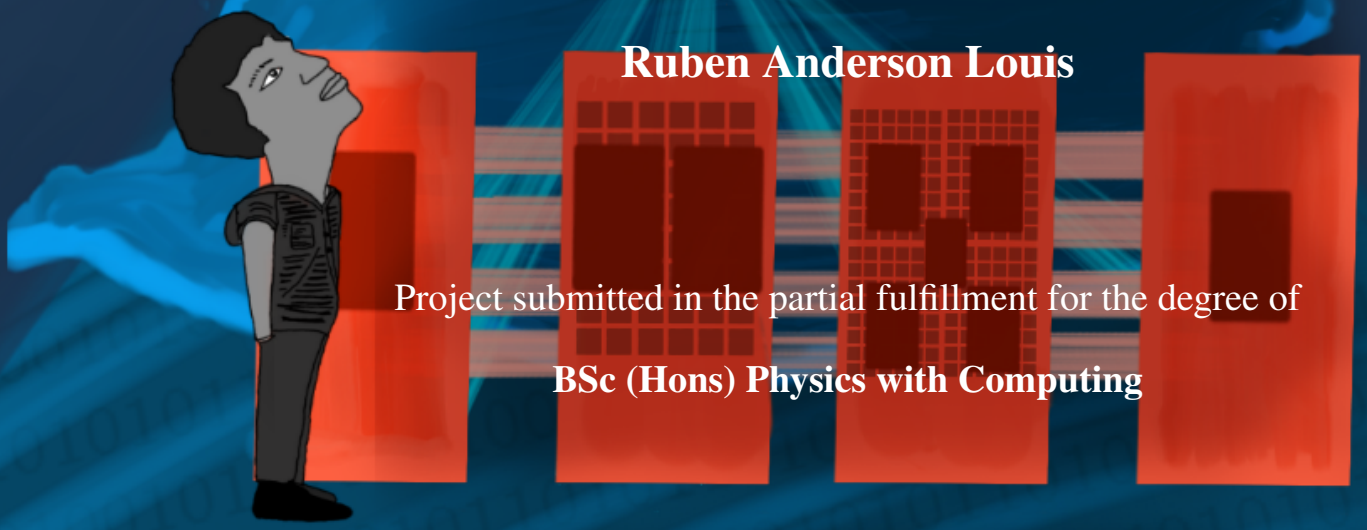
Digital Systems for the MITRA

Submitted by

Ruben Anderson Louis



Project submitted in the partial fulfillment for the degree of

BSc (Hons) Physics with Computing

UNIVERSITY OF MAURITIUS

FACULTY OF SCIENCE

DEPARTMENT OF PHYSICS

April 3, 2015

UNIVERSITY OF MAURITIUS

PROJECT/DISSERTATION DECLARATION FORM

Name:	LOUIS, Ruben Anderson
Student ID:	1210449
Programme of Studies:	BSc(Hons) Physics with Computing
Module Code/Name:	PHYSI 3000Y(5) Project/Dissertation
Title of Project/Dissertation:	Digital systems for the MITRA
Name of Supervisor(s):	Dr. Beeharry Girish Kumar

Declaration:

In accordance with the appropriate regulations, I hereby submit the above dissertation for examination and I declare that:

(i) I have read and understood the sections on **Plagiarism and Fabrication and Falsification of Results** found in the University's "General Information to Students" Handbook (20…./20….) and certify that the dissertation embodies the results of my own work.

(ii) I have adhered to the 'Harvard system of referencing' or a system acceptable as per "The University of Mauritius Referencing Guide" for referencing, quotations and citations in my dissertation. Each contribution to, and quotation in my dissertation from the work of other people has been attributed, and has been cited and referenced.

(iii) I have not allowed and will not allow anyone to copy my work with the intention of passing it off as his or her own work.

(iv) I am aware that I may have to forfeit the certificate/diploma/degree in the event that plagiarism has been detected after the award.

