Efficient Computerized-Tomography Reconstruction Using Low-Cost FPGA-DSP Chip

Bassam A. Abo-Elftooh, Mohamed H. El-Mahlawy and Mahmoud E. A. Gadallah

***Abstract*—In this paper, filtered back-projection algorithm is optimally implemented using low-cost Spartan 3A-DSP 3400 chip. The optimization enables parallel implementation. The combination of the pixel parallelism and projection parallelism is presented to significantly reduce the total reconstruction time to produce the image. The applied data is presented in fixed point format to achieve efficient implementation with maximum speed. The selection of data bus-width is optimized with very little error and good visual quality required for medical images. Before implementation, the computer tomography (CT) reconstruction simulator is developed to provide a testing reference for the hardware implementation. Using the combination of the pixel parallelism and projection parallelism, the presented hardware design achieves image reconstruction of a 512-by-512 pixel image from 1024 projections in 134.8 ms using 50 MHz clock cycles. It achieves the reduction of the required number of clock cycles to form an image from projections by 60 % comparing to the state of the art of the reconstruction time using field programmable gate array (FPGA) design.**

***Index Terms*— CT image reconstruction using FPGA, Filtered Back-Projection (FBP), Pixel parallelism, Projection parallelism.**

Original Research Paper
DOI: 10.7251/ELS1721012A

I. INTRODUCTION

Computerized tomography (CT) is a medical image processing application, marked by computationally intensive algorithms. It is the process of generating a cross sectional image of an object from a series of projections collected around the object [1-2]. The reconstruction process uses these projections to calculate the average x-ray attenuation coefficient in cross-sections of a scanned slice. The most common approach for the CT image reconstruction is filtered back-projection (FBP). The FBP comes in parallel-beam and fan-beam variations [2]. Parallel-beam back-projection is focused on this paper, but the presented design in this paper can be extended to the fan-beam back-projection with modifications. The FBP includes digital signal processing (DSP) functions and requires accurate and fast processing of large data, which forces CT equipment manufacturers to strike a balance between image resolution, image generation time, and system cost when designing a new scanner.

Hardware implementation of FBP accelerates this processing. Field programmable gate array (FPGA) vendors introduce economical devices with large capacity and DSP capabilities. The FBP can benefit from the features of the FPGA. Furthermore, the fine-grained parallelism inherent to FPGAs is suitable for a highly parallelizable process like FBP [3-4].

Higher image resolution provides more diagnostic details to the radiologist, but it takes longer time to generate the images. It requires more processing power that significantly adds to the cost. Cost-effective acceleration of image reconstruction would allow manufacturers to generate higher resolution of the CT images while maintaining the required balance in their system design. In this paper, it is required to implement the parallel-beam FBP algorithm used in the CT on a single FPGA chip based on the mentioned features. Before hardware implementation, a software simulator is built to analyze and get accurate validation of each reconstruction process, used as a reference to validate each stage in the hardware design. The generated image from hardware is imported to the software simulator for the comparison with the generated image from the software. In addition, the simulator assists the hardware implementation to be optimized and minimized the hardware overhead.

Several previous works were attempted in the area of hardware implementation of the parallel-beam FBP algorithm. The authors in [5] demonstrated the fixed point architecture of the sinogram data quantized with 12-bit, and it does not be optimized for medical applications. The author in [6] showed the main divisions of CT scanner and suggests the areas that can be implemented using the Application Specific Integrated Circuits (ASICs) and the FPGA. The authors in [7] presented an ASIC hardware implementation of Radon transform, and the multiprocessing of the parallel-beam back-projection. The authors in [8-9] presented a proposal to accelerate the FBP but they do not show the hardware implementation and do not consider the quality of the medical image. The authors in [10]

Manuscript received 31 July 2015. Received in revised form 30 May 2017. Accepted for publication 28 June 2017.

Bassam A. Abo-Elftooh is with Electronics and Communication Engineering department, Faculty of Engineering Sciences and Arts, Misr International University (MIU), Cairo, Egypt (corresponding author: +201141978844; e-mail: engineerbassam@gmail.com).

Mohamed H. El-Mahlawy is with Electrical Engineering department, faculty of Engineering and Technology, Future University in Egypt (FUE), Cairo, Egypt (email: mohamed.elmahlawy@fue.edu.eg)

Mahmoud E. A. Gadallah is with the Modern University, Maadi, Cairo, Egypt (e-mail: mgadallah1956@gmail.com).

used fixed point implementation and maximized the parallelism but they deal with Joint Photographic Experts Group (JPEG) image that is not used in medical applications. The authors in [11] implemented the parallel-beam FBP algorithm based on projection parallelism. They achieved image reconstruction of a 512-by-512 pixel image from 1024 projections in 250 ms using expensive Virtex FPGA chip with 65 MHz. The main objective of this paper is to build hardware for fast reconstruction of high image quality using economical chip. Therefore, the combination of the pixel parallelism and projection parallelism is utilized using low-cost Spartan 3A-DSP 3400 chip with 50 MHz clock cycles. It achieves image reconstruction of a 512-by-512 pixel image from 1024 projections in 134.8 ms using 50 MHz clock cycles. The presented FPGA hardware implementation of the the parallel-beam FBP algorithm, based on the pixel parallelism and projection parallelism, is the most efficient in the speed and the hardware cost with respect to all previous published works.

This paper is organized as follows: section II presents the concept of the FBP algorithm. Developing of the CT image reconstruction simulator will be presented in section III. Developing of the non-parallel hardware implementation of the FBP algorithm will be presented in section IV. The parallel hardware implementation of the FBP algorithm will be presented in section V. Finally, the conclusion and the future work will be discussed in the last section.

II. Filtered Back-Projection Algorithm

Radon found the way for reconstruction of the image from projections, which is only one part of the CT system [12]. A practical method was made by Hounsfield [13], and Cormack independently invented the same process [14]. The basis of tomographic imaging is described in other papers [15-16]. In the BP, the measurements at each angle (projection) are smeared back along the same line (θ), as shown in Fig. 1. It is known the point of density is somewhere along that line. Therefore, a crude reconstruction results when the measured value is assigned along the entire line. Adding up the values for all θ will yield a picture of the object. When only six projections are used, a star-shaped pattern emanating from the dense points emerges as shown in Fig. 2(a). When enough projections are used, the blurring is apparent as shown in Fig. 2(b).

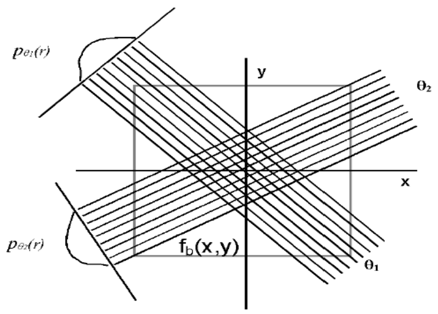


Fig. 1. Two projections are back-projected and added together.

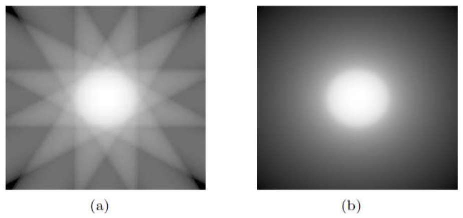


Fig. 2. Back-projecting of an object. a) Six projections. b) Blurred object.

The BP for a single projection of unknown density is [2]:

$$imn_{\theta}(x,y) = \int P_{\theta}(r)\delta(x\cos\theta + y\sin\theta - r)dr \tag{1}$$

where, $imn_{\theta}(x, y)$ is the back-projected density due to the projection $P_{\theta}(r)$. The δ () function is the Dirac delta function. Sum over all angles to obtain a summation image; $img_b(x, y)$.

$$img_b(x,y) = \int_0^{\pi} imn_{\theta}(x,y)d\theta \tag{2}$$

$$img_b(x,y) = img(x,y) ** \frac{1}{r} \tag{3}$$

where *img*(*x*,*y*) is the actual image, and ** is a 2-D convolution. The actual image is blurred by *1/r*. In frequency space, it means that the Fourier amplitudes have been multiplied by *1/ω*. This blurring must be removed (filtered). To fix that, it is required to pre-weight the Fourier Transform (FT) of each 1-D projection with |ω| prior to the BP, known as FBP and is still the standard technique in commercial scanners. This method is efficient because it only involves a 1-D FT which is used in practice. Alternatively, one may perform a space-domain convolution [17-18].

Due to its shape in the frequency domain, |ω| is what is called a ramp function. Such a filter in practice is impossible to build. It is infinite in length and it has the drawback of amplifying high frequency noise. A variety of different filters may be used instead. The basic and widely used filter is the Ram-Lak filter [19], but it still amplifies high frequency noise. Superior results are obtained by multiplying |ω| filter by a smoothing window that attenuates the higher frequencies which is mostly represents observation noise. The commonly used windows are presented in Fig. 3.

