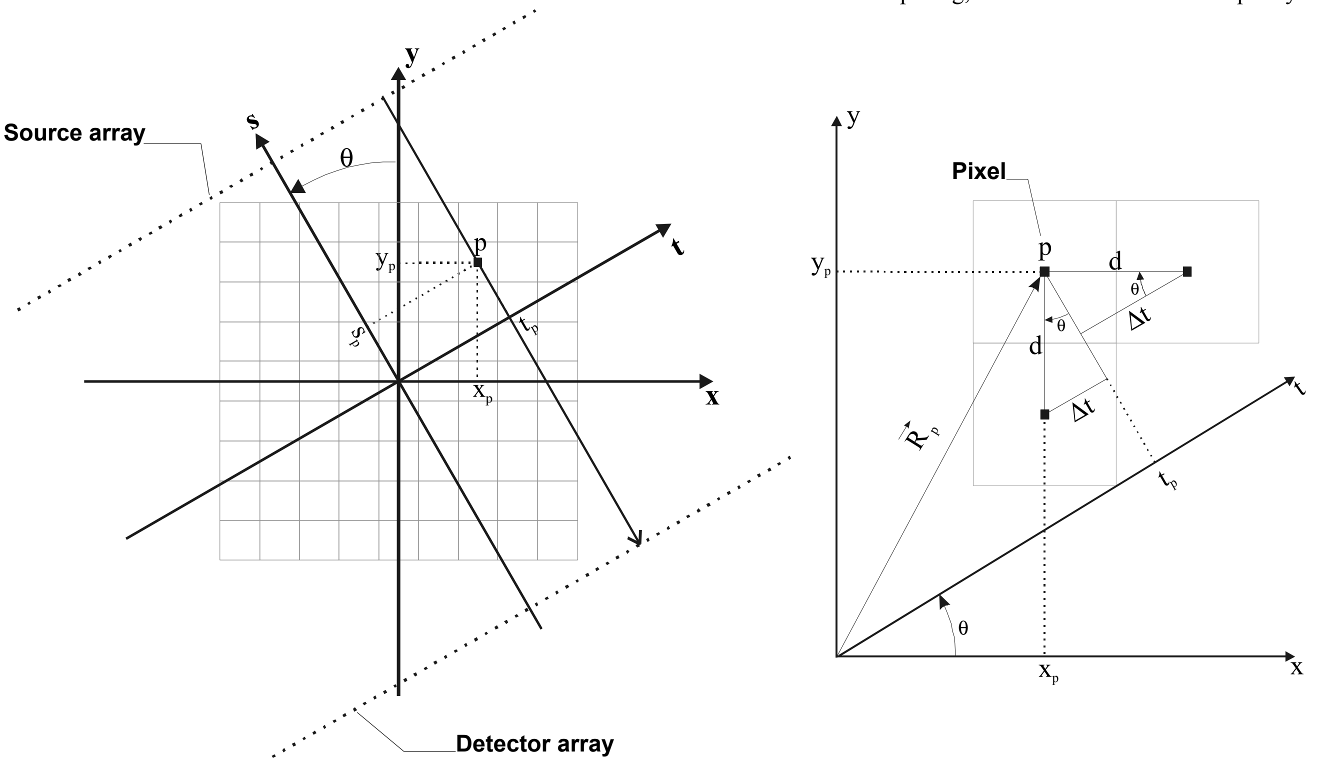
$$\log_e \frac{I_0}{I_d} = \iint m(x,y) d\left(x\cos q + y\sin q - t\right)dxdy \equiv P(t,q) \qquad (1)$$

where I_o is the source intensity, I_d is the detected intensity, and $d(\cdot)$ 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 $t = x\cos q + y\sin q$. The Radon transform represents an operator that maps an image $\mu(x, y)$ to a sinogram $P(t, q)$. Its inverse mapping, called the inverse Radon transform, when applied to a sinogram results in an image. The filtered backprojection (FBP) algorithm performs this mapping [6].