(v) Notwithstanding the supervision provided to me by the University of Mauritius, I warrant that any alleged act(s) of plagiarism during my stay as registered student of the University of Mauritius is entirely my own responsibility and the University of Mauritius and/or its employees shall under no circumstances whatsoever be under any liability of any kind in respect of the aforesaid act(s) of plagiarism.

Signature: 	Date: 03/04/2015

To my parents

“Computer science is no more about computers than astronomy is about telescopes.”

Edsger Dijkstra

UNIVERSITY OF MAURITIUS

Abstract

Faculty of Science

Department of Physics

BSc (Hons) Physics with computing

Digital Systems for the MITRA

by Ruben Anderson Louis

This dissertation describes the development and testing of software in relation to the integration of Digital Systems in the Multifrequency Interferometry Telescope for Radio Astronomy (MITRA). The feasibility of high performance software correlation is investigated, with the use of the DiFX (Distributed FX) software by Deller et al. (2007), developed as part of the Australian Major National Research Facilities Programme, and operated under licence. The DiFX software was tested on heterogeneous setups, for the sake of re-conversion of unused hardware, to perform software correlation tasks. Also the possibilities offered by General-purpose computing on Graphics Processing Units (GPGPU) for software correlation, with low power, and highly efficient SoCs GPUs is discussed.

Acknowledgements

The road which lead to the accomplishment of my undergraduate project and dissertation was a long one and was not taken alone.

Above all, as I am a faithful believer, and I believe that nothing happens without a reason, I wish to thank God for everything, and making all of this possible whichever way it may have happened.

I also wish to acknowledge the following, first I am very grateful to my supervisor, Dr. Girish Kumar Beeharry, for the opportunity he gave me to work on the topic, he has been an inspirational lecturer who encouraged me to pursue in the field of scientific computing, also most importantly I wish to thank him for his guidance, useful suggestions during the course of this project, and for his reviews, remarks and comments during the writing process of the dissertation. I express my thanks to the lecturers of the department of physics who during those 3 years here at the University of Mauritius contributed to build up the physicist I am today.

My special thanks goes to the MPhil/PhD student, Mr. Vinand Prayag, who was the first to trust me to work on the specific topic of software correlation, for his help, and useful advices from his sole experience with the ongoing MITRA project.

I am also very thankful to the technical staff of the Mauritius Radio Telescope and that of the department of physics, the university CITS unit, who were keen to let me use the facilities available, both at the MRT and the laboratory of the department of physics, for practical sessions, and the implementation of my project.

Extra credit goes to my classmates with whom we struggled with the BSc (Hons) Physics course, specially those of the computing branch who will recognise themselves, where the last two years together were a lot of fun and made my stay at the university a lot nicer, and also not forgetting my year 2 classmates with whom I worked on small projects.

I would also like to thank those closest to me for their support, and my family, who has always been behind me and helped me greatly to continue in the field of physics at a university level.

Contents

Declaration of Authorship

Abstract ii

Acknowledgements iii

List of Figures vi

List of Tables viii

Abbreviations ix

Physical Constants x

1 Introduction 1
- 1.1 Radio Astronomy . . . 1
- 1.2 Software Defined Radio . . . 4
- 1.3 Digital Systems for the MITRA . . . 5
- 1.4 Outline . . . 6

2 Synthesis Imaging 7
- 2.1 Principles of Radio Interferometry . . . 7
  - 2.1.1 Visibility – Sky intensity distribution relationship . . . 10
- 2.2 The Van Cittert-Zernike theorem . . . 11
- 2.3 The correlator . . . 13
- 2.4 Digital Correlators . . . 16
  - 2.4.1 FX Correlators . . . 16

3 Distributed FX Correlation - The DiFX software 18
- 3.1 Introduction . . . 18
- 3.2 Working with datasets . . . 19
  - 3.2.1 The workflow . . . 20
  - 3.2.2 Visualising the output . . . 25
- 3.3 Experiment - Pseudo Data . . . 28