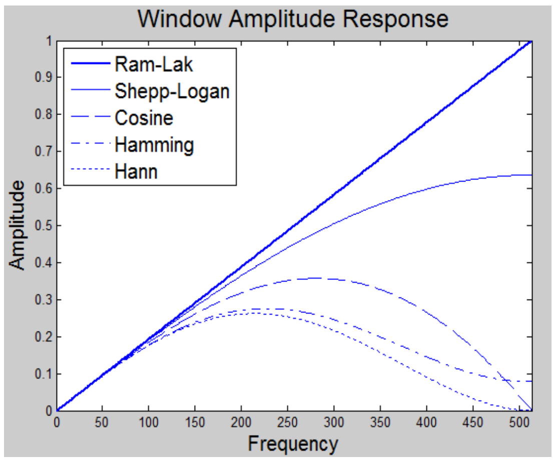


Fig. 3. Different smoothing windows

The projection data is measured into the polar coordinates while it is back-projected into the Cartesian coordinates of the final image so interpolation algorithm is required [20]. Various methods of interpolation algorithm are specified (nearest neighbor, linear or cubic). The interpolation can be performed either on the filtered projection samples (ray-driven) or on the reconstructed images (pixel-driven) [21]. At the first type, the filtered sample is traced along the x-ray path at a fixed increment. At each stop, the intensity of the sample is distributed to the neighboring pixels. Alternatively, we can start at an image pixel location and calculate the intensity contribution from the projection samples that intersect the ray pass through the center of this pixel. The pixel-driven takes place in the projection space (1-D) which makes it preferred over a ray-driven occurred in the image space (2-D) to save the intensive computation.

III. Development of the CT Image Reconstruction Simulator

The presented CT software simulator (CTSIM) in this paper is developed to simulate the CT image reconstruction process and to provide the reference for the validation of each stage in the hardware implementation [22]. The main components of the CTSIM are the object, the projections and the image reconstruction process [23]. Fig. 4 shows the block diagram of the CTSIM. The projections are generated from radon transform of Matlab Shepp-Logan phantom [24], test image, or text file including the projections (Sinogram). The object attenuation coefficients are represented by gray level intensities of the image. The first step of the image reconstruction algorithm is the filtration, done in either time or frequency domain as shown in Fig. 5, based on window selection and frequency band scaling. After that, the BP process is performed.

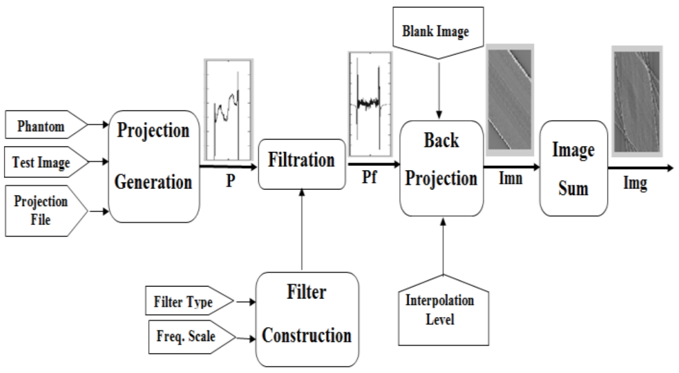


Fig. 4. General block diagram of the CTSIM.

In the BP process, the output image size is estimated from the projection length. The projection length is equal to the image diagonal. An empty image with this size is formed. This image is a 2-D matrix *T* that includes indices. Smearing the projection through this matrix is done by replacing each index by the gray level of the projection element with the same index, as shown in Fig.6. The BP is done with the selected interpolation algorithm.

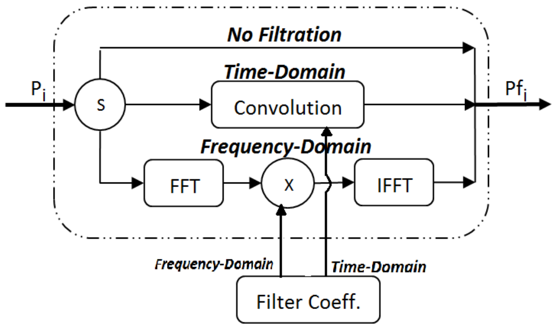


Fig. 5. The block diagram of the filtration process.

Matrix T rotates according to its corresponding angle through the equation:

$$T_\theta(x,y) = X\cos(\theta) + Y\sin(\theta) \tag{4}$$

This matrix is divided into an integer part (a_θ) and a fraction part (T_θ - a_θ). The integer part is used for addressing, while the fraction part is used as the interpolation factor of the BP process. After the BP process of each projection is finished, the corresponding image is:

$$imn_\theta(x,y) = (1-(T_\theta(x,y)-a_\theta(x,y)))*P_\theta(a_\theta) + (T_\theta(x,y)-a_\theta(x,y))*P_\theta(a_\theta+1) \tag{5}$$

Each new image is added to the previous one to produce the final reconstructed image:

$$img(x,y) = \sum_{\theta=0}^{179} imn_\theta(x,y) \tag{6}$$

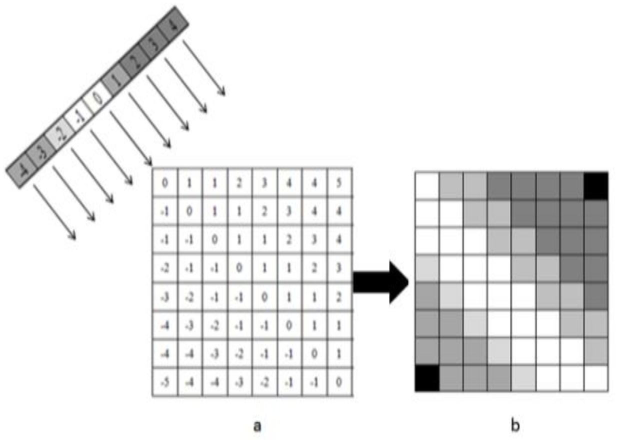
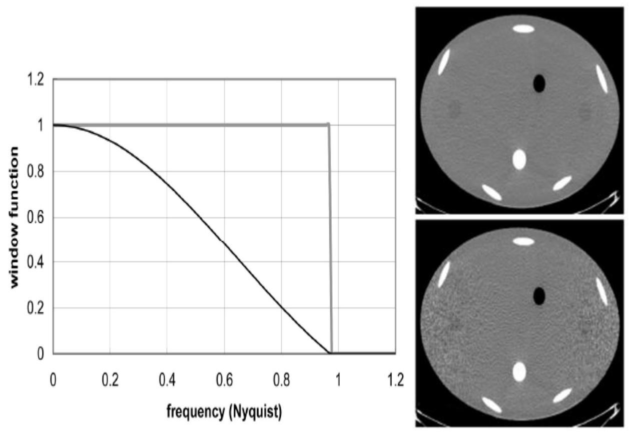


Fig. 6. Back-projection process at angle 45°.

During image reconstruction process, the program records the behavior of each projection separately through each stage to enable the tracing and analysis of the reconstruction process through different reconstruction stages. The impact of different windows on the reconstructed image is studied. For example, an oval phantom is reconstructed with a rectangular and a sinc function as shown in Fig. 7(a). The reconstructed images and the difference between them are shown in Fig. 7(b). Sinc function shows a significant noise reduction plus a slight reduction in spatial resolution [21]. The CTSIM monitors the filtration and reconstruction time.



(a) (b)

Fig. 7. (a) Different windows. (b) Upper image from Sinc window and the lower image from rectangular window.

Some techniques are used to evaluate the accuracy of the image reconstruction process using the CTSIM. The error is measured using the normalized mean absolute distance measure (ABS) and the worst case distance (WORST). The quality of reconstructed image can be evaluated using the peak signal to noise ratio (PSNR), mean structural similarity index (MSSIM) [25]. The PSNR and MSSIM tests use the Matlab Shepp-Logan phantom as a reference image. The projections are generated from this reference image and these projections are used to get the reconstructed image. They are calculated according to the following equations.

i- ABS: The normalized mean absolute error measurement.

$$ABS = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} | p_{i,j} - r_{i,j} |}{\sum_{i=1}^{n}\sum_{j=1}^{m} | p_{i,j} |} \tag{7}$$

where, p denotes the reference image and r denotes the reconstructed image. Each of the images has a size of $m \times n$.

ii- WORST: The worst case error over a 2 × 2 pixel area.

$$WORST = \max_{1 \le k \le [n/2], and 1 \le l \le [m/2]} (|P_{k,l} - R_{k,l}|) \tag{8}$$

where,

$$P_{k,l} = \frac{1}{4}(p_{2k,2l} + p_{2k+1,2l} + p_{2k,2l+1} + p_{2k+1,2l+1}) \tag{9}$$

$$R_{k,l} = \frac{1}{4}(r_{2k,2l} + r_{2k+1,2l} + r_{2k,2l+1} + r_{2k+1,2l+1}) \tag{10}$$

In equation 8, the term [n/2] and term [m/2] denote the largest integers less than n/2 and m/2, respectively.

iii- PSNR: the peak signal to noise ratio in decibels between two images. First calculates the mean-squared error:

$$MSE = \frac{\sum_{M,N} [p_{m,n} - r_{m,n}]^2}{M * N} \tag{11}$$

M and N are the number of rows and columns of input image, respectively, then computes the PSNR:

$$PSNR = 10\log_{10}(\frac{F^2}{MSE}) \tag{12}$$

where, F is the maximum fluctuation in the input image (255 for 8-bit).

iv- MSSIM: The mean structural similarity index between two images. If one of the images being regarded as perfect quality (phantom or test image), then MSSIM can be considered as the quality measure of the other image (reconstructed) [25]. When the projection source selection is a phantom or test image, we have a reference image that provides us the opportunity to test the output image compared to the input one.