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

$$m(x, y) = \frac{\pi}{K}\sum_{i=1}^{K} P_{q_i}\left(x\cos q_i + y\sin q_i\right), \qquad (2)$$

where $P_q(t)$ is a filtered projection at angle q, and K is the number of projections taken during CT scanning at angles q_i over a 180° range. The number of values in $P_q(t)$ depends on the image size. In case of $n \times n$ pixel images, $N = \sqrt{2}nD$ detectors are required. The ratio $D = d/t$, where d is the distance between adjacent pixels and t is the detector spacing, is a critical factor for the quality of



(a) (b)

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

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, q. For each q, 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 $P_q(t)$ 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

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

where IF denotes the interpolation factor, $P_q(t)$ is the 1024 element array containing filtered projection data at angle q, 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 for two reasons. First, this substantially reduces the amount of data that must be transmitted to the reconfigurable board. Second, interpolation in hardware is much faster, thus speeding up the reconstruction process.

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 Δt 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 $t = x\cos q + y\sin q$. Assuming that pixels are processed in raster-scan fashion, then $\Delta t = d\cos q$ for two adjacent pixels in the same row ($x2 = x1 + d$) and similarly $\Delta t = d\sin q$ for two adjacent pixels in the same column ($y2 = y1 - d$). Our implementation is based on pre-computing and storing these deltas in look-up tables. 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 q required to update the pixel at row 1, column 1. LUT 2 stores the increment in t required as we increment columns. LUT 3 stores the increment for columns

3. QUANTIZATION

We quantize all our data and 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 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 [5], which generates its 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. We compute the quantization error by comparing a fixed-point image reconstruction with a floating-point one.

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.

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

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

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:

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

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

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

where ws is the word size in bits of integer Q, and *round* represents rounding to nearest. Since sinogram data are unsigned numbers, in their 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 quantization value Q is a 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, 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 evaluation of the error introduced by quantization. Our goal was to find a metric that would accurately describe visual differences between

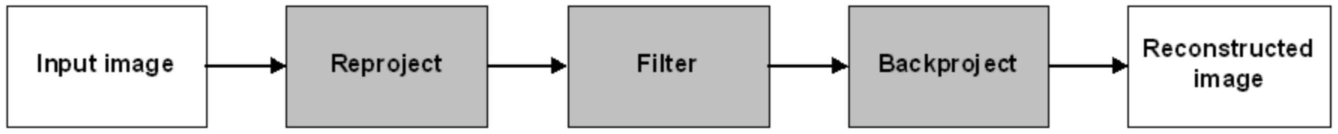


Figure 2: Major simulation steps

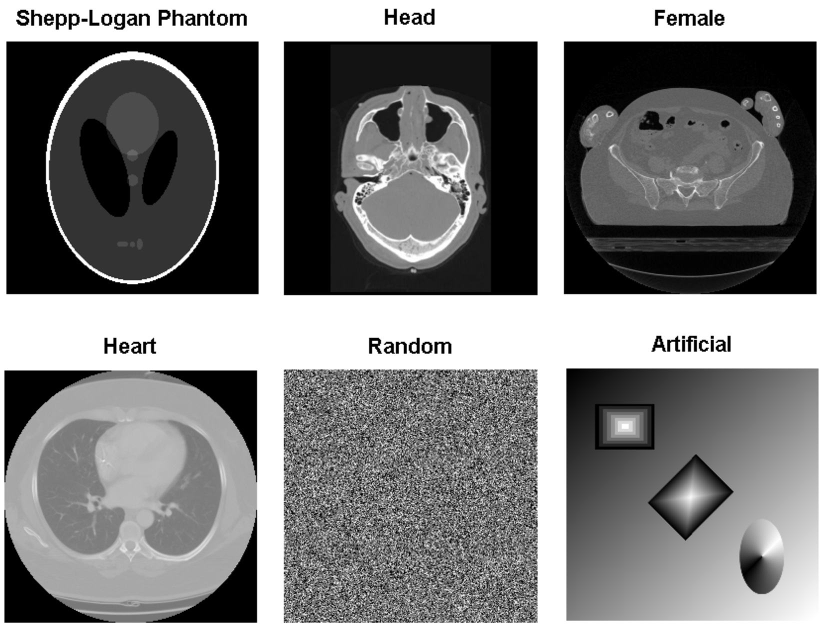


Figure 3: Some of the images used as inputs to the simulation process

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):

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

Here, N 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, $\bar{x}$ and $\bar{y}^{FP}$ and 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 [6]. Other medical images were Female, Head, and Heart. They are all generated by commercial CT systems. 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. This is also characteristic of the Head image, which contains a rectangular border around the head slice.