4 Alternative General Purpose Computing Hardware - GPGPU prospects 40
- 4.1 General Purpose Computing on Graphics Processing Units (GPGPU) . . . 40
  - 4.1.1 The GPU and Software correlators . . . 40

5 Conclusions and future work **44**
5.1 Conclusions 44
5.2 Future work 44

A DiFX - Custom documentation **45**
A.1 DiFX - Setup 45
A.2 Using DiFX 48
A.3 Custom Modification 49
A.4 Experiments 58

B Progress Logs **61**

Bibliography **65**

List of Figures

1.1 The Southern sky at 151.6 MHz . . . 1
1.2 The effect of diffraction . . . 2
1.3 Antenna Pattern . . . 2
1.4 Angular Resolution . . . 3
1.5 Sir Martin Ryle . . . 4
1.6 Cygnus A . . . 4

2.1 An elementary interferometer . . . 7
2.2 Free coordinate system . . . 8
2.3 Element of Solid Angle . . . 9
2.4 Fringe Visibility . . . 11
2.5 Amplitude and fringe of the visibility function, where the source is offset the phase centre . . . 12
2.6 Amplitude and fringe of the visibility function . . . 12
2.7 A simplistic representation of a correlator . . . 13
2.8 FX correlator . . . 16

3.1 DiFX a distributed FX correlator . . . 18
3.2 DiFX the correlation workflow (Deller et al., 2007) . . . 20
3.3 DiFX Run Time . . . 22
3.4 DiFX Run Time . . . 22
3.5 DiFX Speed Up . . . 23
3.6 DiFX Speed Up . . . 23
3.7 DiFX Speed Up efficiency . . . 23
3.8 DiFX Speed Up efficiency . . . 24
3.9 DiFX Speed Up per watt . . . 24
3.10 DiFX Speed Up per watt . . . 24
3.11 DiFX Speed Up per cost . . . 25
3.12 DiFX Speed Up per cost . . . 25
3.13 DiFX - RDV70 . . . 26
3.14 DiFX - RDV70 - Power Spectrum for the band 8406-8414 MHz Baseline KK-KK . . . 26
3.15 DiFX - RDV70 - Power Spectrum for the band 8406-8414 MHz Baseline KK-KP . . . 27
3.16 DiFX - V252-LBA Dataset - Baseline AT-AT data . . . 28
3.17 DiFX - v252-LBA - 2-bit +ve one - baseline AT-AT . . . 29
3.18 DiFX - v252-LBA - 2-bit +ve one - Power Spectrum - baseline AT-AT . . . 30
3.19 DiFX - v252-LBA - 2-bit +ve one - baseline AT-CD . . . 31
3.20 DiFX - v252-LBA - 2-bit +ve one - Power Spectrum - baseline AT-CD . . . 31
3.21 DiFX - v252-LBA - 2-bit -ve one - baseline AT-AT . . . 31
3.22 DiFX - v252-LBA - 2-bit -ve one - Power Spectrum - baseline AT-AT . . . 32
3.23 DiFX - v252-LBA - 2-bit -ve one - baseline AT-CD . . . 32

3.24 DiFX - v252-LBA - 2-bit -ve one - Power Spectrum - baseline AT-CD 33
3.25 DiFX - v252-LBA - 2-bit zeros - baseline AT-AT 33
3.26 DiFX - v252-LBA - 2-bit zeros - Power Spectrum - baseline AT-AT 34
3.27 DiFX - v252-LBA - 2-bit zeros - baseline AT-CD 34
3.28 DiFX - v252-LBA - 2-bit zeros - Power Spectrum - baseline AT-CD 35
3.29 DiFX - v252-LBA - 2-bit random values - baseline AT-AT 35
3.30 DiFX - v252-LBA - 2-bit random values - Power Spectrum - baseline AT-AT . . 36
3.31 DiFX - v252-LBA - 2-bit random values - baseline AT-CD 36
3.32 DiFX - v252-LBA - 2-bit random values - Power Spectrum - baseline AT-CD . . 37
3.33 DiFX - v252-LBA - 2-bit +ve square pulse - baseline AT-AT 37
3.34 DiFX - v252-LBA - 2-bit +ve square pulse - Power Spectrum - baseline AT-AT . 38
3.35 DiFX - v252-LBA - 2-bit +ve square pulse - baseline AT-CD 38
3.36 DiFX - v252-LBA - 2-bit +ve square pulse - baseline AT-CD 39