In addition, the CTSIM helps to trade-off the interpolation algorithm which affects the smoothness at the cost of image reconstruction time as shown in Fig. 8. Nearest neighbor interpolation is the fastest method. However, it provides the worst smoothness. Linear interpolation slightly requires more execution time but its results are continuous and smooth. Cubic results are close to those of linear interpolation but, it requires extremely more execution time. Image quality is shown in Fig. 9.

Finally, another analysis is available; sinogram analysis. If a certain detector doesn't work probably, zero corresponding projection element value or a problem occurs at a certain angle. It is no projection element values at this angle during projections acquisition process. This will be detected by scanning all sinogram elements. This analysis guarantees complete data acquisition process.

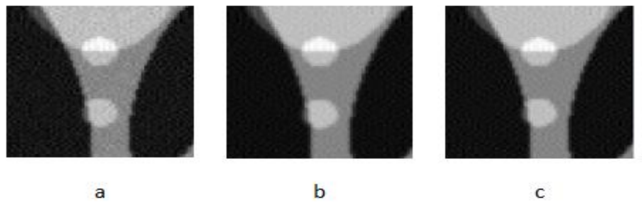
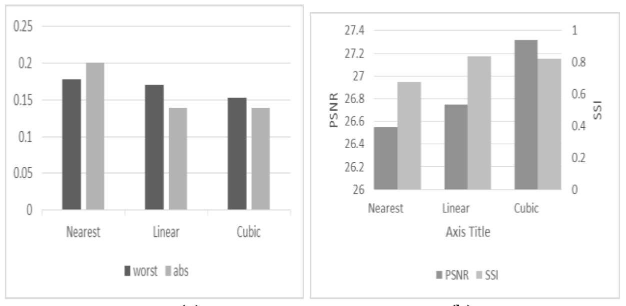


Fig. 8. a) Nearest neighbor, b) linear and c) cubic interpolation algorithm.



(a) (b)

Fig. 9. (a) Worst case and absolute error. (b) PSNR and MSSI.

IV. Development of Non-parallel Hardware Implementation of FBP

The main goal of this section is to reconstruct 512-by-512-pixel image from sinogram of 1024 projections (each projection has 1024 samples). The presented non-parallel hardware implementation is enhanced by applying the data in fixed point format. The block and the schematic diagram of the proposed FPGA implementation are shown in Fig. 10 and Fig. 11, respectively. It consists of the FBP main block, the projection buffering memory and the image summation memory. These two memories are off-chip to provide adequate space for the implemented FBP algorithm.

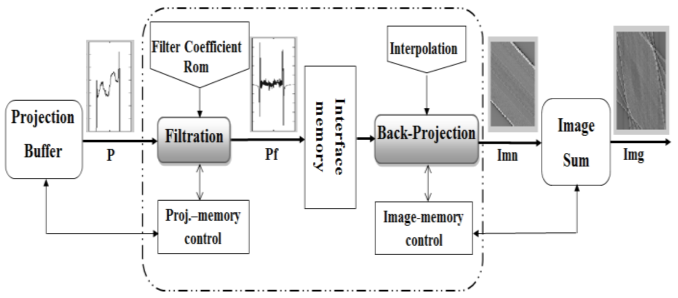


Fig. 10. The main block diagram of the presented implemented algorithm.

The main block has two controllers; the first one controls the projection acquisition process while the other controls the output image summation. FBP main block consists of three blocks; filtration, back-projection and ping-pong dual-port memory. After a projection is acquired and filtered, it is back-projected.

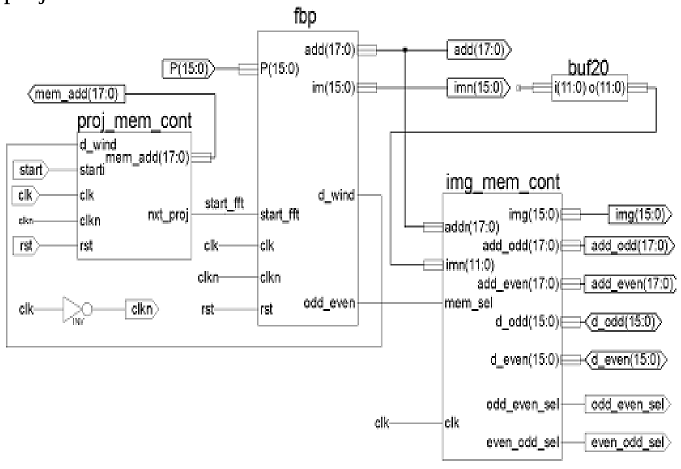


Fig. 11. Schematic diagram of the top level implemented algorithm.

A. Projection acquisition

The projection is acquired through projection memory controller, shown in Fig. 10 and Fig. 11, from the *projection buffering memory*. The controller organizes handshaking between the projection buffering memory and the FBP module, through two control signals (*start_filt*, *d_wind*). The first control signal tells the filtration module that there is a projection ready to be filtered. The filtration module in turn responds with control signal *d_wind* which enables projection acquisition process from the memory, shown in Fig. 12. Using the CTSIM, presented in section III, it is found that the suitable format to represent the projection (sinogram data) is (1, 8, 7) for sign, integer, and fraction, respectively. The sinogram has 1024 projections (each projection has 1024 samples). The required memory capacity is 1 Mega words. The word of the projection sample is 16 bits.

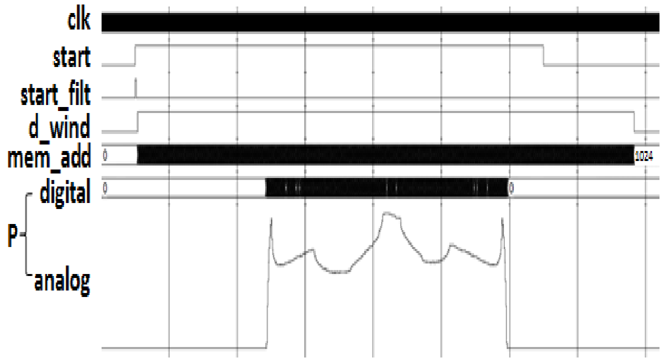


Fig. 12. Control signals and projection extraction.

B. Filtered back projection (FBP)

The FBP subsystem receives the acquired projection, *P*(15:0), from the external *projection buffering memory* and produces a corresponding image, *imn*(15:0). It is divided into three parts, shown in Fig. 13 and explained as follows.

1. *Filtration*: The filtration is implemented in the frequency domain with direct mapping from the software of the simulation (Fast Fourier Transform (FFT), filter coefficient multiplication, and then Inverse Fast Fourier Transform (IFFT)). Fig. 14 shows the timing diagram of the filtration process. Most signals in Fig. 14 are internal signals of the filtration core in the schematic diagram of Fig. 13. The FFT is implemented with FFT intellectual property (IP) core [26] that is used in the pipelining architecture to provide faster transform but with maximum hardware utilization. After the assertion of the control signal *start_filt*, the FFT core starts and produces required indices, *xn_index*, corresponding to the order of the input samples which are used to address the projection buffering memory.

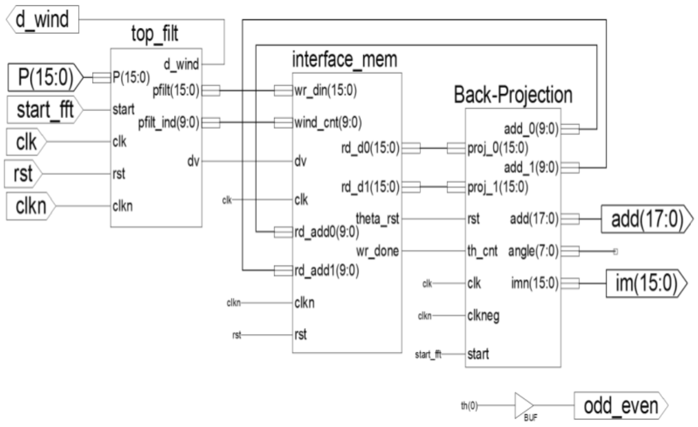


Fig. 13. Schematic diagram of the FBP.