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

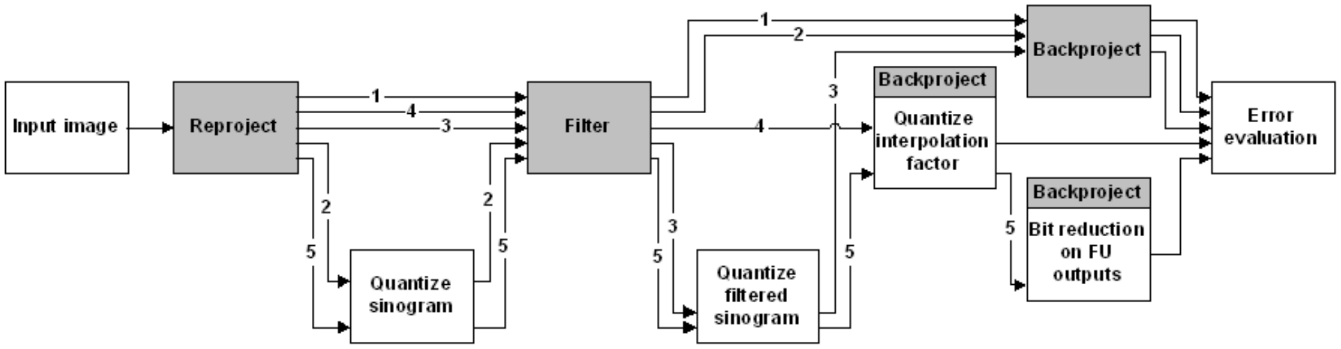


Figure 4: Detailed flowchart of the simulation process

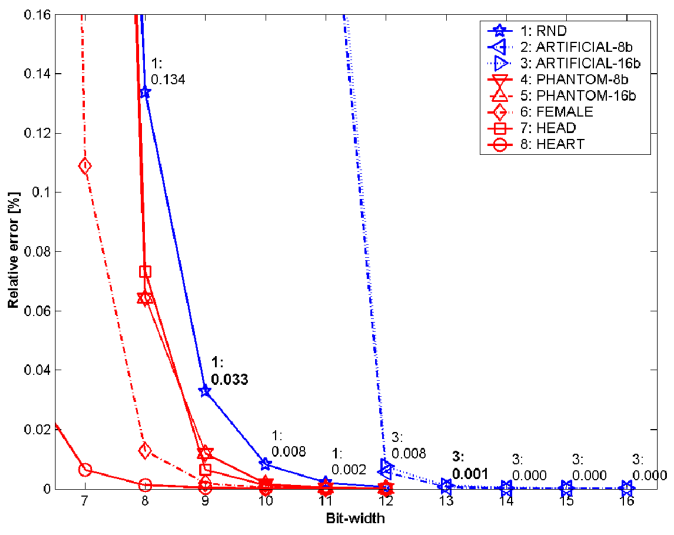


Figure 5: Simulation results for the quantization of filtered sinogram data

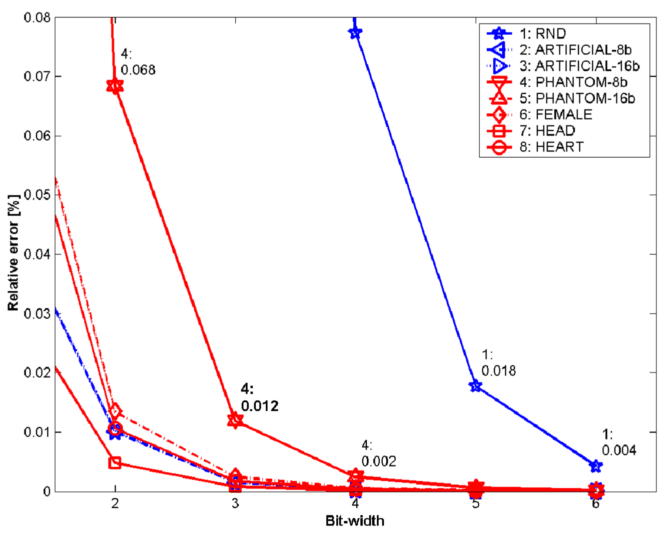
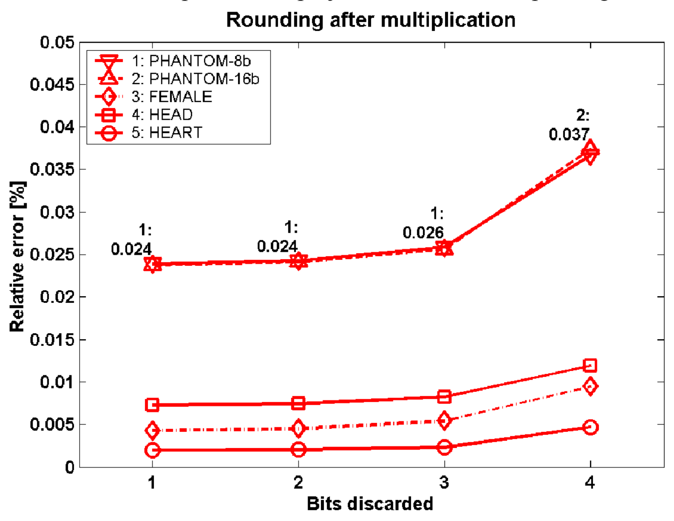


Figure 6: Simulation results for the quantization of the interpolation factor

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 reprojection 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 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 7 and 8, which require 13 bits. From Figure 6 (path 4 in Figure 4), we conclude that 3-bit words (meaning the maximal error for the spatial address is 2^{-4}) are sufficiently accurate. As expected, image 1 is more sensitive to the precision of the linear interpolation because of its randomness.



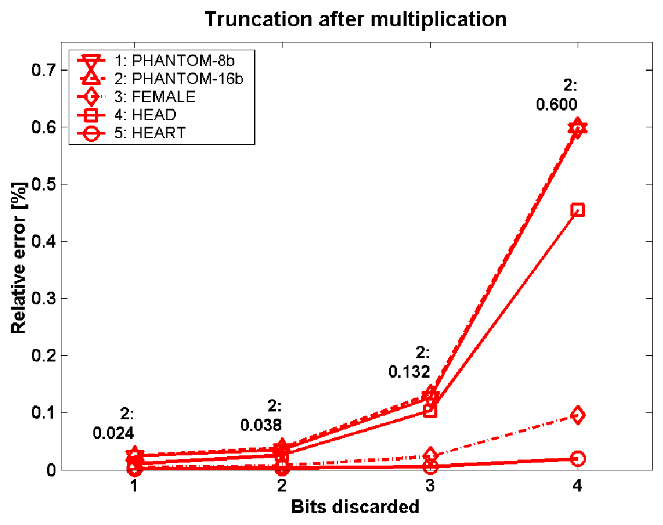


Figure 7: Bit reduction on the output of the interpolation multiplier

We also investigated whether it is feasible to disregard 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 symbolizes simulation cycles that investigate bit reduction at 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 7 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 9 and 10). However, in our case there was no need for using bit reduction to achieve smaller resource consumption because the targeted FPGA chip (Virtex1000) provided sufficient logic resources.

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 $P_q(\cdot)$, 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. Since in simulations we are using a 3-bit interpolation factor quantized by rounding to nearest with error of up to 2^{-4}, in hardware the spatial address *addr* should be calculated with 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 $2^{-6} + 512\cdot2^{-16} + 512\cdot2^{-16} + 2^{-5}$ 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.

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. Therefore, we decided to base our architecture primarily on projection parallelism.