4.1 (CUDA, 2015) CPU vs GPU 41
4.2 (Woods et al., 2010) GPU correlator implementation 41
4.3 Ratcliffe (2015) Process Load 42
4.4 Ratcliffe (2015) Embedded GPU SoC Design 43

List of Tables

3.1 Setups used 19
3.2 Data sets 20

Abbreviations

CUDA™	**C**ompute **U**nified **D**evice **A**rchitecture
CPU	**C**entral **P**rocessing **U**nit
EM	**E**lectro**m**agnetic
FFT	**F**ast **F**ourier **T**ransform
GPU	**G**raphics **P**rocessing **U**nit
GPGPU	**G**eneral **P**urpose computing on **G**raphics **P**rocessing **U**nits
HDD	**H**ard **D**isk **D**rive
LBA	The Australian **L**ong **B**aseline **A**rray
MITRA	**M**ultifrequency **I**nterferometry **T**elescope for **R**adio **A**stronomy
MPI	**M**essage **P**arsing **I**nterface
MRT	**M**auritius **R**adio **T**elescope
R.A.	**R**adio **A**stronomy
SDR	**S**oftware **D**efined **R**adio
SoC	**S**ystem **o**n **C**hip
SSD	**S**olid **S**tate **D**rive
TDP	**T**hermal **D**esign **P**ower
VLA	**V**ery **L**arge **A**rray (Radiotelescope)
VLBI	**V**ery **L**ong **B**aseline **I**nterferometry

Physical Constants

Speed of Light	c	$=$	$2.997\,924\,58 \times 10^{8}$ ms^{-1} (exact)
1 Jansky		=	$10^{-26}\,\mathrm{W\,m^{-2}\,Hz^{-1}}$

Chapter 1

Introduction

In this chapter, a brief introduction is given on

- radio astronomy,
- the goals of the MITRA project and
- the contents of this dissertation.

1.1 Radio Astronomy

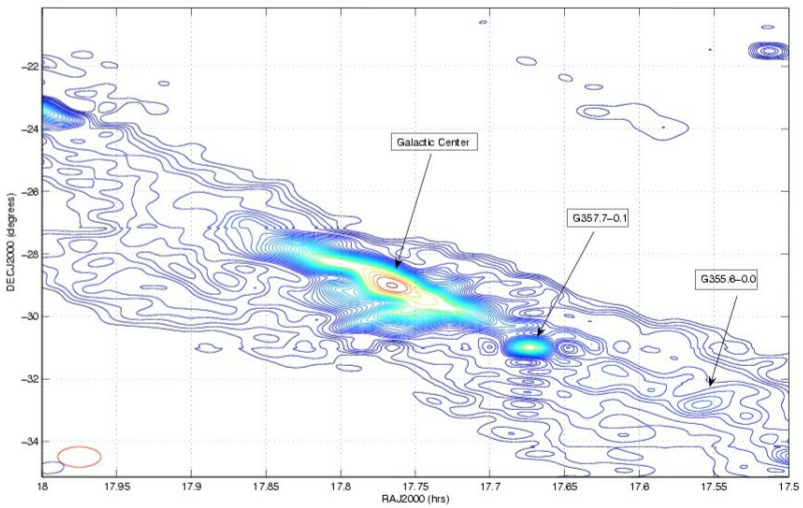


FIGURE 1.1: The Southern sky with the MRT T-array at 151.6 MHz[1]. At these low frequencies synchrotron sources are much more dominant than at higher frequencies (Golap et al., 1998)