The projection samples are applied to the real input of the core (*xn_re*), while the imaginary input (*xn_im*) is grounded. The core produces FFT real and imaginary outputs *xk_re* and

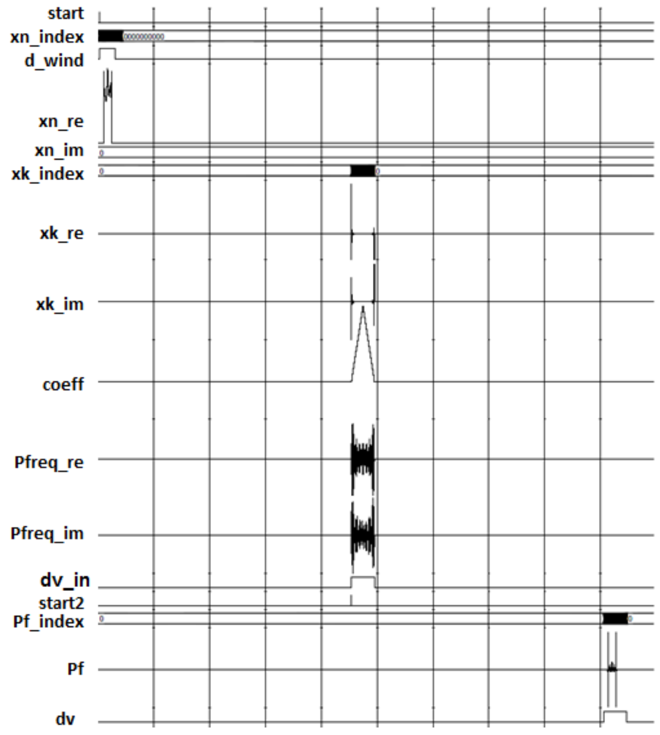


Fig. 14. Timing diagram of the filtration process.

xk_im with its corresponding indices, *xk_index*. These indices address the ROM containing the filter coefficients to synchronize the output samples with its corresponding coefficients. The filter coefficients are exported from the CTSIM, presented in section III, to a COE file and loaded to the ROM during the FPGA chip download. The FFT real and imaginary outputs are then multiplied by the filter coefficients. The input sample (1, 8, 7) format is expanded through the FFT core to be (1, 19, 7) output sample. The coefficients format is (0, 0, 9) and therefore the multiplication output is (1, 19, 16). This output could be rounded from 36 bit to 16 bit with format (1, 8, 7). The multiplier is implemented with three stages of pipelining that minimizes the hardware utilization and, in the same time, provides the IFFT core the desired three clock cycles latency [26].

The multiplier outputs are applied to the IFFT, implemented with pipelining architecture. The output of the IFFT has (1, 19, 7) format which in turn rounded to 16 bit with (1, 14, 1) format. The output of the IFFT has to be divided by the number of samples (N = 1024) that is done by shifting the data 10 bits to the right. So the filtered projection has (1, 4, 11) format. All the rounding schemes depend on the analysis done by the CTSIM for the maximum and minimum possible values in each stage. The IFFT produces three outputs; the filtered projection, *Pf*(15:0), output indices, *Pf_ind*(9:0), to address the memory to which *Pf* be written and control signal *dv* which tells the memory that the *Pf* data is valid.

Using 50 MHz clock cycles, the latency of the FFT core for 1024 input samples is 63.66 µs (3183 clock cycles), so the total filtration time equals to FFT latency plus multiplication time, and IFFT latency. But the multiplication is executed simultaneously with the FFT outputs with three clock cycles latency (3x20ns = 0.06 µs). Therefore, the filtration process takes 127.38 µs (63.66 µs + 0.06 µs + 63.66 µs), based on 6369 clock cycles.

2. *Ping-Pong Dual-Port Memory (PPDM)*: The PPDM is the interface memory, illustrated in Fig. 13, between the filtration module and the back-projection module. This memory includes two internal RAMs; M_1 and M_2. It is designed to let the current filtered projection Pf_i be written to M_1 while the previous projection Pf_{i-1} is back-projected from M_2. Therefore, the filtration does not need to wait until the BP finishes, which enhances the speed of the design implementation. After that projection Pf_i is read from M_1 while projection Pf_{i+1} is written to M_2. Therefore, the memory PPDM is called ping-pong memory. When one of the internal RAM is in the read mode, the other will be in the write mode. In the write mode, the RAM is write-enabled and addressed with P_f*_ind*(9:0). In the read mode and from equation (5), the BP reads two successive samples simultaneously, so the RAM needs two data outputs read from two addresses. The PPDM enables the BP to simultaneously read two successive samples. The address inputs of the PPDM, *rd_add0*(9:0) and *rd_add1*(9:0), are fed from the address outputs *add_0*(9:0) and *add_1*(9:0) of the BP module, respectively. The data outputs from the PPDM, *rd_d0*(15:0), and *rd_d1*(15:0), simultaneously feed the BP module.

3. *Back-projection*: Direct implementation of equation (5) consumes two multipliers. This equation can be modified to:

$$imn_{\theta} = P_{\theta}(a_{\theta}) + (T_{\theta} - a_{\theta})(P_{\theta}(a_{\theta}+1) - P_{\theta}(a_{\theta})) \qquad (13)$$

This modification reduces the resources to one multiplier only. The hardware implementation of this process is done using three blocks as shown in Fig. 15. The first block (*T_gen*) that generates matrix T_θ with its integer part *tint*(*10:0*) and fraction part *tfr*(*13:0*), the second block (*Proj_address*) uses the integer part for addressing of the *PPDM* containing P_{filt}, and the third block (*BP_interp*) uses the fraction part as the BP interpolation factor (T_θ- a_θ).

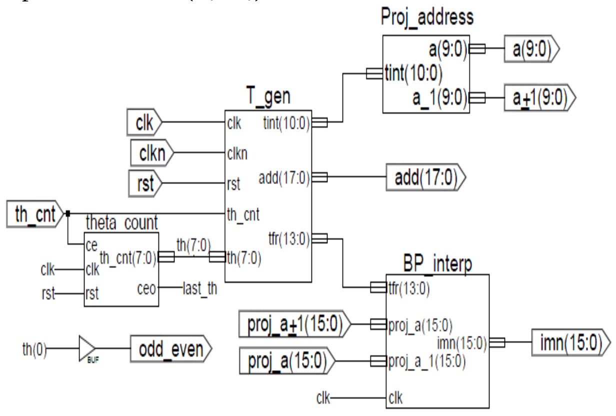


Fig. 15. Schematic diagram of the back-projection.

Matrix generator (*T_gen* shown in Fig. 15) generates matrix T_θ by direct hardware realization of equation (4). Two counters (X and Y) generate the Cartesian coordinates starting from the top left pixel of the image in a zigzag direction.

Counter X starts from -256 to 255, while counter Y starts from 255 to -256. Counter Y is enabled when counter X ends to 255, to go to the next row and counter X starts from -256 again. Counter X and counter Y format is 9 integer bits. The cosine and sine values of the projection angles are stored in two ROMs in a (1, 1, 14) format. The outputs of two counters (X and Y) and two ROMs (COS & SIN) are asserted at the rising edge, while the multiplication is done at the falling edge to emphasize enough setup and hold time for the multiplier. The resultant T_θ is 25 bits; with fixed point format (1, 10, 14). From equation (13), T_θ will be used as two separate parts; the integer part (*tint*) and the fraction part (*tfr*). An 18-bit counter is enabled for each T_θ to generate the address (*add*(17:0)) corresponding to each pixel from 0 to 512^2. 0 is the address of the pixel that has coordinates (-256, 255) which is the first pixel in the zigzag, while address 512^2 corresponds to last pixel with coordinates (255, -256).

The block *Proj_address*, shown in Fig. 15, uses the integer part a_θ to address its corresponding projection samples in the *PPDM*. The integer part of matrix T_θ (*tint*) is supposed to be a matrix of positive values resembling addresses but actually it has negative values, so the center of *tint* has to be shifted by adding a positive value to all of *tint* elements. This value equals to the minimum possible negative value plus one (not to have zeros), which is the left bottom corner of $T_{45} = -256 \times \cos(45^o) - 256 \times \sin(45^o) \approx -363$. The added value is 363 + 1 = 364, i.e. $a_\theta = tint + 364$. The addressing part of the BP, *Proj_address*, generates two successive addresses (a_θ, $a_\theta + 1$) to the *PPDM* to read two successive samples ($proj(a_\theta)$, $proj(a_\theta+1)$). These samples have to be interpolated, discussed in section III.

The BP with linear interpolation (*BP_interp*) is implemented using equation (13). Two inputs projection samples, read from the *PPDM*, have (1, 4, 11) format. These two samples are subtracted, and the output of the subtractor has the same format. The interpolation factor (*tfr*) format is (0, 0, 14). This factor is multiplied by the output of the subtractor. The multiplication output format is (1, 4, 25), rounded to have the same format of the projection sample $proj(a_\theta)$; (1, 4, 11). Then, this output is added to the projection sample. Each output of the adder represents an image pixel with (1, 4, 11) format that forms the desired back-projected image, *imn*. Image *imn* is generated pixel by pixel with its corresponding address. To form an image with 512-by-512 pixel, it takes 512^2 clock cycles. That image takes 5.243 ms using 50 MHz clock cycles.

