Optimizing MRI Registration using Software/Hardware Co-Design Model on FPGA

Yasmeen Farouk, Sherine Rady

Abstract**: *The correct localization of brain tissue deformation and determination of the tumor growth relies majorly on the accuracy of the process known by image registration. Poor registration may lead to misclassified diseases and highly affect image-guided surgery and radiation therapies. Voxel-based morphometry (VBM) is an image analytical technique encompassing accurate registration but suffers from intensive time computations, similar to most of image registration techniques. Achieving the compromise between accuracy and computations is a challenging mission. Field programmable gate arrays have fast-evolving and customizable hardware acceleration capabilities that promise to help speed up computational tasks. This paper presents a software/hardware co-design model for accelerating the implementation of the diffeomorphic image registration algorithm 'DARTEL' as a part of VBM that analyzes MRI images. An optimized and pipelined hardware architecture is proposed and integrated into the Statistical Parametric Mapping (SPM) software tool that runs the DARTEL. Acceleration of the DARTEL registration algorithm resulted in a speedup factor of 114x on function-level, compared to the CPU with a contribution of 8x faster for the overall performance in the registration process of the SPM. The proposed model is successfully validated for the identification of Alzheimer's disease based on T1-weighted MRI. A proposed software/hardware co-design model for VBM achieves remarkable acceleration while maintaining classification accuracy and proving proficiency against other CPU and GPU implementations.*****

Keywords**: *Alzheimer's disease, Field programmable gate array, Image registration, Magnetic resonance imaging, Software/Hardware co-design, Voxel-based morphometry.*****

I. INTRODUCTION

Registration is considered an essential process in medical imaging analysis. Computerized Tomography (CT) scans used in many clinical routines are normally registered to screen the growth of a tumor over a period of time. Similarly, Magnetic resonance imaging (MRI) scans are registered for localizing the deformation of brain tissue to identify certain diseases such as Alzheimer's disease (AD) or any other form of dementia. Neuroscientists also rely on accurate image registration of functional MRI (fMRI) for motion correction prior to the brain activation analysis. Additional, anatomical images, such as CT, and functional images, such as Positron Emission Tomography (PET), are both used in the radiotherapy treatment planning in a process known as multi-modal registration and fusion.

Image registration is the process of aligning two images; a 'source' image and a 'reference' image. This alignment aims at finding a set of parameters that guarantees the optimal spatial transformation between the points in one image and those in the other. These parameters determine the relative shapes of the images. There are two broad categories of parameterization [1]; (a) Small deformation framework that commonly does not preserve topology and (b) Large deformation framework that generates deformations (diffeomorphisms) and enforces the preservation of topology. Diffeomorphic Anatomical Registration using Exponentiated Lie algebra (DARTEL) is a large-deformation registration framework that outperforms the small deformation setting by generating diffeomorphic deformations which can be inverted simply and measured fast. It has shown high accuracy relative to other non-linear spatial deformation registration algorithms [2]. The diffeomorphic framework in DARTEL registers images by computing a flow field and uses its corresponding exponential values to generate deformations. Its main advantage lies in the effective iterative approach used for computing the first and second derivatives generated for optimization and in solving equations by using a full-multi-grid method. Despite DARTEL's good performance, its intensive iterative behavior is very time-consuming. The large number of parameters involved to generate deformations needs computationally efficient methods for solving the equations. Efficient schemes are needed to handle such extremely large matrices. Many medical image processing software tools have been developed to assist scientists. A survey on their functionalities has been conducted by Lay-Khoon Lee in [3]. Statistical Parametric Mapping (SPM) is one of the most popular tools [4]. It has been designed for the analysis of different brain imaging data sequences such as MRI, fMRI, and PET. DARTEL toolbox is implemented in SPM for registration tasks based on John Ashburner work [1]. Moreover, SPM uses a voxel-based approach [5], Voxel-based Morphometry (VBM), where images are first segmented, then registered, then spatially normalized into a standard space and smoothed before finally applying the statistical analysis. VBM approach is widely used in medical image analysis domain [6-8]. Field-Programmable Gate Arrays (FPGA) and Graphics Processing Units (GPU) are hardware accelerators with intrinsic parallel processing powers. FPGA is a fast-evolving technology.

Revised Manuscript Received on December 05, 2020.

***** Correspondence Author

Yasmeen Farouk*, Department of Information Systems, Faculty of Computer and Information Sciences, Ain Shams University, Cairo, Egypt. Email: yasmin.farouk@cis.asu.edu.eg

Sherine Rady, Department of Information Systems, Faculty of Computer and Information Sciences, Ain Shams University, Cairo, Egypt.. Email: srady@cis.asu.edu.eg

New FPGA models provide an on-chip RAM, embedded Digital Signal Processing (DSP) blocks for realizing fixed and floating-point arithmetic operations, and fast bandwidth interfaces with efficient power consumption. Some models come with an embedded processor core such as ARM. FPGAs provide deterministic timing in the order of nanoseconds [9]. Choosing the best accelerator between FPGA and GPU has been always a difficult and confusing question for system designers. From an architectural perspective, FPGAs are re-configurable silicon chips that use pre-built logic blocks and programmable routing resources, while GPUs are specialized electronic circuit having thousands of small microprocessor cores. Studies have been made comparing their performance and their capabilities [10-13]. The analysis in [9] shows that GPUs are cost-efficient with huge floating-point processing capacity. They have less development effort because many algorithms are designed directly to run on them. On the other side, FPGAs are power efficient accelerators with flexible interfaces. They can connect to any other device via standard or custom interface, contrary to GPUs that can only interface via PCIe. The work recorded in the literature for accelerating medical image analysis has focused only on the use of GPUs, while neglecting the fact that FPGAs still poses attractive design advantages. In this paper, the use and implementation of FPGA are in-depth explored, evaluated, and compared to GPUs while applied to image registration and analysis processes. Specifically, this paper presents a software/hardware co-design model for accelerating the implementation of the diffeomorphic image registration algorithm DARTEL. An optimized FPGA architecture is proposed and integrated into the SPM software tool that holds the DARTEL implementation. For evaluation, the accelerated system is employed in an experimental study for the identification of AD based on T1-weighted features MRI [14]. The rest of the paper is organized as follows: section 2 gives an insight into the related work presented for accelerating the medical image registration problem. Section 3 presents an overview of the DARTEL registration algorithm. Section 4 describes the scientific approach and methods used. Section 5 presents the experimental work and discuses the achieved results. Finally, Section 6 concludes the work presented in the paper and introduces the future work.