Figure 8 shows a functional view of the particular hardware implementation that is presented here. There are three flows of data going on simultaneously that can be identified from this diagram: pipeline flow, sinogram data flow, and accumulation data flow.

The pipeline flow, which is horizontal in Fig. 8, consists of seven blocks, each representing one stage of the pipeline. 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.

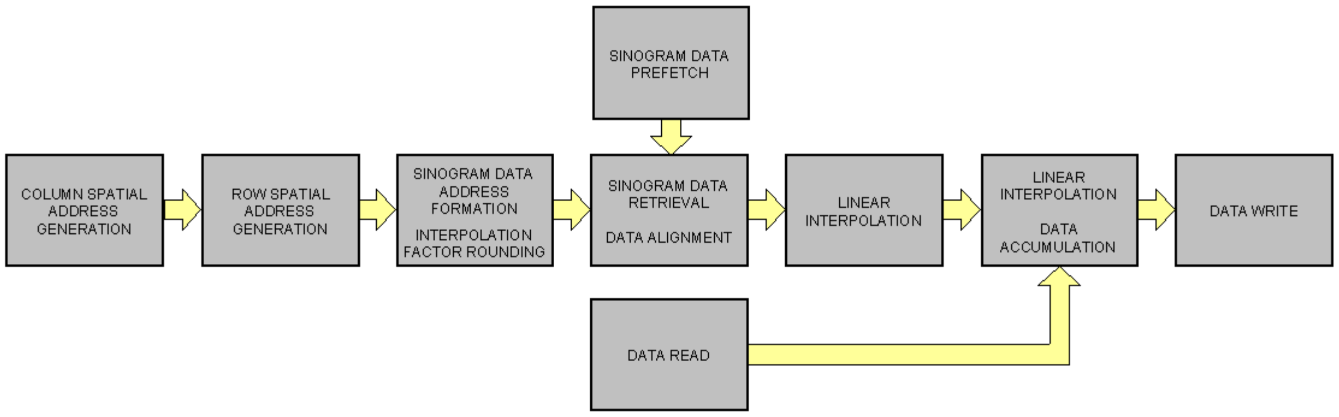


Figure 8: Data flow of the backprojection hardware

The fourth stage retrieves sinogram data from the on-chip memory banks and aligns them to proper inputs of the subtracter 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 7 completes the interpolation and accumulates the result to the value for the same pixel reconstructed from previous projections. The last stage stores this updated reconstruction into an off-chip memory bank.

The sinogram data 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.

The accumulation data flow (ADF) brings partially reconstructed pixel values from an off-chip memory bank into the sixth stage of the pipeline where accumulation with reconstructions from new projections takes place. The ADF has to be synchronized with the pipeline flow so that in each clock cycle both flows in stage six correspond to the same pixel. This is achieved through the pipeline stall mechanism.

Figure 9 shows the datapath of the non-parallel backprojection hardware. 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.

A double buffering scheme is used for storing projections. Each buffer is comprised of two blocks of on-chip RAM, one storing odd addressed projection data, and the other storing even addressed data. These buffers represent the boundary between pipeline stages 3 and 4. While one projection is being processed from one of the buffers, the next is being loaded into the other. With every new projection, the buffers switch their roles.

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. After that, processing and fetching are overlapped. The filtered and quantized sinogram is stored in one off-chip memory bank (Mezzanine RAM). 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 requirement possible. If $\lfloor addr \rfloor$ is an odd number, the projection value that corresponds to this number is located in the