C. Image Memory Accumulator

The new back-projected image is added to the previous ones through image accumulator memory. The accumulator consists of two RAMs. The first image is written to a RAM (odd RAM). Then the second image is added to the image stored in the odd RAM and the result is written to the other RAM (even RAM). The third image is added to the image stored in the even RAM and the result is written to the odd RAM and so on. In general, the new image is added to the sum of the previous images stored in a RAM, and the result is written to the other RAM under the control of the image memory controller, *img_mem_cont*, shown in Fig. 11. Therefore, when a RAM is in read mode, the other is in write mode as shown in Fig. 16.

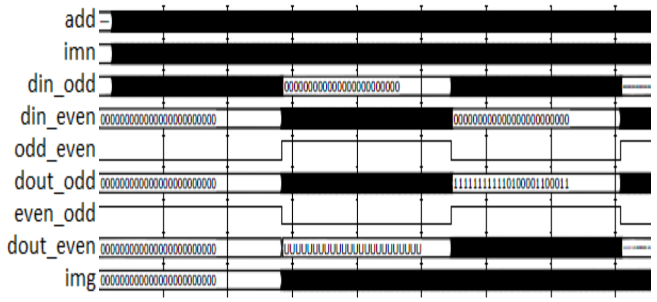


Fig. 16. Timing diagram of first Image accumulation process.

When a new odd image, *imn*, is generated from the BP, the signal, *odd_even*, sets to low, and the data is read from the even RAM (*dout_even*) which is the sum of the previous images. The odd *imn* is added to this sum and the new sum is written to the input of odd RAM (*din_odd*). Next, when an even image is generated, the control selections, *odd_even* and *even_ odd*, are inverted and so on. To avoid the overflow due to the sum accumulation, the least significant four bits of pixel value of the image *imn* are neglected. So the accumulated image format is (1, 8, 7). The effect of neglecting these bits will be studied in the next subsection by comparing the output image from the hardware (with four bits reduction) to the corresponding image from the CTSIM (without four bits reduction). The accumulation is simultaneously done with the BP process.

D. Results

After the accumulation of all images, the last accumulated image is exported quantized to its 16-bit fixed point format (1, 8, 7). The data are arranged as 1-D from address 0 to 512^2, so it is rearranged from 1-D series of data into a 2-D pixel by pixel matrix using the zigzag scheme. The output image is shown in Fig. 17.

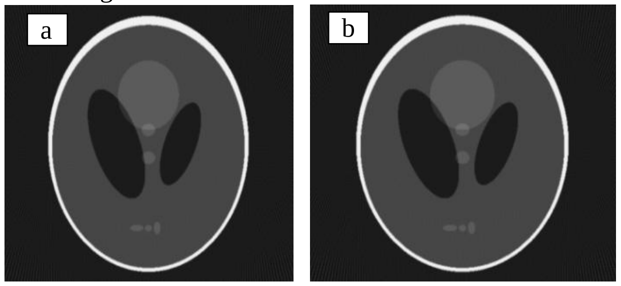


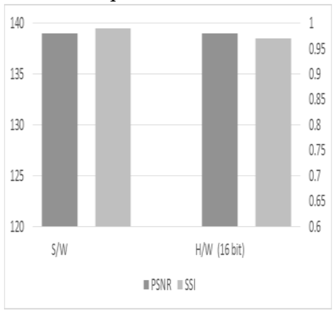
Fig. 17. Output image from (a) Software based on the CTSIM (b) Hardware implementation (16-bit).

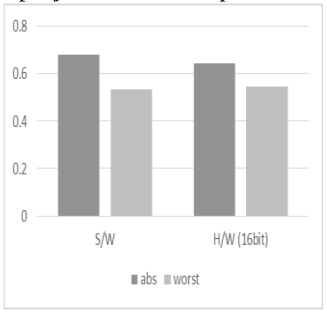
Finally, the image generated by hardware implementation is compared to the image generated by the CTSIM. This comparison is evaluated based on the accuracy of the image quality. The image quality is tested in two ways. The first way is the subjective test that visually compares between the reconstructed images, based on the CTSIM and the hardware implementation, shown in Fig. 17. The second way is the objective test that contains the PSNR, the MSSIM

(Percentage), the ABS and the WORST, shown in Fig. 18. As mentioned before, Matlab Shepp-Logan phantom is used as a reference image. For more information of the test image used, how it was obtained and its projection data was obtained, refer to [22].

It is obvious that the subjective test and the objective test are almost the same that indicates that the design is functionally successful.

The BP process takes 512^2 clock cycles. So, there is enough time to acquire and filter the next projection. The acquisition





(a) (b)

Fig. 18. Software based CTSIM vs. hardware implementation (16-bit): (a) Image quality b) Errors measurement.

and filtration of the next projection is synchronized to let the BP of this projection start just after the BP of the current one is completed. That saves time and let the overall time limited to the BP time of all projections plus the filtration time of the first projection only. The summation of the back-projected images is executed with the BP pixel by pixel. The exemplar timing diagram of the 1st, 2nd, intermediate and last projection processing is shown in Fig. 19. The time required to produce the final image, *img*, is the required time to perform the BP process multiplied by the number of projections (5.243 ms x 1024) plus the first projection filtration time (127.38 µs). The total reconstruction time of the non-parallel hardware implementation is 5.37s.

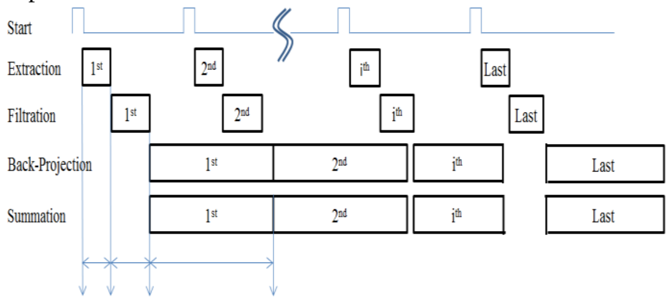


Fig. 19. Timing diagram of the implemented algorithm.

Now, the non-parallel hardware design presented in this section is ready to the timing simulation. The presented design is synthesized, mapped, placed and routed on the targeted chip Spartan 3A-DSP 3400. Device utilization is shown in Table I.

E. Optimization

In this section, the optimum hardware utilization will be considered without affecting the reconstruction time and the image quality. It is done based on hardware reduction of the FFT IP core, the reduction of the data-bus width, and the efficient utilization of the chip supplied benchmarks, DSP and RAM Blocks. The time reduction through the parallelism of the image reconstruction process will be presented in the next section.

TABLE I
DEVICE UTILIZATION SUMMARY ON XC3SD3400A

Logic Utilization	Used	Available	Utilization
Number of Slice Flip Flops	25,448	47,744	53%
Number of 4 input LUTs	38,857	47,744	81%
Number of occupied Slices	22,890	23,872	95%
Total Number of 4 input LUTs	39,956	47,744	83%
Number of bonded IOBs	144	469	30%
Number of RAMB16BWERs	11	126	8%
Number of DSP48As	0	126	0%

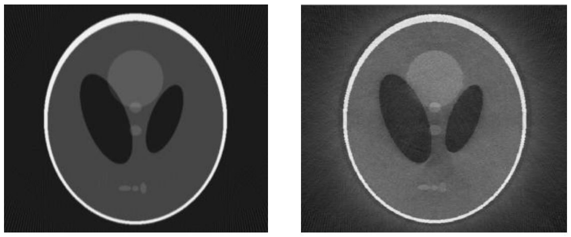
1. *FFT IP Core Implementation Architecture*: However, the filtration, the BP, and the image accumulation are done simultaneously, it is noticed that the filtration time is much smaller than the time required for the BP or the image accumulation. Therefore, the filtration can be performed slower to get the benefit of the hardware reduction without affecting the overall speed or image quality. Radix-2 lite FFT core is implemented with much smaller resources at the expense of additional latency [26]. This latency will not affect the BP time. The FFT latency is 246.44 µs (12320 clock cycles), so the filtration time (FFT latency + Multiplication time + IFFT latency) will be 246.4 + .06 + 246.4 = 492.86 µs (24643 clock cycles). The BP time is 5.243 ms, so the overall speed is not affected while we gain a considerable reduction in the utilized resources of the target chip. The device utilization is shown in Table II.

TABLE II
DEVICE UTILIZATION SUMMARY WITH FFT LITE ARCHITECTURE.