II. RELATED WORK

Many successful trials have been issued in the literature to embed FPGA in the medical imaging analysis domain. In classification problems, Vladimir Kasik et al. [15] utilized FPGA in CT and MRI processing for the diagnosis and evaluation of hydrocephalus. The study counts the ratio of intracranial fluid in the skull. Their implementation used Virtex-4 FPGA board. Jahyun J. Koo et al. [16] achieved performance improvement for an FPGA-based PVE algorithm by 5.1x over CPU in an MRI brain tissue classification problem. The work used RASC RC100 FPGA-accelerators and the Mitrion-C HLL. Binay Chandra et al. [17] proposed an FPGA implementation for extracting features of MRI using wavelet transform for segmenting images. Their implementation used Zedboard FPGA.

In the field of image filtering, S. Allin Christ et al. [18] presented an FPGA architecture for image filtering algorithms for tumor identification. Their work was implemented on SPARTAN-3E starter kit and combined Xilinx System Generator (XSG), MATLAB, and Simulink. S. Hasan et al. [19] presented an FPGA prototyping for nine 2-D MRI image filtering algorithms by using XSG. Their FPGA implementation targeted two Virtex-6 FPGA boards and they compared their frequency and power performances. For the prediction of neural activity and brain connectivity, Will X. Li et al. presented a prototype for the generalized Laguerre-Volterra model (GLVM), which is a mathematical abstraction for the description of neural processes [20]. The presented work achieved more than 2K speedup factor using Vitrex-6 FPGA board over an Intel Core i7 CPU. Ludovico Minati et al. [21] analyzed resting-state functional MRI (rs-fMRI) and presented a parallel implementation of Dijkstra’s algorithm on FPGA to overcome intensive computational needs of the high-resolution geodesic mapping of brain connectivity analysis. An acceleration factor between 15 and 18 is recorded. Most of the software tools developed for medical image processing are open source tools. ITK, FSL, Elastix and SPM are examples [22]. Several works used GPU to accelerate these tools. The study presented by Hernández et al. [23] accelerated the estimation of fibre orientations from diffusion-weighted magnetic resonance images (DW-MRI) by the GPU parallelization of the Bedpost tool within the FSL framework. Their work reported a speedup of 85x compared to the sequential version. The work presented in [24] accelerated SPM in conjunction with Resting-State fMRI Data Analysis Toolkit (REST). Yan et al. developed in this work a MATLAB toolbox called Data Processing Assistant for Resting-State fMRI (DPARSF). In [25] an accelerated version of elastix, an image registration package is presented. Denis P. Shamonin et al. combined in this work several parallelization techniques on CPU and GPU extending the OpenCL framework from ITKv4 software package. MRI registration in SPM is accelerated via GPU in [26] and [27]. Teng-Yi Huang et al. in the work [28] re-implemented the dynamic-link library responsible for the image registration in SPM with a CUDA model and achieved a 14x increase in speed using single-modality intra-subject datasets. The image registration algorithm used in this work is based on information theory. It uses collinear affine transformation where DARTEL uses deformable B-spline transformation. The study conducted by P. Valero-Lara in [27] proposes a GPU implementation for DARTEL parallelization that reduces the overhead caused by memory transfers and achieved an overall speedup of 13x for a single GPU implementation. The author explored as well the use of multiple (2 and 4) GPUs. The latter study is considered a relevant work for accelerating MRI registration using GPU. This work is compared to the one presented in this paper with respect to the commonly used DARTEL implementation. In other words, both studies try to accelerate DARTEL using parallelization on hardware chips; P. Valero-Lara uses GPU while this work uses FPGA.

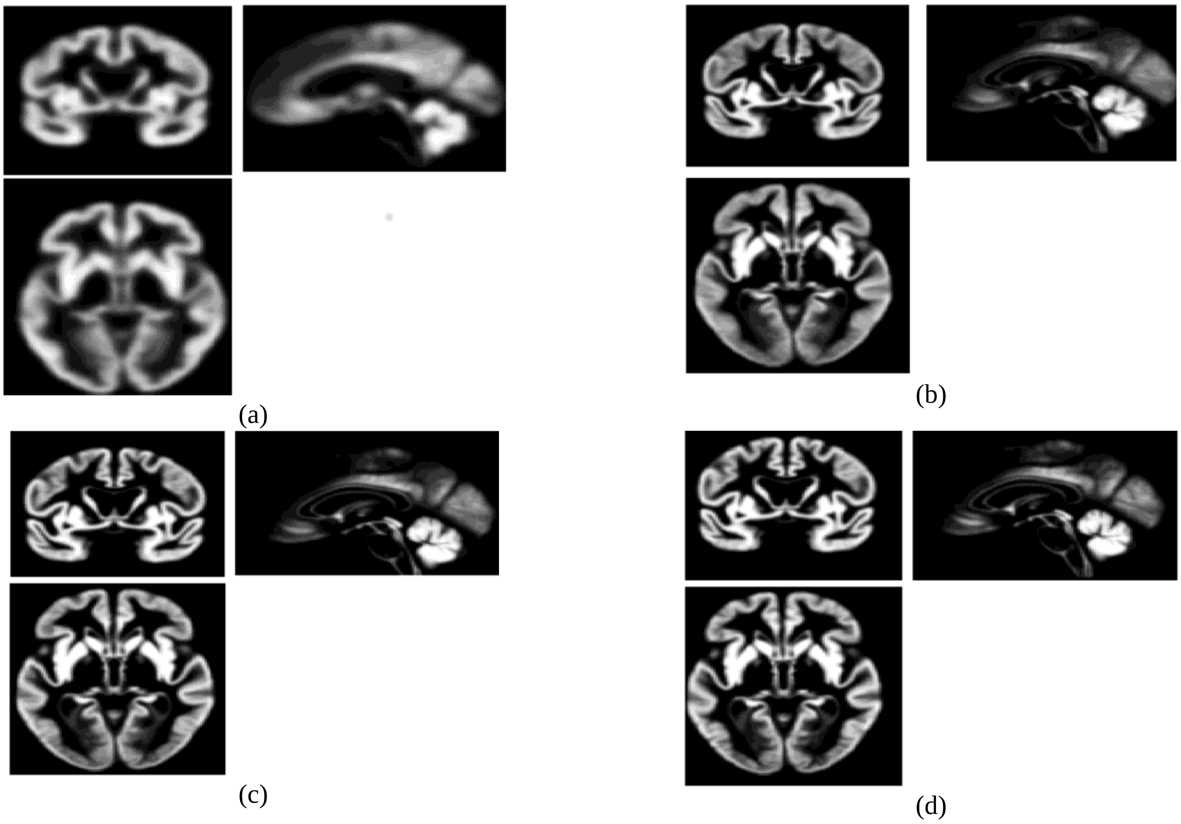
III. DARTEL

DARTEL is an image registration algorithm that is composed of two steps; template generation and warped images generation [1].

