COBEM 2023 27TH INTERNATIONAL CONGRESS OF MECHANICAL ENGENEERING

ABCM Brazilian Society of Mechanical Sciences and Engineering
27th ABCM International Congress of Mechanical Engineering
December 04-08, 2023, Florianópolis-SC, Brazil

COB-2023-1502
NUCLEAR MAGNETIC RESONANCE IMAGE RECONSTRUCTION USING RECONFIGURABLE SIMULATED SYSTEM

Amauri da Costa Júnior
Gerardo Antonio Idrobo Pizo
University of Brasilia - Campus Gama - St. Leste Projeção A - Gama Leste, Brasília - DF, 72444-240
amauri_cj@hotmail.com / gerardo.idrobo@gmail.com

Abstract. *Magnetic resonance imaging has been one of the most researched ways of obtaining medical images in recent years due to its range of possibilities in imaging, more efficient reconstructions appear daily and this work will address hardware reconstruction. The reconstruction of resonance images must be done quickly with quality and precision, this process is usually performed by computers, however, the use of algorithms performed on hardware can be efficient because it has high speed and accuracy over time. Simulations of the hardware reconstruction process in VHDL were performed using different images to test the algorithm, the results were then compared with the original images and with the expected patterns for resonance exams. The algorithm proved to be efficient, producing images with high quality, quickly and with high precision in time, performing the reconstruction of different images without fluctuations in the reconstruction time. The work is able to show the advantages of hardware reconstruction, obtaining at least 60% similarity between the reconstructed image and the original and a formula that allows to calculate the processing time of the algorithm according to the clock frequency and the dimensions of the image, thus being able to reconstruct even 4000 images per second.*

Keywords: *MRI, FPGA, VHDL,Reconstruction, Reconfigurable, Medical Imaging*

1. INTRODUCTION

Magnetic resonance images are obtained through a series of procedures that begin with applying a series of pulses of magnetic fields to the human body. The body will react to these magnetic fields producing a magnetic field of its own, this new field is then measured by local coils.

The set of data obtained by these coils are stored in a matrix called k-space, the horizontal coordinates of the matrix correspond to frequency variations in the applied signal, and the vertical coordinate corresponds to the phase changes of this signal.

To transform the k-space into an image that represents a section of the patient's body, an inverse Fourier transform of two dimensions is performed on the matrix and then the sum of the absolutes of the real and imaginary parts.

This sequence of steps has been optimized since the invention of magnetic resonance imaging, from changes in the pulse sequence, reduction in the size of the k-space, and new ways of performing the reconstruction, all with the objective of reducing the examination time without sacrifice the final image quality.

The objective of the project was to show the feasibility of performing the reconstruction of magnetic resonance images in reconfigurable systems, such as FPGAs. Implementations of this type are performed in hardware and have the advantage of being extremely accurate in time, that is, they perform operations of the same size in the same time regardless of the type of input, this characteristic can be advantageous for applications in real time, where several inverse transforms are performed to form a video, and an operation cannot be timed differently from the other so as not to harm the final result.

To verify whether MRI reconstruction would be feasible if performed in hardware, a simulation of the processing carried out by an FPGA was carried out, using the ISE 14.7 design tool and the choice of model was the XC6SLX45 board from Spartan 6 family. Image reconstruction was performed using LogiCORE from Xilinx, which allow the creation of blocks of memory and Fourier transforms with greater ease.

The images used for the tests were images of resonance exams that underwent double Fourier transforms using MATLAB to form the k-space, then they are recorded in two .coe documents, real part and imaginary part, which are read by the algorithm in VHDL. The k-spaces were created this way so that we can test with a wider variety of MRI images and can refine the algorithm without relying on MRI scan datasets.

2. PROCESSING

The image reconstruction steps can be seen below. These steps will be referenced throughout this document.