Logic Utilization	Used	Available	Utilization
Number of Slice Flip Flops	4,699	47,744	9%
Number of 4 input LUTs	11,653	47,744	24%
Number of occupied Slices	7,215	23,872	30%
Total Number of 4 input LUTs	11,820	47,744	24%
Number of bonded IOBs	144	469	30%
Number of RAMB16BWERs	11	126	8%
Number of DSP48As	0	126	0%

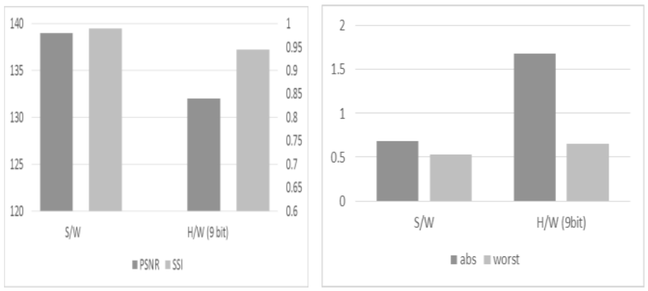
2. *Data-Bus Width*: Another issue that reduces the utilized hardware area but may affect the image quality is the width of the data bus. This width is reduced from 16 bits to 9 bits. Therefore, the effect of the bus-width reduction on the image quality needs to be studied. This effect is tested by comparing the reconstructed image before and after the bus-width reduction. The Subjective test is shown in Fig. 20, and the objective test is shown in Fig. 21. It is obvious that the image quality is greatly affected by the bus-width reduction from 16 to 9 bits. This effect is in both the subjective and the objective test.

To overcome this problem, we used the CTSIM to analyze the data values through all the reconstruction stages. It is found that the data is enlarged by the image accumulation process.A proposed solution is to use different data bus-width before and after the image accumulation. The 9-bit bus-width



(a) (b)

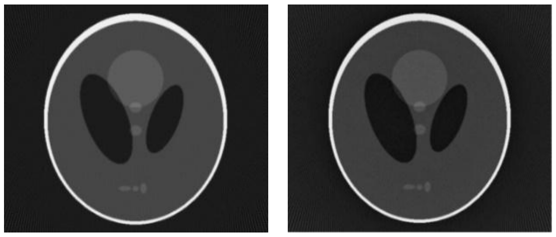
Fig. 20. Software based the CTSIM vs. Hardware implementation (9-bit): a) Original image b) Reconstructed.



(a) (b)

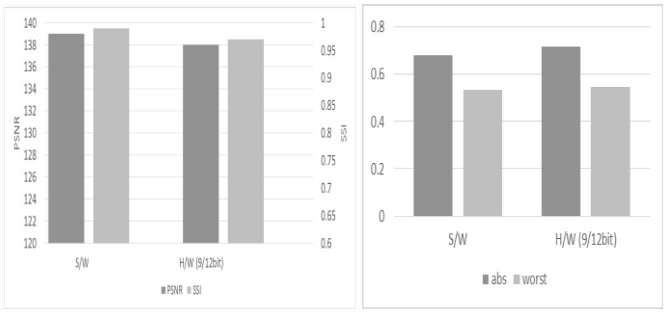
Fig. 21. Software based the CTSIM vs. Hardware implementation (9-bit): a) Image quality b) Error measurement.

represents the data before the image accumulation while, after the accumulation, the bus-width is increased by 3 bits. This 9/12 design is subjectively and objectively tested shown in Fig. 22 and Fig 23, respectively.



(a) (b)

Fig. 22. Software based the CTSIM vs. Hardware implementation (9/12 bit): a) Original image b) Reconstructed.



(a) (b)

Fig. 23. Software based the CTSIM vs. Hardware implementation (9/12 bit): a) Image quality b) Error measurement.

This analysis illustrates that the image quality is not affected by this reduction. It is obvious that the reconstruction quality in the subjective test and the objective test are very close. At the same time, the utilized hardware area is significantly reduced. The device utilization summary is shown in Table III.

TABLE III
DEVICE UTILIZATION SUMMARY OF 9/12 BUS-WIDTH DESIGN.

Logic Utilization	Used	Available	Utilization
Number of Slice Flip Flops	3,102	47,744	6%
Number of 4 input LUTs	7,163	47,744	15%
Number of occupied Slices	4,716	23,872	19%
Total Number of 4 input LUTs	7,401	47,744	15%
Number of bonded IOBs	156	469	33%
Number of RAMB16BWERs	11	126	8%
Number of DSP48As	0	126	0%

3. *The Exploitation of supplied benchmarks*: Another optimization issue is the optimum usage of the supplied benchmarks like DSP and RAM blocks. When implementing the FFT core, the complex multipliers and the butterfly arithmetic of the core can be implemented using DSP blocks [26] and the buffering memories can be implemented using the RAM blocks. Also, other multipliers and memories in the design can be implemented using these benchmarks that provides the advantage of saving more area, shown in Fig. 24, and maximize the utilization of the already supplied benchmarks without affecting the overall speed or image quality. The device utilization is shown in Table IV.

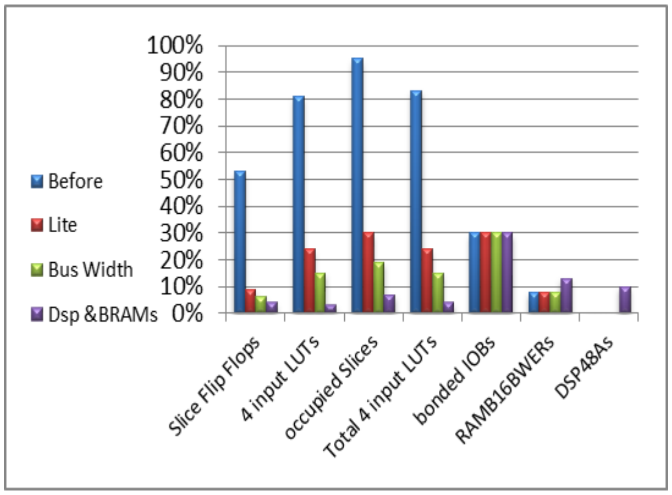


Fig. 24. Different fields of optimization to reduce the hardware utilization.

TABLE IV
DEVICE UTILIZATION SUMMARY OF THE ALREADY SUPPLIED BENCHMARKS.

Logic Utilization	Used	Available	Utilization
Number of Slice Flip Flops	1,910	47,744	4%
Number of 4 input LUTs	1,738	47,744	3%
Number of occupied Slices	1,814	23,872	7%
Total Number of 4 input LUTs	1,914	47,744	4%
Number of bonded IOBs	156	469	33%
Number of DSP48As	17	126	13%
Number of RAMB16BWERs	13	126	10%

V. PARALLEL HARDWARE IMPLEMENTATION OF FBP

FPGA devices provide fine-grained parallelism. Therefore, the time reduction from parallel processing of the FBP algorithm can be achieved. Due to the small hardware utilization achieved, the FBP algorithm core can be redesigned to achieve the desired parallelism and fit to the same chip

(Spartan 3A-DSP 3400). It can be implemented in two schemes of parallelism; *pixel parallelism*, and *projection parallelism*.

In the *pixel parallelism*, the image is divided into multiple disjoint segments. Each segment will be reconstructed through BP block, image accumulation controller and image accumulation RAM. Each BP block will generate its own sub-matrix of T_θ corresponding to its segment. These segments are arranged together to form the resultant image. Due to the small hardware overhead of the BP block, the total utilized area will not be greatly affected, while the BP time will be reduced by a factor equal to the number of the segments. The limitations of this parallelism are the increased number of input/output pins and the minimum time of BP which should be more than the filtration time based on the lite FFT architecture.

In the *projection parallelism*, the BP images from different projections are independent so projections can be divided into multiple groups. Each group of projections is processed separately while all groups are simultaneously processed through multiple FBP blocks. The reconstructed images from these blocks are added together to form the final reconstructed image. The total reconstruction time will be reduced by a factor equal to the number of groups. This parallelism is limited by the size of the chip as the design is almost repeated with each projection group, while the problems of limited input/output pins and minimum BP time do not exist.

From the above discussion, it is noticed that the pixel parallelism is more efficient in the hardware utilization than the projection parallelism as long as the design is away from its limitations. Therefore, the pixel parallelism is first designed to get the maximum available parallelism. After that the projection parallelism is designed. In this approach, the limitations of two schemes are avoided and their advantages are exploited. The 512-by-512 image is divided into eight disjoint segments with size of 256-by-128 pixels. The first and last value of the eight X and Y counters generating matrix T_θ can be easily achieved. In Fig. 25, eight BP blocks, image sum controllers and image sum memory are used in parallel to produce eight reconstructed image segments. These segments are arranged to form the final reconstructed image. The device utilization summary of this parallelism is shown in Table V.

TABLE V
DEVICE UTILIZATION OF 8 SEGMENTS OF PIXEL PARALLELISM ON XC3SD3400A.