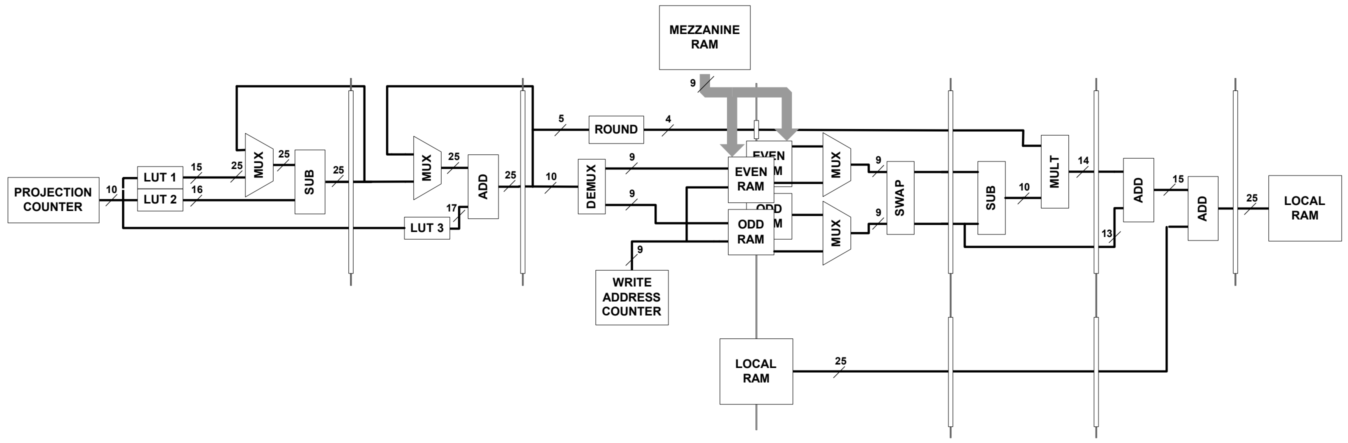


Figure 9: Datapath implementation of the non-parallel backprojection hardware

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, 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 (local RAMs in Figure 9) 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 local 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 local 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 ADF and pipeline flow are synchronized. The two registers inserted in the ADF enable detection of valid data coming from the source local RAM and give enough time for the control unit to stall or reactivate the pipeline.

The same basic principles of operation apply to the 4-way parallel architecture whose datapath is shown in Figure 10. The main difference is that this architecture processes 4 projections simultaneously. This requires replication of most pipeline resources so that 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 Mezzanine 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.

More projection parallelism can be extracted by processing more than four projections at the same time. This would require further replication of functional units and a larger addition tree in stage 6. The problem would be that the addition tree might not fit in a single pipeline stage. This is easily solved without adding a