1. The image stored in two ROM memories (real part and imaginary part) are loaded into the core that performs the

first transform;

2. The first inverse transform (IFFT) is performed;
3. The result of the first IFFT is sent to the RAM memory and the transposition of this matrix is carried out, inverting the address values from line to column, the result is loaded into the core that will carry out the second transform;
4. The second inverse transform (IFFT) is performed;
5. The result of the second IFFT is sent to the RAM memory and the transposition of this matrix is carried out inverting the values of the addresses from row by column;
6. The sum of the absolute of the real part and the absolute of the imaginary part is performed;
7. The reconstruction process is completed.

The reconstructed image is saved in a text file containing 16384 values, one for each point of a 128x128 image and the visualization of this image was done through MATLAB, the codes made in MATLAB do not perform any reconstruction or operations on the image, they are only used to prepare the data for reconstruction and to view this data after the process is complete. A flowchart with image processing can be seen in Fig. 1.

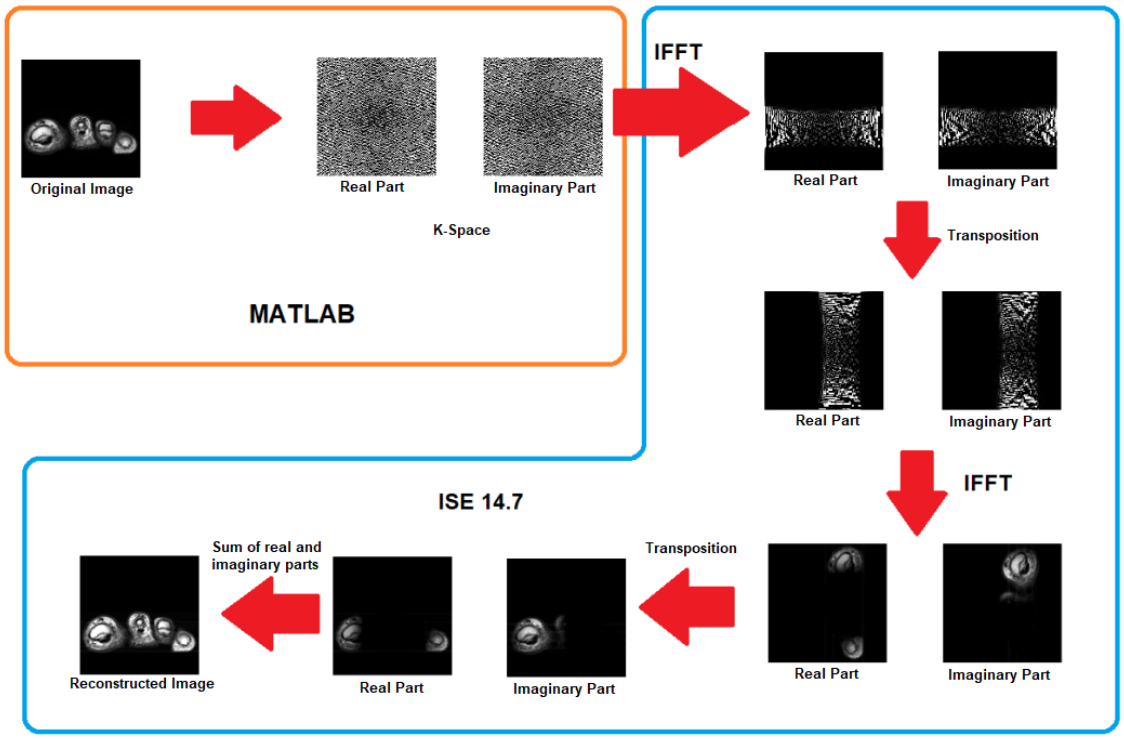

Figure 1. Image processing flow

The result was compared with the original image used to create the test k-space to reach a conclusion whether the reconstruction process was successful or not.

3. ALGORITHM

During the development of the TestBench, some LogiCORES were used, computational tools created by Xilinx to assist in performing tasks in FPGAs, their applications and explanations can be seen in the following subsections.