Logic Utilization	Used	Available	Utilization
Total Number Slice Registers	2,351	47,744	4%
Number of 4 input LUTs	2,571	47,744	5%
Number of occupied Slices	2,398	23,872	10%
Total Number of 4 input LUTs	2,871	47,744	6%
Number of bonded IOBs	447	469	95%
Number of BUFGMUXs	3	24	12%
Number of DSP48As	40	126	31%
Number of RAMB16BWERs	27	126	21%

Although there is enough area to implement more than eight segments, the limited number of input/output pins of the chip Spartan 3A-DSP 3400 and the projection filtration time (492.86 μs) limit the pixel parallelism to only eight BP blocks. This pixel parallelism reduces the BP time to be 655.36 μs (256 x 128 clock cycles) instead of 5.24 ms (512^2 clock cycles). Finally, the BP time based on the pixel parallelism is reduced by the factor of 8.

The parallelism will be continued by the projection parallelism. Input projections (1024 projections) are divided into five groups; each group contains 205 projections. Five groups are separately and simultaneously processed through five FBP blocks. Each FBP block is 8-segments of the pixel parallelism. The block diagram of this scheme is shown in Fig. 26. The total reconstruction time of the projection parallelism has the time of the BP (t_{BP}) multiplied by the number of projections (N_{proj}) plus the time of the first projection filtration (t_{filt}). Therefore, the presented projection parallelism in this section reduces the total reconstruction time to be 205 t_{BP} + t_{filt} instead of the 1024 t_{BP} + t_{filt}. It is reduced by factor of 5 in the projection parallelism beside to the reduction of t_{BP} by factor of 8 in the pixel parallelism. The total reconstruction time (t_R) is calculated according to equation (14).

$$t_R = (\frac{t_{BP}}{8} \times \frac{N_{proj}}{5}) + t_{filt} \qquad (14)$$

By substitution in equation (14) with the values of t_{BP}, N_{proj}, and t_{filt}, it implies the following calculations.

$$t_R = ((5243 /8) \text{ x } (1024/5)) + 492.94$$
$$= (655.36 * 205) + 492.94 = 134.8 \text{ ms.}$$

The t_R is reduced by the factor of 40. Before the grouping of input projections, the 8-segment pixel parallelism is optimized to restrict the hardware utilization into less than 20% of the chip to permit the implementation of 5-group projection parallelism. The device utilization summary is shown in Table VI.

TABLE VI
DEVICE UTILIZATION OF 8-SEGMENT PIXEL PARALLELISM AND 5-GROUP PROJECTION PARALLELISM.

Logic Utilization	Used	Available	Utilization
Total Number Slice Registers	11,801	47,744	24%
Number of 4 input LUTs	16,791	47,744	35%
Number of occupied Slices	13,217	23,872	55%
Total Number of 4 input LUTs	18,397	47,744	38%
Number of bonded IOBs	469	469	100%
Number of BUFGMUXs	3	24	12%
Number of DSP48As	120	126	95%
Number of RAMB16BWERs	126	126	100%

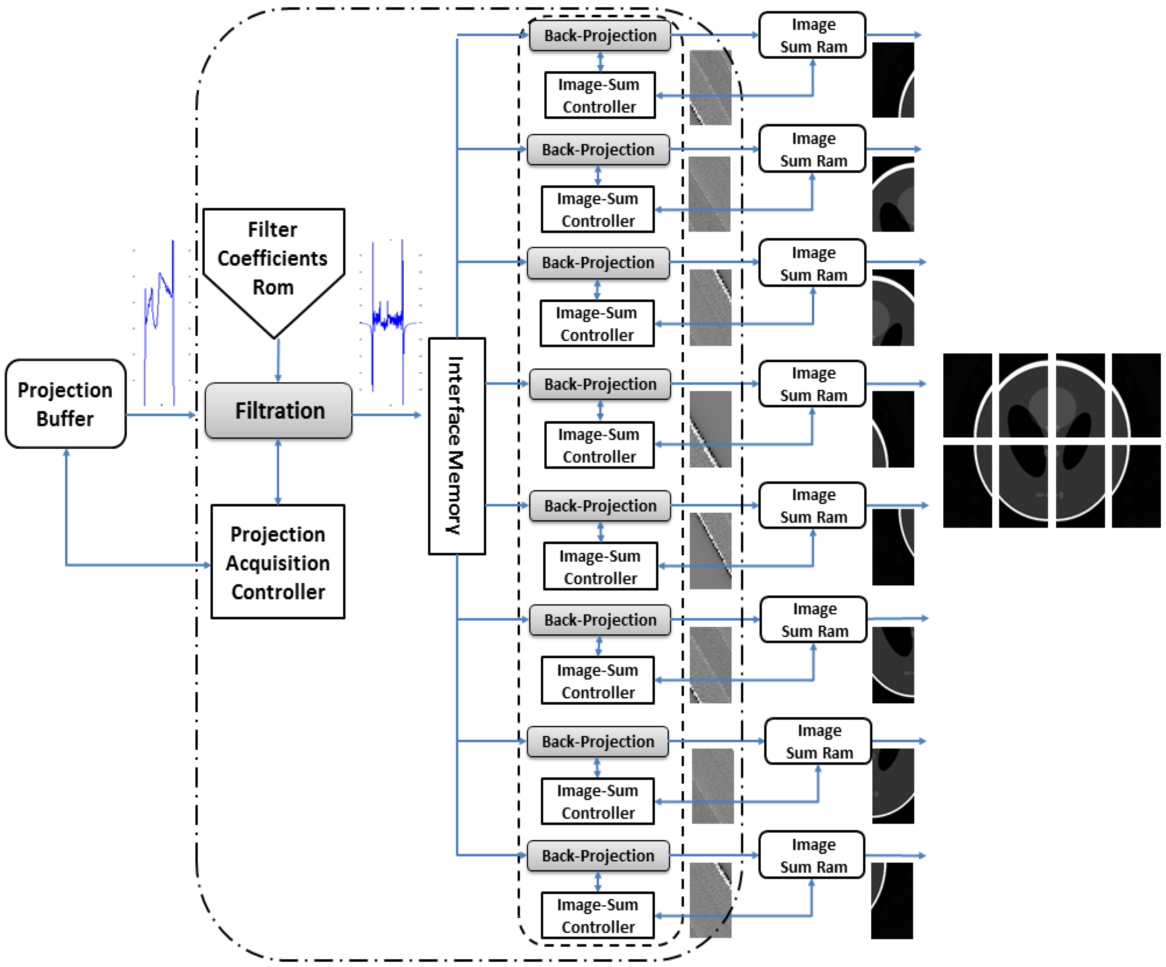


Fig. 25. Block diagram of the FBP using the pixel parallelism.

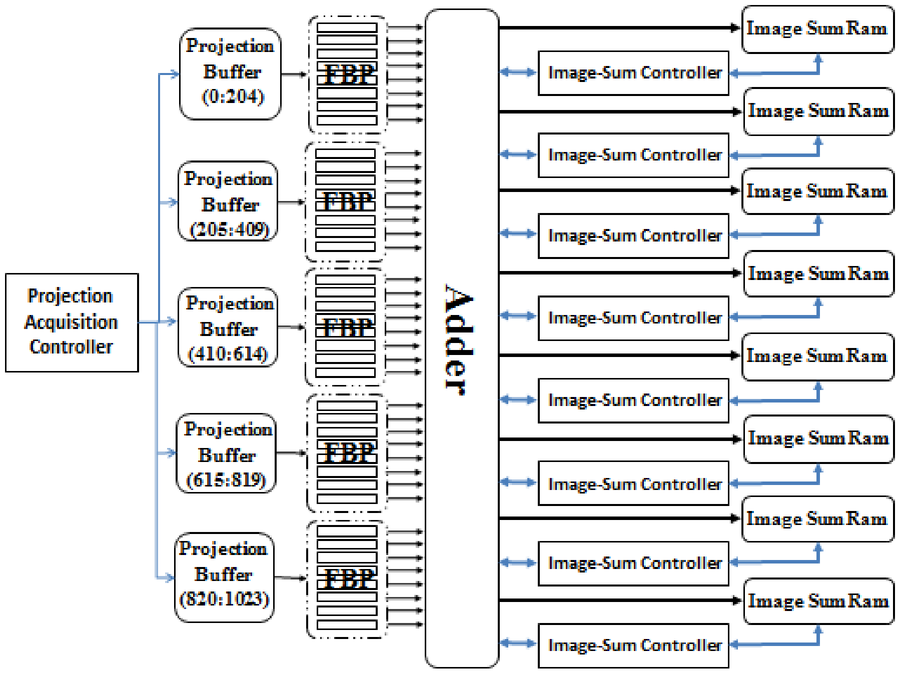

Projection Acquisition Controller
Projection Buffer (0:204)
Projection Buffer (205:409)
Projection Buffer (410:614)
Projection Buffer (615:819)
Projection Buffer (820:1023)
FBP
Adder
Image Sum Ram
Image-Sum Controller

Fig. 26. Block diagram of the combined parallelism.