[1] http://www.uom.ac.mu/mrt/17.5-18h.jpg

Radio astronomy is simply put the study of celestial objects by collecting and analysing the radio waves (electromagnetic radiation usually at frequencies of the order of the MHz to the THz they emit (www.skatelescope.org, 2014)). Typical radio emission result from mechanisms such as thermal processes, as in blackbody-radiation, free-free emission in an ionized gas, and spectral line emission, where a notable feature is the hydrogen 21 cm line (1420 MHz), and non thermal processes, which include synchroton radiation, gyrosynchroton emission, and amplified emission from masers for example (NRAO, 2014). With radio astronomy, scientists can study astronomical phenomena that are often invisible (e.g. see fig. 1.1) in other portions of the electromagnetic spectrum (NRAO, 2014).

FIGURE 1.2: (a) A point source, (b) The diffracted recording of a point source through a finite sized aperture (Woods et al., 2010, Appendix B.1, Fig. B.1)

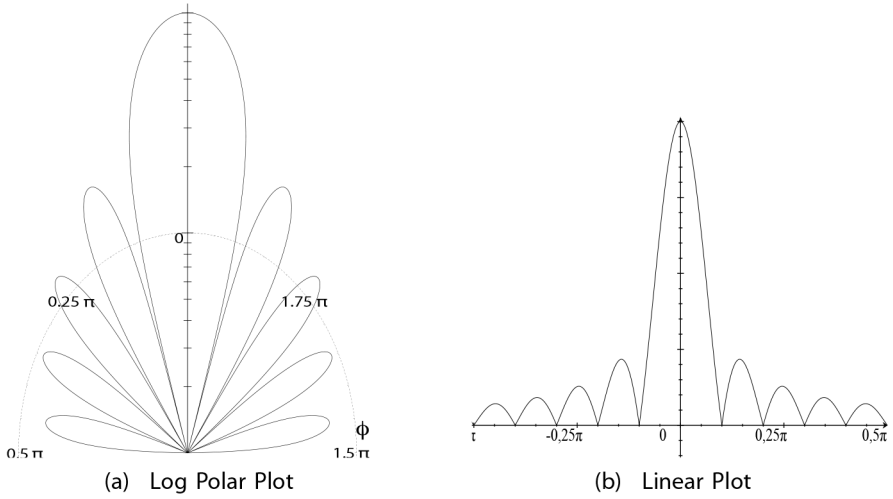


FIGURE 1.3:
Antenna Pattern (Woods et al., 2010, Appendix B.1, Fig. B.2)

For doing such observations radio telescopes are used, these are actually receiving antennas, which basically are devices that respond to incoming electromagnetic radiation and output electrical currents related to this response. The effective receiving area of an antenna is known as its aperture, when a planar electromagnetic wave enters an aperture, it is distorted in what is called a diffraction pattern (Fig. 1.2) (Woods et al., 2010, Appendix B.1). Therefore a finite sized aperture cannot

correctly record the radio brightness without some distortion of the original signal (Woods et al., 2010, Appendix B.1). The diffraction distortion is due to the interaction of the original EM wave with the edges of a finite sized aperture, which creates the fringe pattern of destructive and constructive interference (Woods et al., 2010, Appendix B.1). Diffraction effects all types of EM waves when entering an aperture, but is more severe for longer wavelengths. The distance to the first zero of the diffraction pattern of a circular aperture is given by

$$R \simeq 1.22\frac{\lambda}{D} \tag{1.1}$$

If two objects are closer than the first minima (1.4(a)), for a particular aperture, they cannot be