3.1 FFTs (Fast Fourier Transforms)

To perform the fast transforms, LogiCORE IP Fast Fourier Transform v7.1 from Xilinx was used, which implements the Cooley-Tukey method of fast Fourier transform. The type of implementation chosen was Pipelined, as it processes the data constantly by executing Radix-2 Butterfly as the data is received by the core.

The transform is calculated as the data is read, this allows for both faster processing and constant processing in case of images being acquired in real time. The kernel accepts 32-bit fixed-point inputs and outputs them in natural order. One

of the advantages of the core is to control the constants that are multiplied within each sum performed by the FFT in each of its stages (scale_sch), thus being able to avoid errors when performing two transforms in sequence.

The transform performed is of size 128, so each line of inputs is transformed at a time and the result is available in natural order, costing a little more processing time but being necessary to visualize the results. The core performs the transform at a maximum frequency of 395 MHz and for the simulations a frequency of 100 MHz was used. The core was configured to use convergent rounding to avoid any errors that truncated rounding could generate. A variable was included that restarts the core at the end of the transform (sclr), in order to make it ready for a new sequence of data (LogiCORE (2011)).

By only doing transforms in one direction, due to the data entering serially, the transform is performed vertically. In order to perform the transform in both directions, the images after the first transform undergo a transposition and then undergo the second transform, in this way the operation is performed on all rows and columns as shown in Fig. 2.

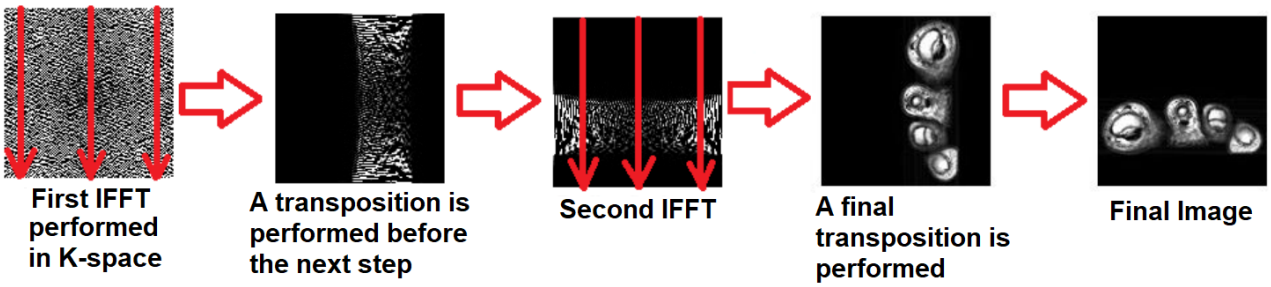

Figure 2. Transpositions are performed to allow the IFFT to be performed both vertically and horizontally.

Two different cores were used because for images in which the length is different from the width, the same core could not be reused to perform the two transforms, as the length of the transform is fixed and cannot be changed during the execution of the algorithm. In Fig. 3 it is possible to observe the FFT core configuration screen.