In the template generation, all Gray Matter (GM), White Matter (WM) images in the dataset contribute to form an initial average which is called Template Zero. All images are then registered to this Template. The resulted images are used to create a new average image, Template One, to which all images are again registered to. This procedure is repeated several times, typically 6 iterations. The average template generated becomes increasingly crisp as the registration proceeds.

In the warped images generation step, all images are registered to this final average template obtained from the previous template generation step. For each image, a set of deformation matrices is created that describes how the deformation is obtained.

Fig. 1 shows how the DARTEL algorithm works by illustrating the development of average image templates. Intensity averages of GM and WM images are considered an initial template as shown in Fig. 1a. Every brain MRI image is registered with the template image using an inverse-consistent formulation. Iterations follow to map the scans to their average forming a new average. The initial template becomes sharper each time it is re-generated as shown in Fig. 1b and in Fig. 1c until it reaches its final form shown in Fig. 1d. Finally, warped versions of the images can be generated.



(a)

(b)

(c)

(d)

Fig. 1: DARTEL Template Generation: Intensity averages of GM images (coronal, sagittal and axial views) after multiple iterations a) average after initial rigid-body alignment b) average after two iterations c) average after four iterations d) average after six iterations.

DARTEL is considered a diffeomorphic registration algorithm. Assuming a flow field u that preserves constancy over time, Φ is calculated by the differential equation (1) that defines the deformation evolution:

$$\frac{d\Phi}{dt} = u(\Phi^{(t)}) \tag{1}$$

To obtain diffeomorphisms, an identity transform ($\Phi^{(0)} = x$) is first initialized and next integrated over time to produce $\Phi^{(1)}$.

The small deformation mechanism can be seen as an Euler integration that has simple time step of one. Accurate solution is produced by a large number of these simple time steps. Scaling and squaring are used to find the solution given that the squared number of steps of time are used. Equation (2) shows the Euler integration method with eight time steps.

$$\begin{aligned} \Phi^{(1/8)} &= x + u(x)/8 \\ \Phi^{(1/4)} &= \Phi^{(1/8)} \circ \Phi^{(1/8)} \\ \Phi^{(1/2)} &= \Phi^{(1/4)} \circ \Phi^{(1/4)} \\ \Phi^{(1)} &= \Phi^{(1/2)} \circ \Phi^{(1/2)} \end{aligned} \tag{2}$$

The derivatives of the deformations or the Jacobian matrices marks the local changes in the deformation field such as rotating and stretching. Given the composition of two deformations Φ_A and Φ_B, the resultant deformation Φ_C is computed by equation (3):

$$\Phi_C = \Phi_A \circ \Phi_B \tag{3}$$

Finally, the Jacobian field is produced by equation (4):

$$J_{\Phi_C} = (J_{\Phi_B} \circ J_{\Phi_A}) J_{\Phi_A} \tag{4}$$

The composition of the deformations, or Jacobian composition, obtained with the Jacobian matrices are very intensive time consuming processes.

Storing deformation matrices with the high-dimensional nature of this model is very difficult due to memory limitations. DARTEL uses an FMG approach [37] to store those matrices. The FMG method is a recursive approach and was extended by [1] so that it can be applied on more complex three dimensional images.

IV. Proposed MRI Registration Software/Hardware co-design framework

A. Proposed Framework

The proposed FPGA software/hardware co-design framework is illustrated in Fig. 2. VBM analysis is applied on an input MRI dataset to extract significant features. This is used next by a classifier to identify AD patients from normal controls (NC). The core of the AD classification framework is the VBM features analysis. Its registration block is the target for the proposed FPGA acceleration framework.

VBM [29] involves different processing steps. The first is the image segmentation *that is carried out for the identification of different tissue types within the images. Corresponding segmented regions are GM, WM, and Cerebrospinal Fluid (CSF). GM is a major component of the nervous system. It is controlled by the nerve cells while WM is made of axons connecting GM together. VBM measures the volume or shape of GM structures such as the temporal cortex. GM is the primary tissue involved in the framework for a later classification stage. GM and WM are both used in the* registration *which is the second processing step of VBM while CSF is not used. The output of the* segmentation *is used by* registration *for achieving accurate inter-subject alignment.*

The registration algorithm used in VBM is called DARTEL. Its primary goal is to best align images together. This is achieved by continuous alignment of the images and their average as described earlier in the previous section.

VBM contains two additional processing steps: *Normalizing and smoothing* and *statistical analysis*. Images are normalized in the former process in the MNI space and a set of smoothed Jacobian scaled GM images are generated using the shapes encoded by the deformations. The Final process, *statistical analysis*, uses the statistical model GLM to infer data about the concentration of GM. The final output of VBM is a statistical parametric map that shows the regions where GM tissue in MRI differs clearly in MRI images under evaluation.

.

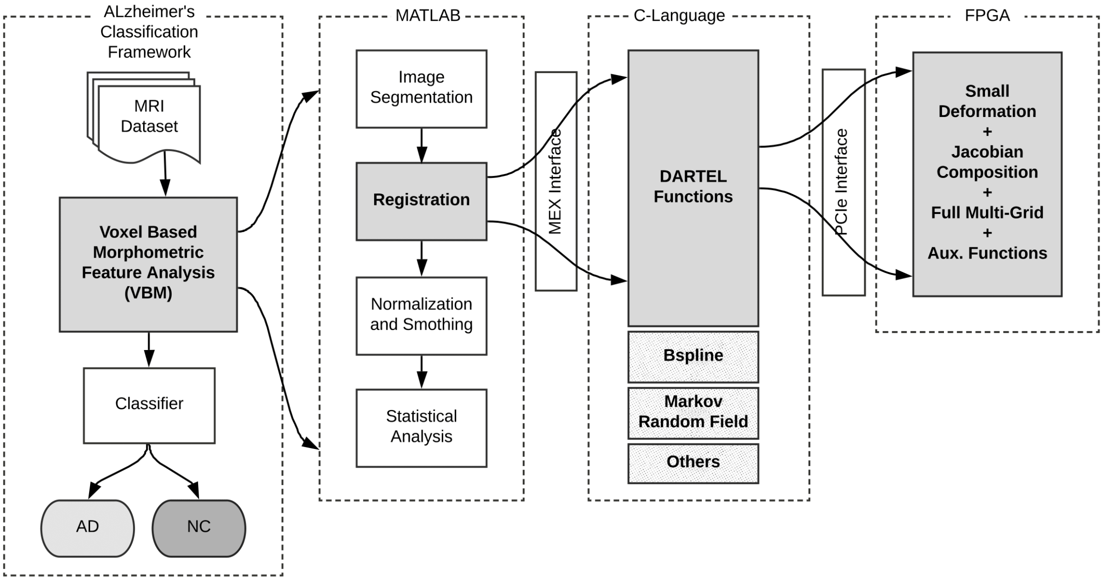

ALzheimer's Classification Framework
MRI Dataset
Voxel Based Morphometric Feature Analysis (VBM)
Classifier
AD
NC
MATLAB
Image Segmentation
Registration
Normalization and Smothing
Statistical Analysis
MEX Interface
C-Language
DARTEL Functions
Bspline
Markov Random Field
Others
PCIe Interface
FPGA
Small Deformation + Jacobian Composition + Full Multi-Grid + Aux. Functions

Fig. 2: Framework of the proposed FPGA acceleration of MRI registration used for AD classification.