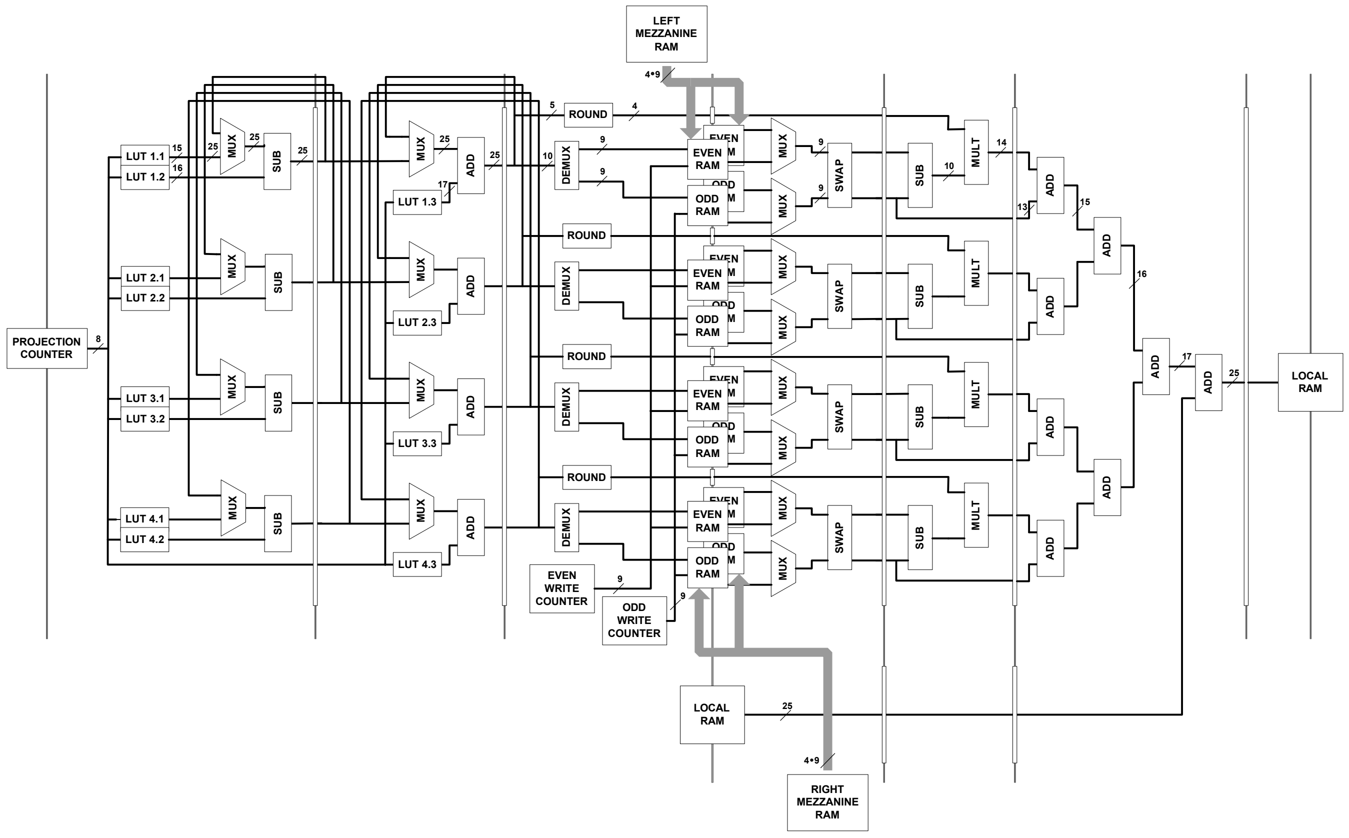


Figure 10: Datapath of the implemented 4-way parallel backprojection hardware

new stage by spilling excess units into stage 7, which (see Figure 10) is almost empty. The main limiting factor for extracting further parallelism is the amount of on-chip RAM. Our implementation uses the entire RAM available on the targeted FPGA chip (Xilinx Virtex1000), and 21% of logic resources. There are other similar FPGAs currently available, with larger RAM capacity, which can accommodate more than 4-way parallel implementations.

Theoretically, it would be possible to eliminate the use of on-chip RAM completely, and access sinogram data directly from off-chip memory, but there are some practical considerations that render this approach less desirable. The number of available memory banks, response time of the memory, number of I/O pins, increased access latency, increased cost in logic resources for control, and higher overall complexity of the design are some of the factors that would make such a solution inferior in terms of performance. Supplementing on-chip with off-chip memory would alleviate some of these difficulties, but that was not possible to implement in our case since we already used all the available memory banks. There is a possibility of using excess logic resources (look-up tables) of the chip to build additional RAM banks and thus make more parallelism extraction possible. However, this would significantly increase complexity of the design, making efficient place and route difficult, which could cause a large decrease in the clock frequency.

5. RESULTS AND PERFORMANCE

We have implemented parallel-beam backprojection on an Annapolis Micro Systems Wildstar board using one Xilinx Virtex1000 FPGA chip.

Figure 11 summarizes our performance results by comparing backprojection execution times in seconds for software and hardware implementations. The software implementation performs all calculations on integer values obtained after the quantizations have been performed (to be similar to the hardware implementation). The hardware implementation labeled as test case D is the 4-way parallel version (Figure 10), while C denotes the non-parallel version shown in Figure 9.

Figure 12 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.

6. RELATED WORK

Agi et. al.[1] present the only 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. [2] present a similar investigation of

A.	Software – 450 MHz Pentium:	**50 s**
B.	Software – 1 GHz Pentium:	**28 s**
C.	Hardware (non-parallel)- 50 MHz:	**5.4 s**
D.	Hardware (4-way parallel)- 50 MHz:	**1.3 s**