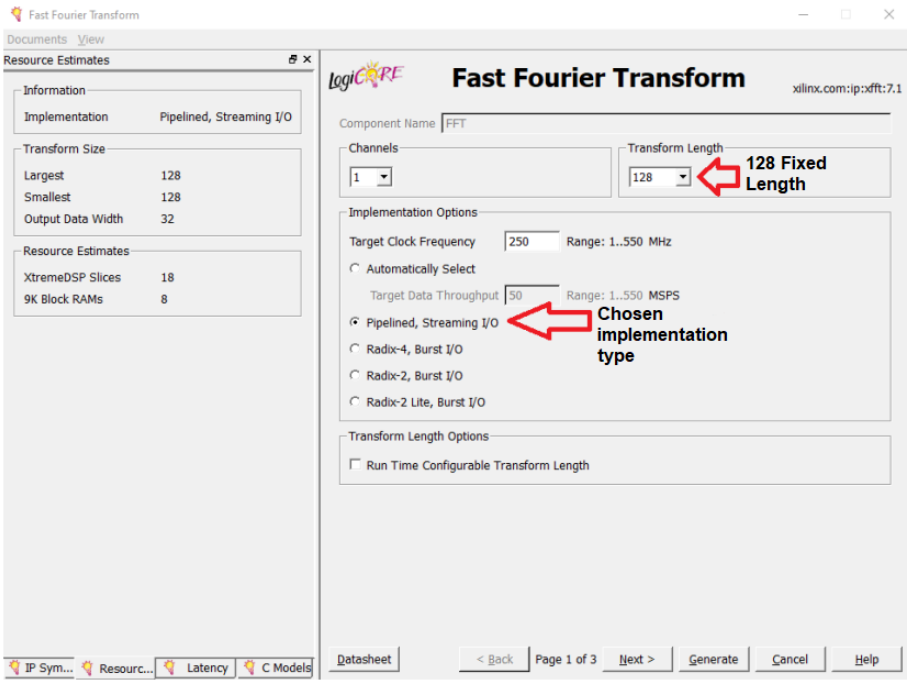

Figure 3. FFT LogiCore Configuration Screen.

3.2 ROM memory

The ROM memories were generated using the LogiCORE IP Block Memory Generator v7.3 from Xilinx, using the single port mode because only one data reading output would be used. The Read Width must be configured to be the size of the number of bits of each pixel in the stored image, in this case 32 bits, and the Read Depth adjusted to the number of pixels in the image, for a 128x128 image there are 16,384 (Logicore (2012)).

Two ROM memories are used to store the image, one for the real part and another for the imaginary part, the images

are stored in .coe format to be read by the cores, within each of the files each pixel of the image is represented by a decimal value, so each file has 16,384 values representing each part of the image. Each of these values is transformed into a binary variable of 32 bits, the first bit being the sign bit, thus the maximum amplitude of the bits is 2,147,483,647.

Each value in this document represents a shade of gray, where lower values are darker shades and higher values are lighter shades, as shown in Fig. 1 k-space area.

3.3 RAM memory

Like the ROM memories, the RAMs were generated using the LogiCORE IP Block Memory Generator v7.3 from Xilinx, the settings are the same as the ROM memory, single port with Read Width of 32 bits and Read Depth of 16,384 in No Change mode, which does not allow the output to change while the memory is being written, to prevent any transposition errors from occurring (Logicore (2012)).

The RAM memories are used to temporarily store the values of the matrix during the transpositions, it accepts up to 16,384 values of 32 bits, after each transform the result is stored in the RAM memories and each pixel of the image has an address composed of a pair of values, thus, to carry out the transposition, just invert the values of this pair. In the last stage of the reconstruction, the absolute sum of the real and imaginary parts of the values coming from the last transposition, carried out in the RAM memories, is also carried out.

4. RESULTS

Several reconstructions of images were made, the final result obtained which contains the image of the fingertips from a personal MRI exam can be seen in Fig. 4, in this figure we can observe the absolute of the real part, the absolute of the imaginary part, and the reconstructed image with the sum of real and imaginary parts next to the original image for comparison.