The VBM analysis is implemented in the SPM software tool that runs under MATLAB. Intensive time-consuming routines are typically implemented in C language for fast implementation and are called from MATLAB via MEX interface. These routines include the DARTEL functions responsible for registration and other functions such as Bspline and Markov Random Field that are not used by the DARTEL registration algorithm. The DARTEL function itself includes three subroutines that are responsible for obtaining small deformation, calculating the Jacobian composition, solving differential equations using the full multi-grid method (FMG) and some other auxiliary functions such as the one that converts the velocity field to a momentum field. In the proposed framework, those most time-consuming functions are designed to run on FPGA for acceleration and are hence called via a PCIe interface for execution. Profiling these functions and describing the PCIe interface are described in the upcoming subsections.

On CPU, this processing forms a heavy computational and very time-consuming task. It is quite remarkable to mention that registering a total of 471 images of size 128x128x128 took 14 days on a standard 3 year relatively old desktop by DARTEL's author John Ashburner [1].

B. Profiling VBM Run Time

In the proposed software/hardware co-design, part of the VBM analysis runs on CPU, while the rest runs on the accelerating hardware FPGA. Referring to Fig. 2, the part that runs on the CPU includes image segmentation, normalizing and smoothing, and the statistical analysis. The *registration* is the part that runs on the FPGA hardware for acceleration for being the most complicated and iterative process in the framework.

In VBM analysis, the DARTEL registration runs too slow. Analyzing 275 MRI images consumed 34 hours for an image size of 121x145x121 pixels. The memory usage and run time duration of DARTEL subroutine calls are analyzed by a code profiler to discover the performance bottlenecks. Hardware specifications of the underlying machine are provided in the upcoming section.

Fig. 3 illustrates the results of profiling VBM functions run time. The VBM most time-consuming functions are implemented in C programming language and are called from MATLAB by MEX functions. This is shown in the block diagram of the proposed framework in Fig. 2. These MEX functions account for 81.4% of the total pre-processing time of the VBM execution. Profiling these functions shows that the DARTEL function which is responsible for registration takes 65% from the total MEX functions time. The DARTEL function includes four subroutines: obtaining small deformation, calculating the Jacobian composition, solving differential equations using FMG method, and other auxiliary functions (such as the function that converts the velocity field to momentum field). The percentage of the run time of each function with respect to the DARTEL function run time is given in Fig. 3. These former four functions are chosen for their high computational time-consuming demands to run on FPGA. The interface between these four functions and the CPU is held via PCIe.

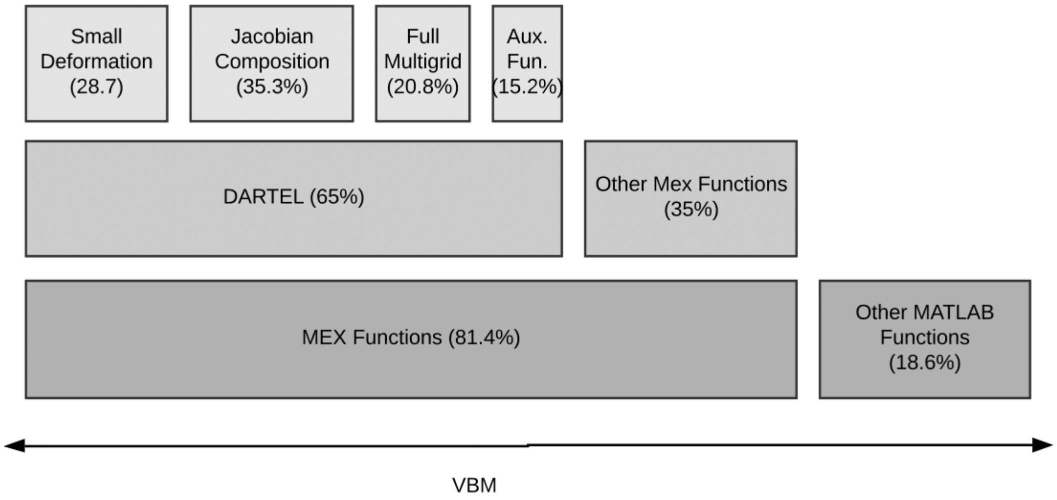


Fig. 3: Profiling VBM Functions Run Time.

C. Software/Hardware Co-design Interfacing

The used FPGA kit is interfaced with the workstation through a PCIe interface as shown in the block diagram of the proposed framework in Fig. 2. Data is transmitted from CPU to FPGA and received back by CPU after computation. This total round trip time per call τ_{call} is calculated by equation (5) where τ_{PCIe} is the transmission and reception latency taken by PCIe to transmit and receive data. The communication latency over the PCIe link is almost negligible when large amounts of data are transferred from/to the PCIe card. It is estimated by 2.3 µsec [30].

The payload is the amount of data that is transmitted/received through PCIe. $\tau_{tpayload}$ is the transmission payload calculated by equation (6). It depends on the size of the MRI used. $\tau_{rpayload}$ is the reception payload calculated by equation (7).

Throughput of the PCIe is given by [31] as 27Gbps. FPGA Operation time τ_{FPGA} is the time taken by the FPGA to calculate the DARTEL functions. Equation (5) shows that the round trip time is dominated by the FPGA Operation time τ_{FPGA} as it runs in the order of seconds while other equation variables run is microseconds.

$$\tau_{call} = 2\tau_{PCIe} + \tau_{tpayload} + \tau_{FPGA} + \tau_{rpayload} \tag{5}$$

$$\tau_{tpayload} = \frac{Bits_{arg}}{Throughput} \tag{6}$$

$$\tau_{rpayload} = \frac{Bits_{result}}{Throughput} \tag{7}$$

D. FPGA Design Optimization

The original implementation of the four functions described in the previous subsections is in the C language. It is written with no restrictions on data type range, arithmetic operations, or loop counts. The code achieved good accuracy but at the expense of a long run time.

Several optimization techniques have been introduced to the original C-code of these four functions to achieve high operating frequency, low latency, and low power circuits. These techniques include:

- Replacing floating-point data types with fixed-point data types;
- Optimizing complex and/or non-linear arithmetic functions;
- Exploiting the concurrent behavior of FPGAs by parallel loop unrolling and function inlining;
- Pipelining.