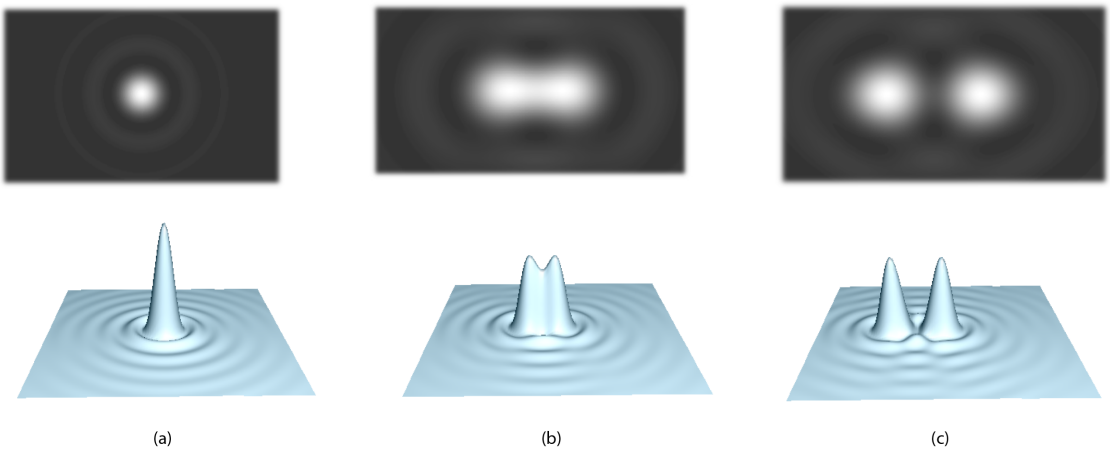


FIGURE 1.4:
Angular Resolution (Woods et al., 2010, Appendix B.1, Fig. B.3)

distinguished (Woods et al., 2010, Appendix B.1). This is Rayleigh's criterion. Therefore, the first minimum determines the resolving capabilities of an aperture and is called the angular resolution R (Woods et al., 2010, Appendix B.1). The angular resolution, represented in the right-hand side of the equation 1.1, depends on both the wavelength, λ and the aperture diameter, D (Woods et al., 2010, Appendix B.1).

Radio waves have a long wavelength. Hence, radio astronomy requires large telescopes for improved resolution. This results in detailed images (radio brightness readings) (Woods et al., 2010, Appendix B.1). The dimensions of a single dish radio telescope, needed to match the angular resolution requirements of an optical telescope, are extremely impractical. This is so both in terms of engineering precision required and physical constraints (Thompson et al., 2008, CH. 5 pg 124-125). For example, to achieve the same angular resolution as the naked human eye, a radio antenna's aperture observing a source at $1.4\,\mathrm{GHz}$ must be $750\,\mathrm{m}$ in diameter (Woods et al., 2010,

Appendix B.1).

This issue can be resolved by using a radio interferometer setup. This can be described as the use of an array of multiple radio telescopes working in phase. The effective aperture diameter can be mathematically proven to correspond to the largest separation of any 2 telescopes in the array. With this setup, a resolution comparable to a large aperture diameter telescope can be obtained more easily, as it is more practical and feasible to build from an engineering point of view (Woods et al., 2010, Appendix B.1). This is from that fact that the technique is coined *Aperture Synthesis*. It was introduced by Sir Martin Ryle, as with that setup the aperture diameter and in consequence the resolution of a much larger telescope can be emulated by the aforementioned means. Knowing the impulse response of the aperture, a closer reconstruction of the original source can be made by performing a deconvolution (Woods et al., 2010, Appendix B.1).

FIGURE 1.5:
Sir Martin Ryle[2]

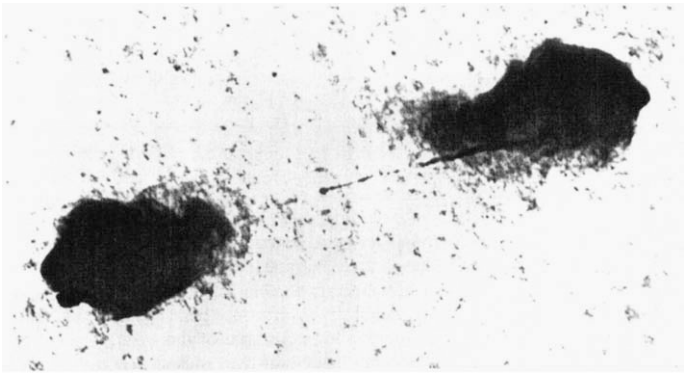


FIGURE 1.6:
Radio image of Cygnus A made with the VLA at 4.9 GHz by Perley, Dreher, and Cowan (1984) (Thompson et al., 2008, Pg. 33, Fig. 1.18)