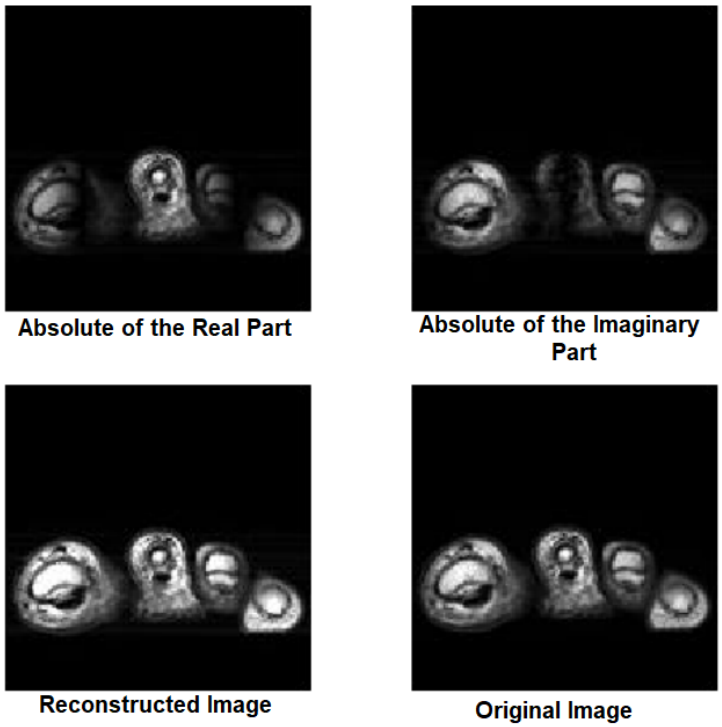

Figure 4. Images generated by the algorithm compared to the original image.

4.1 Image Quality

To check the quality of the image reconstructed by the algorithm, the difference between the original image and the reconstructed one with the VHDL code was analyzed, using the MATLAB function ssim(A, ref), which shows the structural similarity index between a test image (A) and a reference image (ref). With this function it is also possible to obtain an image that highlights the difference between the images, which can be seen in Fig. 5, the light parts of the image consist of parts that the reconstructed image is more similar to the original image, and the darker parts show the places where they are different. The figure was placed inside a blue background for better visualization of the white part (MATLAB (2014)).

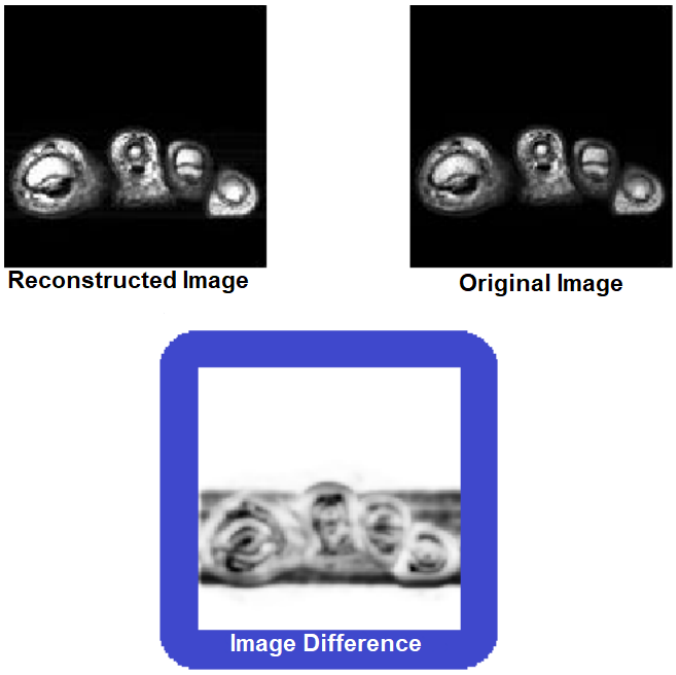

Figure 5. Comparison between reconstructed image and original image.

The coefficient of structural similarity obtained through the function was 85.658%. For other tested images, this value was greater than 60%, showing a high similarity between the original images and those reconstructed by the algorithm. Other reconstructions can be seen in Fig. 6, the algorithm was tested with medical and non-medical images to verify its functioning in different images.