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

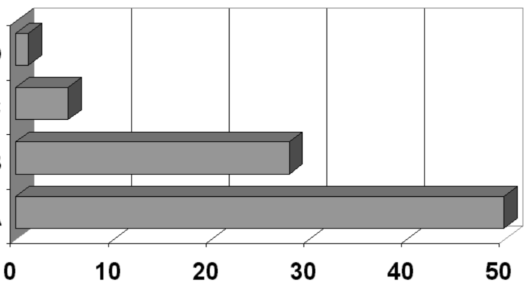


Figure 11: Performance results – Software vs. Hardware

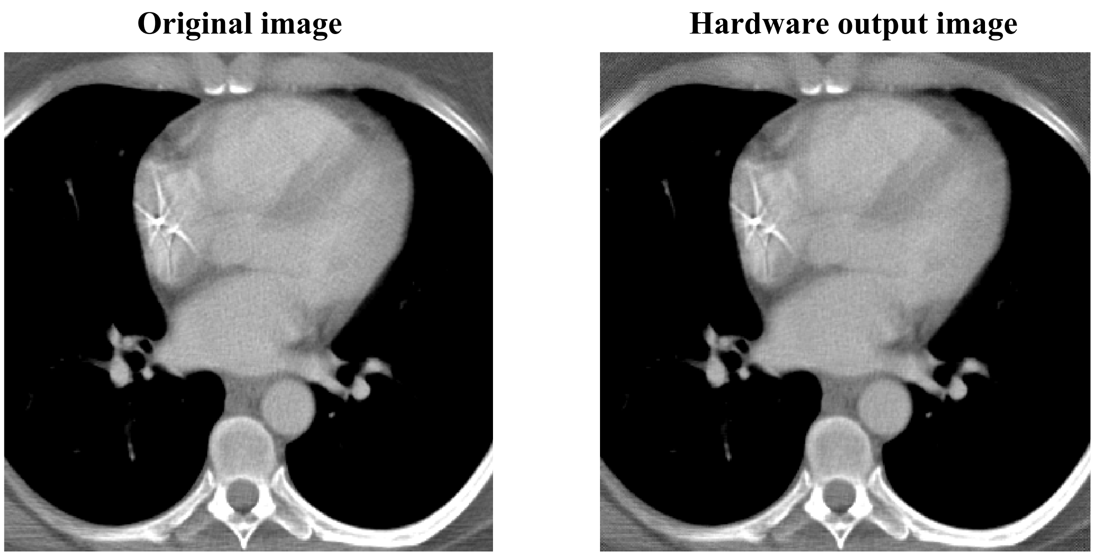


Figure 12: Image comparison – grayscale range mapped to a part of the pixel value range

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 FU 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}. There have been many papers applying reconfigurable hardware to image processing. Bins et. al.[4] 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. JPEG compression is an application that can tolerate a great deal more error than medical imaging.

7. 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 solution shows a 20 times speed-up over a similar software implementation. The combined effect of our quantizations result in a worst case relative error of 0.015% compared to a floating-point implementation. Real-time image reconstruction is easily attainable by exploiting the inherent parallelism of our solution to utilize 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.

8. REFERENCES

[1] Agi, I., Hurst, P.J., and Current, K.W. An image processing IC for backprojection and spatial histogramming in a pipelined array, in IEEE journal of solid-state circuits, vol. 28, no. 3, (1993), 210-221.

[2] Agi, I., Hurst, P.J., and Current, K.W. A VLSI architecture for high-speed image reconstruction: considerations for a fixed-point architecture, in Proceedings of SPIE, Parallel Architectures for Image Processing, vol. 1246, (1990).

[3] Basu, S., and Bresler, Y. $O(N^2 \log_2 N)$ filtered backprojection reconstruction algorithm for tomography, in Transactions of IEEE, Image Processing, vol. 9, no. 10, (October 2000).

[4] Bins, J., Draper, B., Bohm, W., and Najjar, W. Precision vs. Error in JPEG Compression. Parrallel and Distributed Methods for Image Processing III (SPIE), Denver CO, (July 22, 1999), 76-87.

[5] Joseph, P.M. An improved algorithm for reprojecting rays through pixel images, in Transactions of IEEE, Medical Imaging, vol. MI-1, no. 3, (November 1982).

[6] Kak, A.C., and Slaney, M. Principles of Computerized Tomographic Imaging, (New York, 1988), IEEE Press.