1) *Fixed-Point Data Type*

In fixed-point data type, the number is represented with a fixed number of digits after and before the radix point. The fundamental advantage of using a fixed-point data type is performance. Arithmetic operations performed on fixed-point data type are orders of magnitude faster than floating-point data type. The fixed-point conversion reduces the usage of FPGA resources such as DSP48E2s, look-up tables (LUTs), and flip-flops. It also reduces the amount of memory required to store the data.

The downside of using a fixed-point data type is accuracy. Its performance improvement comes at the expense of accuracy due to the reduced range of values that the fixed number of digits can represent. Arithmetic manipulation of large numbers can produce a result that does not fit into the fixed-point representation. Scaling the data becomes a necessity to avoid problems such as overflow or underflow.

However, in many scientific areas, fixed-point implementation achieves comparable performance in terms of accuracy to the floating-point implementation. According to [32], machine learning problems such as image classification requires only INT8 fixed-point data type to reach an acceptable accuracy. In this design, a good trade-off between accuracy and hardware utilization, power consumption, and run time is reached by using a fixed-point data type of 16 precision bits.

2) *Complex and/or non-linear arithmetic functions*

Complex arithmetic operations such as division and IEEE-defined elementary operations (such as log(x) and e^x functions) consume a very high number of clock cycles and utilize a huge number of hardware resources such as LUTs and DSP48E2 slices. These functions were excluded from the code and replaced with faster-approximated implementations that use simpler arithmetic operations such as addition, subtraction, and multiplication.

Division and square root functions are implemented using Newton Raphson iterative algorithm. The algorithm finds the reciprocal of the denominator and multiplies that reciprocal by the numerator to find the final quotient [33]. It computes one multiplicative inverse in one iteration thus providing high computation speed and throughput.

Log and exponential functions are implemented using look-up table based algorithms. This approach is fast and straightforward. However, it requires a large amount of memory to avoid reducing the accuracy. This explains the vast amount of LUTs used by the design.

3) *Loop unrolling and function in-lining*

In order to translate software C-code algorithms into hardware circuits that match the same functionality, High-Level Synthesis 'HLS' tools are used. Pragmas defined by the Vivado HLS tool are embedded in the C-code to fine-tune the synthesized output circuit resulting in a faster, low-latency, and low-power circuit.

The design applied the INLINE pragma. This pragma allows functions that are called from another function to be dissolved into the calling function. This allows operations to be optimized more efficiently. Loop unrolling is applied as well in the design using the UNROLL pragma to insure maximum operations concurrency. Unrolling loops create multiple independent operations by generating multiple copies of the loop body in the design allowing loop iterations to occur in parallel.

4) *Pipelining*

Parallel execution of instructions by using pipelining allows the design to take full advantage of FPGA. The initiation interval is defined as the number of clock cycles between successive instances of the loop. It is considered the inverse of the throughput. Throughout the whole design all pipeline stages are implemented with the lowest possible initiation interval. Concurrent execution of operations is applied in the design by using PIPELINE pragma to reduce the initiation interval for loops. The default initiation interval for the PIPELINE pragma is 1, which means processing a new input every clock cycle.

FPGA design that applied these optimizations will be referred to in the text as optimized FPGA fixed-point implementation.

V. EXPERIMENTAL STUDY, RESULTS AND DISCUSSION

A. Dataset

Alzheimer's Disease Neuroimaging Initiative (ADNI) [34] is the dataset used for testing in this work. It contains various study data that help in defining the progression of AD such as MRI and PET images. T1 and T2, are two relaxation times that are used to characterize MRI scans.

In this study, a total of 275 T1-weighted MR images are considered, including 162 patients with AD and 113 NC. Table-I clarifies the dataset demographics.

Table-I: Dataset Demographics

	Normal Control	Alzheimer's Disease Patient
	Total Number=113	Total Number=162
Gender: (M/F)	72/41	71/91
Age: (Mean/Std.)	77.49/5.88	73.82/7.63
MMSE: (Mean/Std.)	25.74/7.74	21.54/3.92

B. Experimental Results

The study compares three different implementations of DARTEL registration algorithm:

1) Software (CPU) implementation;
2) FPGA floating point implementation;
3) Optimized FPGA fixed-point implementation.

Comparing the software (CPU) implementation and the optimized FPGA fixed-point implementation highlights the impact of using FPGA accelerators on the registration problem. While comparing the optimized FPGA fixed-point implementation to the FPGA floating-point implementation analyzes the trade-off between accuracy and hardware utilization, power consumption, and run time. Finally, a comparison between the three implementations is provided.

1) Acceleration of FPGA implementation over CPU implementation

CPU run time of the DARTEL registration function running on MATLAB is 10.9 seconds per call. The Small Deformation function accounts for 28.7% of its time.

The accelerated version of the Small Deformation function running on FPGA consumes 1.023 seconds. Thus, a speedup factor of 2.75x is achieved over the CPU run time. Table II summarizes the CPU run time, FPGA run time, and the achieved speedup factors for the four implemented DARTEL functions. The speedup factor for the whole DARTEL registration process when combining the four accelerated functions is 7.9x.

CPU results have been carried on a machine with an Intel i5-4200M processor that has a 64-bit architecture and runs at 2.50GHz clock frequency under Ubuntu 18.04 operating system.

The proposed FPGA design is synthesized on a Xilinx Zynq UltraScale+ MPSoC ZCU106 Evaluation Kit that includes a ZU7EV FPGA device and runs at 2 nanoseconds clock.

Table-II: Run Time Speedup Factors of different DARTEL Functions

	CPU	FPGA	Speedup Factor
Small Deformation	3.14s	1.14s	2.75x
Jacobian Composition	3.87s	0.05s	77.5x
FMG	2.28s	0.02s	114x
Auxiliary functions	1.6s	0.17s	9.4x

2) Acceleration of optimized FPGA fixed-point implementation over FPGA floating-point implementation

Fig. 4 compares the floating-point implementation and the optimized fixed-point implementation. The comparison comes in terms of frequency, latency, BRAM, and dynamic power. The maximum frequency of the optimized fixed-point implementation is more than twice the frequency of the floating-point implementation. The optimized fixed-point implementation achieved reductions in Latency, BRAM, and dynamic power as well.

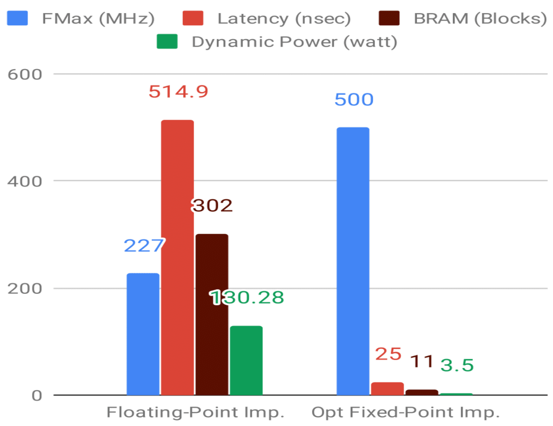


Fig. 4: FPGA Performance Measurement of Floating-Point Implementation and Optimized Fixed-Point Implementation.