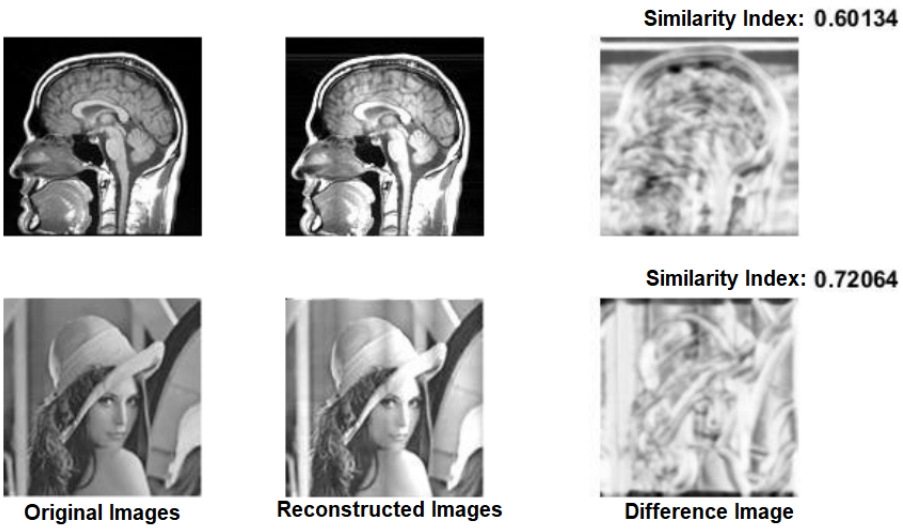

Figure 6. Reconstruction performed on different images (Gonzales and Wintz (1987)).

4.2 Speed and Accuracy in Time

In all tests, the time taken to complete the reconstruction was observed. In order to obtain a more precise measure, a variable was inserted that counted the number of clock cycles, for a 128x128 image 49,900 clock cycles were necessary, this value remained the same regardless of the image being reconstructed, showing accuracy in time.

For a working frequency of 200 MHz, around half the maximum frequency that the FFT block supports, a clock cycle would take 5 ns to be performed, so 49,900 clock cycles would take 0.2495 ms, a frequency of 4008.016 images per second.

As can be seen in Fig. 7, the reconstruction process can be divided into three approximately equal parts, each of these

stages has 16,384 clock cycles, one for each image value of 128x128, totaling 49,152 clock cycles. There are 374 clock cycles from the start of data reading in the FFT core to the start of delivering the results, as there are two FFT cores, totaling 49,900 clock cycles for the entire process.