The authors in [11] implemented the parallel-beam FBP algorithm based on projection parallelism. They achieved image reconstruction of a 512-by-512 pixel image from 1024 projections in 250 ms using expensive Virtex FPGA chip with

65MHz. The required number of clock cycles (N_{CLK}) in the design presented in [11] is 16.25 x 10^6 clock cycles ($N_{CLK} = t_R/t_{CLK}$, where t_{CLK} is the period of the chip clock). In this paper, the combination of the pixel parallelism and projection parallelism is utilized using low-cost Spartan 3A-DSP 3400 chip with 50MHz clock cycles. It achieves image reconstruction of a 512-by-512 pixel image from 1024 projections in 134.8 ms. The N_{CLK} in the design presented in this paper is 6.47 x 10^6 clock cycles. Therefore, it achieves the reduction of the required number of clock cycles to form an image from projections by 60 % comparing to the state of the art in [11]. The presented FPGA hardware implementation of the parallel-beam FBP algorithm, based on the pixel parallelism and projection parallelism, is achieved the superiority in terms of the speed and the hardware cost with respect to all previous published works.

VI. Conclusion

In this paper, the FPGA hardware implementation of the parallel-beam FBP algorithm, based on the pixel parallelism and projection parallelism, was presented. First, the non-parallel hardware was implemented in three main blocks; filtration, BP and dual-port memory. The filtration block receives one by one of the 1024 projections and exports the filtered projection to the dual-port memory from which the BP block read it. The images are externally accumulated to reconstruct the final a 512-by-512 pixel image. The BP and the filtration processes are overlapped to shrink the total reconstruction time to almost the BP time of all projections only.

In addition, the optimization between the speed and the hardware utilization without affecting the image quality was presented. The reduction of the hardware utilization of the implemented algorithm on the target chip can be achieved through the hardware reduction of the FFT IP core, the efficient utilization of the DSP and RAMs blocks of the target chip, and the reduction of the data-bus width. By suitable rounding at each stage, the bus-width is minimized from 16 to 9/12 bit that significantly reduced the hardware utilization.

The reduction of the reconstruction time can be achieved through the pixel and projection parallelism. Eight segments of the pixel parallelism are implemented that achieved reconstruction time reduction by factor of 8. Although there is enough area to implement more than eight segments, the limited number of input/output pins of the target chip and the projection filtration time limits the parallelism to only eight BP blocks. In addition, the parallelism is extended by extra five groups of the projection parallelism which in turn reduce the reconstruction time by factor of 5. The combination of the two schemes of parallelism reduces the reconstruction time by factor of 40. It achieves image reconstruction of a 512-by-512 pixel image from 1024 projections in 134.8 ms using low-cost Spartan 3A-DSP 3400 chip with 50MHz clock cycles. The number of clock cycles in the presented design to form a 512-by-512 pixel image from 1024 projections is 6.47 x 10^6 clock cycles. It reduces the number of clock cycles by 60 % comparing to the state of the art in [11].

Before hardware implementation, a software simulator CTSIM is built to analyze and get accurate validation of each reconstruction process, used as a reference to validate each stage in the hardware design. The image generated from hardware implementation is imported to the CTSIM for the comparison with the image generated from the software. This comparison is tested either in subjective manner or in objective manner. In addition, the CTSIM assists the hardware implementation to be optimized and minimized the hardware utilization.

The presented design can be easily expanded to accommodate more projections with higher densities and to produce larger images. Due to re-configurability of the FPGA, the design can be easily edited to accommodate the fan-beam FBP. That can be done in two approaches; the first is to reformat the fan-beam projections into a set of parallel-beam projections. This reformation is done using an extra module called the rebinning module; the other approach is the use of the modified back-projection algorithm. Where some steps of the FBP is modified to reconstruct the image from the fan-beam projections.

References

[1] G. T. Herman, *Image Reconstruction from Projections: Fundamentals of Computerized Tomography*, Academic Press, 1980.

[2] Kak, A.C., and Slaney, M, *Principles of Computerized Tomographic Imaging*, IEEE Press, 1988.

[3] D. U. Meyer-Baese, *Digital Signal Processing with Field Programmable Gate Arrays*, 3th Edition, Florida: Springer, 2007.

[4] J. Xu, N. Subramanian, A. Alessio, and S. Hauck, "Impulse C vs. VHDL for Accelerating Tomographic Reconstruction," *18th IEEE Annual International Symposium on Field-Programmable Custom Computing Machines*, pp. 171-174, 2-4 May 2010.

[5] 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, pp. 11-24, 1990.

[6] Wu. M. A., “ASIC Applications in Computed Tomography Systems,” *Proceedings of 4th Annual IEEE International ASIC Conference and Exhibit*, Rochester, NY, pp. P1-3/1-4, USA, 1991.

[7] Agi, I., Hurst, P.J., and Current, K.W., “An image processing IC for back-projection and spatial histogramming in a pipelined array,” *IEEE Journal of Solid-state Circuits*, vol. 28, no. 3, pp. 210-221, March 1993.

[8] Stephen G. Azevedo, Brian K. Cabral and J. Foran, “Tomographic image reconstruction and rendering with texture-mapping hardware,” *SPIE Proceedings of Mathematical Methods in Medical Imaging III*, vol. 2299, pp. 280-289, July 1994.

[9] Chen, Chung-Ming, Cho, Zang-Hee, and Wang, Cheng-Yi, “A Fast Implementation of the Incremental Back-projection Algorithms for Parallel Beam Geometries,” *IEEE Transactions on Nuclear Science*, vol. 43, no. 6, pp. 3328-3334, Dec. 1996.

[10] Bins, J., Draper, B., Bohm, W., and Najjar, W., “Precision vs. Error in JPEG Compression,” *SPIE proceeding of Parallel and Distributed Methods for Image Processing III*, Denver CO, pp.76-87, Oct. 1999.

[11] M. Leeser, S. Coric, E. Miller, H. Yu, and M. Trepanier, "*Parallel-Beam Back-projection: an FPGA Implementation Optimized for Medical Imaging*," *Journal of VLSI signal processing*, vol. 39, no. 3, pp. 295-311, March 2005.

[12] Johann Radon, "Über die Bestimmung von Funktionen durch ihre Integralwerte längs gewisser Mannigfaltigkeiten", Berichte über die Verhandlungen der Königlich-Sächsischen Akademie der Wissenschaften zu Leipzig, Mathematisch-Physische Klasse [Reports on the proceedings of the Royal Saxonian Academy of Sciences at Leipzig, mathematical and physical section], Leipzig: Teubner, vol. 69, pp. 262–277, 1917; Translation: Radon, J.; Parks, P.C. (translator) (1986), "On the

determination of functions from their integral values along certain manifolds", *IEEE Transactions on Medical Imaging*, vol. 5, no. 4, pp. 170–176, 1986.

[13] N. Hounsfield, "Computerized transverse axial scanning (tomography)," *Description of system, British Journal of Radiology*, vol. 46, pp. 1016-1022, 1973.

[14] A. M. Cormack, "Reconstruction of densities from their projections, with applications in radiological physics," *Physics in Medicine and Biology*, vol. 18, no. 2, pp. 195-207, March 1973.

[15] Gabor T. Herman, *Fundamental of Computerized Tomography – Image reconstruction from projections*, Second edition, Springer, 2009.

[16] Edwin L. Dove, *Notes on Computerized Tomography*, Univ. of Iowa, 2001.

[17] V. Thakur, A. K. Verma, P. Jena and G. S. Prasad, "Design and implementation of FPGA based Digital Pulse Compression via fast convolution using FFT-OS method," *International Conference on Microwave, Optical and Communication Engineering (ICMOCE)*, Bhubaneswar, pp. 455-458, 2015.

[18] Thingom and P. Khundrakpam, "FPGA implementation of FIR filter using RADIX-2r," *International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, Chennai, pp. 1524-1528, 2016.

[19] J. C. Russ, *The Image Processing Handbook*, 5th edition, North Carolina, CRC Press, 2007.

[20] R. W. S. a. L. R. Rabiner, "A digital signal processing approach to interpolation," *Proceedings of IEEE*, vol. 61, no. 6, pp. 692-702, June 1973.

[21] J. Hsieh, *Computed tomography: principles, design, artifacts, and recent advances*, Washington USA, SPIE and John Wiley & Sons, Inc., 2009.

[22] Bassam Abd-Elwahab, "*CT Image Reconstruction using FPGA*", M.Sc. thesis, Military Technical College, Cairo, Egypt, 2012.

[23] M. Kevin M. Rosenberg, *CT Sim 3.5 User Manual*, New Mexico, 2002.

[24] L.A. Shepp, B.F. Logan, "The Fourier Reconstruction of a Head Section," *IEEE Transactions on Nuclear Science*, vol. 21, no. 3, pp. 21-43, June 1974.

[25] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli, "Image quality assessment: From error measurement to structural similarity," IEEE Transactions on Image Processing, vol. 13, no. 4, April 2004.

[26] LogiCORE IP, FFT v7.1, DS260 Product Specification. [Performance]. Xilinx, April 19, 2010.