Fig. 5 and Fig. 6 compares the LUTs count and the DSP48E slices count of the two FPGA implementations; optimized fixed-point versus floating-point respectively. A significant decline in both counts is obtained by the optimized fixed-point implementation.

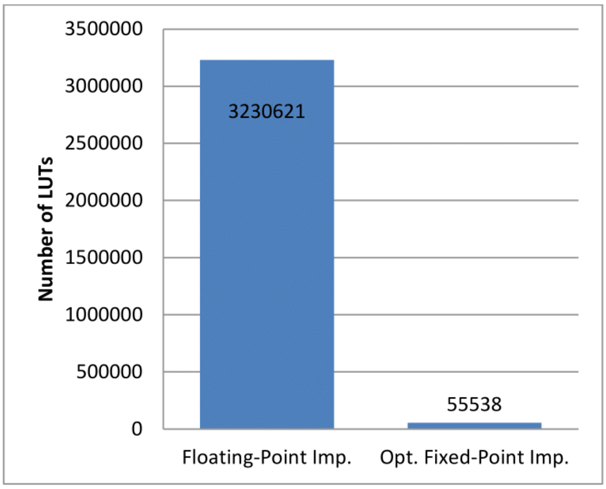


Fig. 5: LUTs Count of Floating-Point Implementation and Optimized Fixed-Point Implementation.

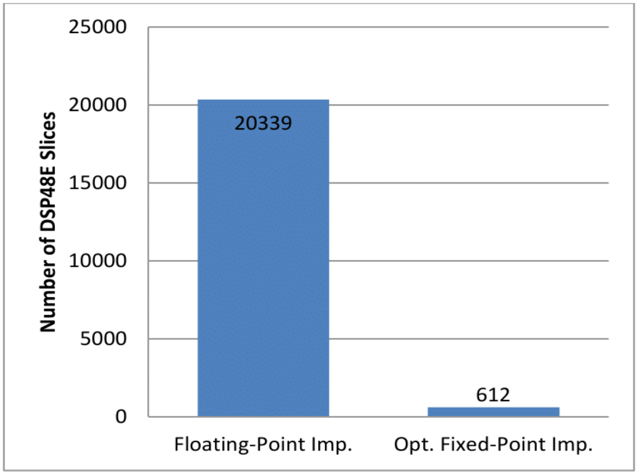


Fig. 6: DSP48E Slices Count of Floating-Point Implementation and Optimized Fixed-Point Implementation.

Fig. 7 illustrates the improvement factors achieved by the optimized fixed-point implementation over the floating-point implementation. A Massive decrease in BRAM, 27x, LUTs, 58x, and in DSP48E slices, 33x is noticed. These resource savings has led to a reduction in the dynamic power by 37x.

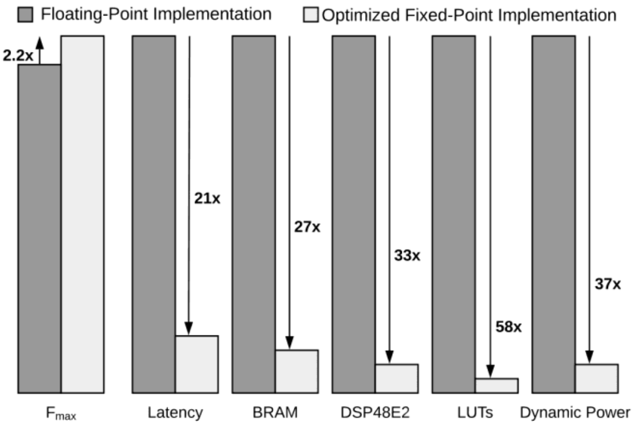


Fig. 7: Improvement Factors achieved by Optimized Fixed-Point Implementation Over Floating-Point Implementation.

Table- III shows a summary of the resources utilized by the design versus the available resources on board. This indicates utilization of about one third for DSP48E and one fourth for LUTs and 1% for BRAM. The power consumption details of the design are illustrated in Fig. 8. Small Deformation function consumes most of the design power which is quite predictable as it extensively uses the LOG and division functions. FMG consumes far less power as it only uses simple arithmetic operations. Auxiliary functions and Jacobian Composition do not utilize BRAM in their circuits and thus no power consumption is shown.

Table-III: FPGA utilization Vs Available resources.

	DSP48E	LUTs	BRAM
Total	612	55538	1600
Available	1728	230000	11
Utilization (%)	35.42%	24.15%	1%

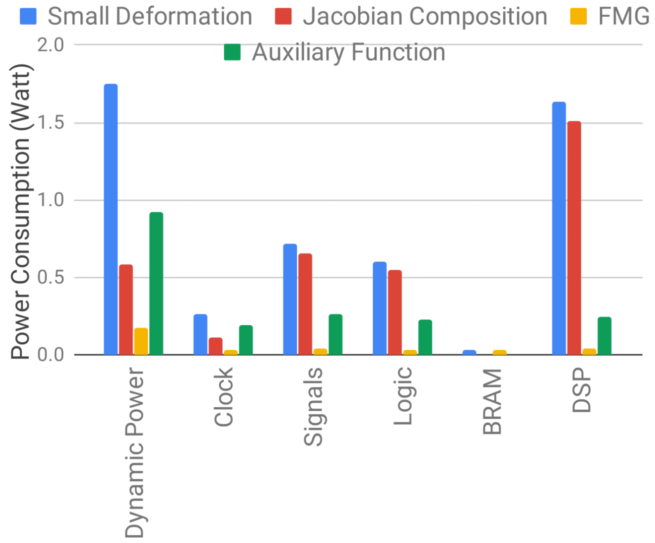


Fig. 8: Power Consumption of different DARTEL Functions

Fig. 9 illustrates the Root Mean Square Error (RMSE) using different data types. RMSE is calculated with reference to double (64-bit) floating-point datatype. Single (32-bit) floating-point and (32-bit) fixed-point datatypes barely affected the accuracy (i.e. their RMSE approaches zero). The figure shows that the proposed design with its 16-bit fixed-point datatype achieved less than 10% RMSE, while the 8-bit fixed-point datatype achieved more than 30% RMSE. Hence, the choice of 16-bit fixed-point datatype is concluded to achieve the best trade-off between accuracy and hardware utilization, power consumption, and run time.