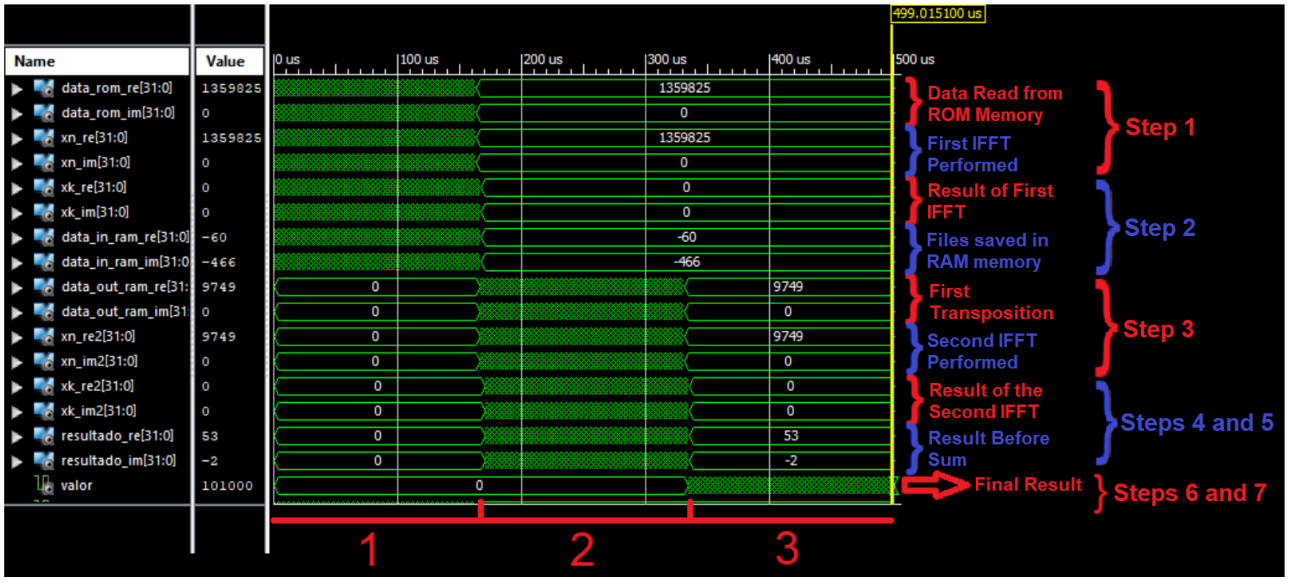

Figure 7. Simulation viewed in ISE 14.7 with data captions.

In this way it is possible to estimate a time for calculations of reconstructions of larger images, the number of clock cycles for the beginning of the results varies according to the size of the transform, however for transforms of size 128 or larger they correspond to less than 1% of reconstruction time. Therefore, it is possible to obtain an equation that approximately calculates the number of clocks.

$$Nclk \simeq 3(C \cdot L), \tag{1}$$

Equation (1) shows how to calculate the approximate number of clock cycles (Nclk), where C is the length of the image in pixels and L the width of the image in pixels, using the number of clock cycles and the processing frequency (F) is possible to calculate the number of images per second (Tx) that the algorithm can deliver according to the image size,

$$Tx \simeq \frac{F}{Nclk}, \tag{2}$$

substituting the value of Nclk for that of Eq. (1),

$$Tx \simeq \frac{F}{3(C \cdot L)}, \tag{3}$$

In Eq. (2) and Eq. (3) it is possible to calculate this rate of images per second according to the frequency F of the system, in Hertz, and the length and width of the image, in pixels. Thus being able to scale the system according to its inputs allowing to design the system so that it fulfills certain requirements of execution speed and image quality, for example increasing the image size and sacrificing part of the reconstruction speed.

5. CONCLUSION

MRI scans are constantly being improved and refined. Different modes of image acquisition, more efficient reconstructions and different ways of looking at the data are developed every year. Hardware image reconstruction is one of the possible steps that will be taken in the coming years, as the use of reprogrammable hardware is growing in several areas of the industry.

The results obtained in the project, both in image quality and in processing speed and precision, were able to compare to the results obtained in processing performed on computers by GPUs. The images reconstructed by the VHDL code have enough sharpness to observe details and are at least 60% identical to the images reconstructed by computers as shown in Fig. 7. The speed of image reconstruction by the algorithm would not be a bottleneck for the machines, Real-time exams are performed at up to 18 images per second (Uecker *et al.* (2010)), the algorithm achieved rates higher than this, and the image size can be increased to gain quality, sacrificing processing time by a large margin. Finally, variations in the number of clocks were not observed in any of the simulations, images of the same size were reconstructed in the same amount of clock cycles regardless of the image complexity, showing accuracy in time, important feature for real-time image reconstruction.

6. REFERENCES

Gonzales, R.C. and Wintz, P., 1987. *Digital image processing*. Addison-Wesley Longman Publishing Co., Inc.

LogiCORE, I., 2011. "Fast fourier transform v7. 1". *Xilinx Application Note*.

Logicore, I., 2012. "Block memory generator v7. 3".

MATLAB, 2014. "Structural similarity index for measuring image quality". URL `https://www.mathworks.com/help/images/ref/ssim.html`.

Uecker, M., Zhang, S., Voit, D., Karaus, A., Merboldt, K.D. and Frahm, J., 2010. "Real-time mri at a resolution of 20 ms". *NMR in Biomedicine*, Vol. 23, No. 8, pp. 986–994.

7. RESPONSIBILITY NOTICE

The authors is are solely responsible for the printed material included in this paper.