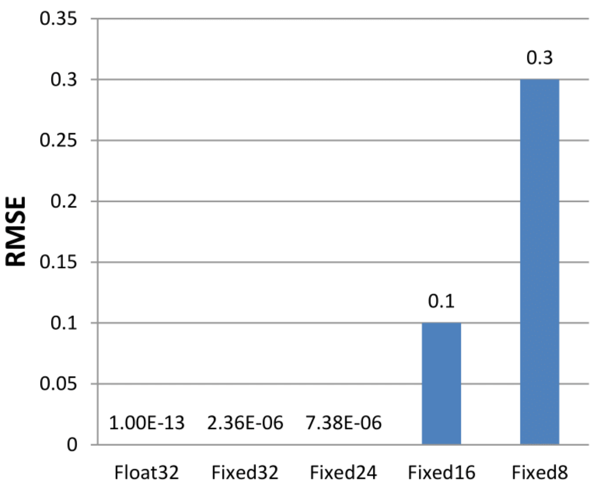


Fig. 9: RMSE using Different Data Types

The DARTEL run time on CPU, FPGA with floating-point implementation, and FPGA with optimized fixed-point implementation is illustrated in Fig. 10. The run time, which is in nanoseconds, indicated that optimized FPGA fixed-point implementation achieves a time acceleration of 7.9x over CPU and 20.7x over FPGA floating-point implementation.

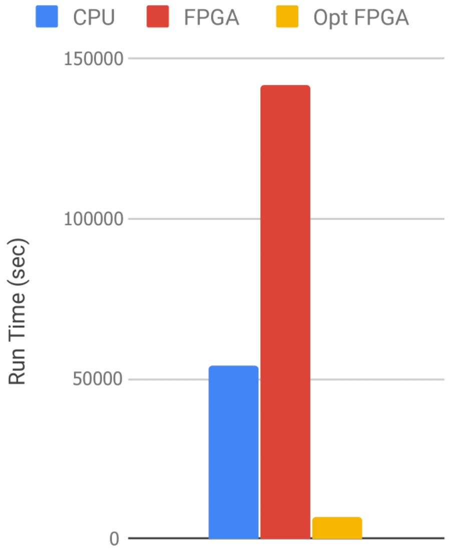


Fig. 10: DARTEL Run Time of CPU, Floating-Point FPGA implementation (FPGA), and Optimized Fixed-Point FPGA implementation (Opt FPGA)

C. Discussion

The proposed design, optimized fixed-point implementation, consumed far fewer resources than the available resources on board. It consumed 24% of LUTs, 35% of DSP48E2 slices, and 1% of BRAM. This shows very good resource utilization and allows for either one of the following:

- Adding more accelerated functions to the chip;
- Replicating the circuit for more acceleration rates.

The design can roughly be replicated three times leading to tripling the acceleration rate.

The optimized fixed-point implementation design can be compared to the GPU implementation of DARTEL registration conducted by P. Valero-Lara in [27]. Though the datasets are different, both designs can be compared in terms of performance measures. The database used for evaluation in the case of GPU implementation is only 30 images with the size 60x60x72 pixels, accounting for a total of 259,200 pixels per image. For a single GPU implementation, an overall speedup of 13x is achieved and is promoted to 63x using four GPUs according to the author.

The work presented here uses 275 images of size 121x145x121 pixels, accounting for a total of 2,122,945 pixels per image. Accordingly, the workload used in this work exceeds the workload used in [27] by approximately 8.2x and the database size is bigger by approximately 9.2x. Moreover, the low resource utilization of the design allows it to be replicated three times as discussed earlier. This can promote the speedup by triple the value, leading to 23.7x speedup over CPU. Thus, the FPGA implementation proposed here presents competitive results to the single GPU implementation conducted by [27] The GPU platform used by P. Valero-Lara consists of four NVIDIA Tesla K20 graphic card, Intel Xeon (E5-2650) runs at 2GHz and 64GB main memory. Table-IV highlights the comparison between FPGA optimized fixed-point implementation and GPU implementation speedup factors on the three main DARTEL functions: Small Deformation, Jacobian Composition, and FMG. 'Auxiliary functions' refer to functions that implement the rest of the DARTEL routine. It is accelerated in this work only, and for the GPU implementation, their result is not recorded by the author. The comparison as indicated by table 4 gives an insight into the powerful acceleration power of FPGA over GPU.

Table-IV: FPGA Vs GPU speedup factors on different DARTEL functions.

	FPGA	GPU
Small Deformation	2.75x	32x
Jacobian Composition	77.5x	19x
FMG	114x	9x
Auxiliary functions	9.4x	-

Pre-processing of data has been carried by MATLAB R2014b [35] by SPM12 on the DARTEL toolbox. C to VHDL code conversion and synthesis is done using Vivado High-Level Synthesis (HLS) software [36] Profiling is done using gprof and google performance tools.

VI. CONCLUSION AND FUTURE WORK

Image registration is an important highly computational process in image analysis. This paper presents an optimized software/hardware co-design model to accelerate the implementation of the diffeomorphic image registration algorithm DARTEL running under the SPM software tool. An optimized and pipelined FPGA architecture is proposed, where faster implementations of complex arithmetic operations and fixed-point data types are exploited to ensure minimum latency. The proposed fixed-point FPGA implementation achieves more than 20x speedup in run time when compared to the floating-point FPGA implementation. Comparing to CPU performance, the speedup factor of 114x is achieved on the function level with an overall speedup factor of 8x of the DARTEL registration process. The low resource utilization of the design on the FPGA board, which is one third, makes it flexible to be replicated three times promising to triple the acceleration rate. The acceleration capabilities of FPGA versus GPU in registering medical images are explored in this paper and evaluated. FPGA proved proficiency by the powerful resource savings and high speedups obtained.

FPGA can be further utilized by using multiple CPU cores on the FPGA board. This can efficiently run the software application with low latency to the hardware accelerator.

Moreover, the accuracy of the system can be enhanced by applying the fixed-point word-length optimization methodologies and techniques.

REFERENCES

1. Ashburner, John. "A fast diffeomorphic image registration algorithm." *Neuroimage* 38.1 (2007): 95-113.
2. Klein, Arno, et al. "Evaluation of 14 nonlinear deformation algorithms applied to human brain MRI registration." *Neuroimage* 46.3 (2009): 786-802.
3. Khoon, Lee Lay, and Liew Siau Chuin. "A survey of medical image processing tools." *International Journal of Software Engineering and Computer Systems (IJSECS)* 2.1 (2016): 10-27.
4. Penny, William D., et al., eds. *Statistical parametric mapping: the analysis of functional brain images*. Elsevier, 2011.
5. Ashburner, John, and Karl J. Friston. "Voxel-based morphometry—the methods." *Neuroimage* 11.6 (2000): 805-821..
6. Zhang, Feng, et al. "Voxel-based morphometry: improving the diagnosis of Alzheimer's disease based on an extreme learning machine method from the ADNI cohort." *Neuroscience* 414 (2019): 273-279.
7. Zhang, Haifeng, et al. "Computerized multi-domain cognitive training reduces brain atrophy in patients with amnestic mild cognitive impairment." Translational psychiatry 9.1 (2019): 1-10.
8. Pergher, Valentina, et al. "Identifying brain changes related to cognitive aging using VBM and visual rating scales." NeuroImage: Clinical 22 (2019): 101697.
9. BERTEN, DSP. "GPU vs FPGA performance comparison." Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays-FPGA'17. 2016.
10. Birk, Matthias, et al. "A comprehensive comparison of GPU-and FPGA-based acceleration of reflection image reconstruction for 3D ultrasound computer tomography." Journal of real-time image processing 9.1 (2014): 159-170.
11. Chase, Jeff, et al. "Real-time optical flow calculations on FPGA and GPU architectures: a comparison study." 2008 16th International Symposium on Field-Programmable Custom Computing Machines. IEEE, 2008.

12. Pauwels, Karl, et al. "A comparison of FPGA and GPU for real-time phase-based optical flow, stereo, and local image features." IEEE Transactions on Computers 61.7 (2011): 999-1012.
13. Nurvitadhi, Eriko, et al. "Can FPGAs beat GPUs in accelerating next-generation deep neural networks?." Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. 2017.
14. Farouk, Yasmeen, Sherine Rady, and Hossam Faheem. "Statistical features and voxel-based morphometry for alzheimer's disease classification." *2018 9th International Conference on Information and Communication Systems (ICICS)*. IEEE, 2018.
15. Kasik, Vladimir, et al. "Advanced CT and MR image processing with FPGA." International Conference on Intelligent Data Engineering and Automated Learning. Springer, Berlin, Heidelberg, 2012.
16. Koo, Jahyun J., Alan C. Evans, and Warren J. Gross. "3-D brain MRI tissue classification on FPGAs." IEEE Transactions on Image Processing 18.12 (2009): 2735-2746.
17. Chandra, Binay, and Manish Sharma. "Segmentatation based feature extraction of MRI images using Wavelet and implementation on FPGA." 2016 International Conference on Automatic Control and Dynamic Optimization Techniques (ICACDOT). IEEE, 2016.
18. Christe, S. Allin, M. Vignesh, and A. Kandaswamy. "An efficient FPGA implementation of MRI image filtering and tumor characterization using Xilinx system generator." arXiv preprint arXiv:1201.2542 (2012).
19. Hasan, Sami, Alex Yakovlev, and Said Boussakta. "Performance efficient FPGA implementation of parallel 2-D MRI image filtering algorithms using Xilinx system generator." 2010 7th International Symposium on Communication Systems, Networks & Digital Signal Processing (CSNDSP 2010). IEEE, 2010.
20. Li, Will XY, et al. "High-performance computing for neuroinformatics using FPGA." High-Performance Computing Using FPGAs. Springer, New York, NY, 2013. 177-207.
21. Minati, Ludovico, Mara Cercignani, and Dennis Chan. "Rapid geodesic mapping of brain functional connectivity: Implementation of a dedicated co-processor in a field-programmable gate array (FPGA) and application to resting state functional MRI." Medical Engineering & Physics 35.10 (2013): 1532-1539.
22. Lee, Lay-Khoon, and Siau-Chuin Liew. "A survey of medical image processing tools." 2015 4th International Conference on Software Engineering and Computer Systems (ICSECS). IEEE, 2015.
23. Hernández, Moisés, et al. "Accelerating fibre orientation estimation from diffusion weighted magnetic resonance imaging using GPUs." PloS one 8.4 (2013): e61892.
24. Yan, Chaogan, and Yufeng Zang. "DPARSF: a MATLAB toolbox for" pipeline" data analysis of resting-state fMRI." Frontiers in systems neuroscience 4 (2010): 13.
25. Shamonin, Denis P., et al. "Fast parallel image registration on CPU and GPU for diagnostic classification of Alzheimer's disease." Frontiers in neuroinformatics 7 (2014): 50.
26. Huang, Teng-Yi, Yu-Wei Tang, and Shiun-Ying Ju. "Accelerating image registration of MRI by GPU-based parallel computation." Magnetic resonance imaging 29.5 (2011): 712-716.
27. Valero-Lara, Pedro. "Multi-gpu acceleration of DARTEL (early detection of alzheimer)." 2014 IEEE International Conference on Cluster Computing (CLUSTER). IEEE, 2014.
28. Huang, Teng-Yi, Yu-Wei Tang, and Shiun-Ying Ju. "Accelerating image registration of MRI by GPU-based parallel computation." Magnetic resonance imaging 29.5 (2011): 712-716.
29. Mechelli, Andrea, et al. "Voxel-based morphometry of the human brain: methods and applications." *Current Medical Imaging* 1.2 (2005): 105-113.
30. Fahmy, H. A., ElDeeb, T., Hassan, M. Y., Farouk, Y., & Eissa, R. R. (2010, December). Decimal Floating Point for future processors. In *2010 International Conference on Microelectronics* (pp. 443-446). IEEE.
31. Goldhammer, Alex, and John Ayer Jr. "Understanding performance of PCI express systems." *Xilinx WP350, Sept* 4 (2008).
32. Finnerty, Ambrose, and Herve Ratigner. "Reduce power and cost by converting from floating point to fixed point." *WP491 (v1. 0)* (2017).
33. Singh, Naginder, and Trailokya Nath Sasamal. "Design and synthesis of single precision floating point division based on newton-raphson algorithm on fpga." *MATEC Web of Conferences*. Vol. 57. EDP Sciences, 2016.
34. Alzheimer’s Disease Neuroimaging Initiative (ADNI), http://adni.loni.usc.edu/.
35. Matlab software. URL http://www.mathworks.com/products/matlab/
36. Vivado high-level synthesis, xilinx software. https://www.xilinx.com/products/design-tools/vivado/integration/esldesign.
37. Haber, Eldad, and Jan Modersitzki. "A multilevel method for image registration." SIAM Journal on Scientific Computing 27.5 (2006): 1594-1607.

AUTHORS PROFILE

Yasmeen Farouk is a PhD student at the Faculty of Computer and Information Sciences of Ain Shams University in Cairo, Egypt. She holds a B.Sc in Computer and Information Sciences (Information Systems Department), Ain Shams University. She got her M.Sc. in Computer and Information Sciences from Ain Shams University**.**

Sherine Rady is an Associate Professor at the Faculty of Computer and Information Sciences of Ain Shams University in Cairo, Egypt. She holds a B.Sc in Electrical Engineering (Computer and Systems), Ain Shams University. She got her M.Sc. in Computer and Information Sciences from Ain Shams University and her Ph.D. from University of Mannheim in Germany. – Dr. Sherine Rady is a DAAD and JICA Alumni and her current research interests are Data Mining, Computer Vision, Robotics, and Big Data.