1.2 Software Defined Radio

Software Defined Radio (SDR) is the use of fully general purpose computing hardware such as CPUs for example to perform tasks related to observations done with radio telescopes/antennas which would otherwise be done by specialised hardware e.g mixers, filters, data-acquisition hardware, and signal processing devices Wikipedia (2014).

[2]Stamp, Sir Martin Ryle - Radio Surveyor of the Universe.
http://colnect.com/en/stamps/stamp/184650-Sir_Martin_Ryle_-_Radio_Surveyor_of_the_Universe-Eminent_Britons-United_Kingdom_of_Great_Britain_Northern_Ireland

Mauritius has an early track record in SDR with the MRT.

From the 2000s to the 2010s, some work has been sparsely done around software defined radio Ginourie (2010), worked on a front-end and back-end receiver system, Pirthee (2013) worked on a digital backend for the MITRA. As concerns software correlators Jheengut (2008) worked on the implementation of a lag XF software correlator in FORTRAN code and pointed out the computationally intensive nature of the operation for large data sets. Platel (2010) worked on the implementation of an FX software correlator on Nvidia 9400 GT GPU in CUDA-C language, a 2 hour observation was done at 151.7 MHz with 4 log periodic dipole antenna oriented E-W 4 other oriented N-S, the frequency was downconverted to 30 MHz using an heterodyne receiver system. The data acquisition was done with a 16-bit ADC sampling at 1 MHz, with 4-bits for specifying the channels and 11-bits for quantisation and 1-bit for the sign. Platel (2010) pointed out the effects of undersampling on the correlated data.

1.3 Digital Systems for the MITRA

The Multifrequency Interferometry Telescope for Radio Astronomy (MITRA[3]) is a joint project between Mauritius and South Africa to build a low frequency array telescope in the range of 200 MHz to 800 MHz (Beeharry et al., 2013, Ingala and MacPherson, 2013). It is a sensitive high resolution multi-frequency dual polarity instrument using multiple stations of low-cost dipoles, such as log-periodic antennas (Beeharry et al., 2013, Ingala and MacPherson, 2013). Interestingly it is aimed at being an interstation project where baselines will range from a few metres, in one station, to some 250-500-1000-3000-5000 km for the whole instrument (Beeharry et al., 2013, Ingala and MacPherson, 2013). In the scope of this dissertation the main aims of interest for the MITRA are, the near real time Very Long Baseline Interferometry (VLBI) and Electronic-VLBI, Digital Data processing and particularly High Capacity correlation (Beeharry et al., 2013, Ingala and MacPherson, 2013).

Thus this project/dissertation focuses on the development & testing of software, and general purpose computing hardware in relation to the MITRA. In so doing, it will pave the way for a higher integration of digital systems, as opposed to specialised hardware, in the synthesis imaging process. The particular application, which was focused on, was that of high performance software correlation by the use of parallel computing. The central objective was to implement the DiFX

[3]MITRA means friend in Sanskrit (uomnews.wordpress.com, 2011)

(Distributed FX) software by Deller et al. (2007) for the MITRA by making it an off-the-shelf, user-friendly and custom documented software.

I hope that the work on this particular project will not stay still and that this dissertation will inspire other people to continue the work. From that initiative follows naturally the creation of a public git repository: `https://github.com/Benzy-Louis/MITRA_FX-CPU-GPU.git`, where those interested to contribute to the project can continue the development, work on the source code, and improve the software and its documentation on their own or contribute for the same to the repository. Those interested are freely encouraged to mail me at louis.ruben@gmail.com, if they want to contribute or have questions, suggestions for the project or this dissertation.

1.4 Outline

This dissertation is composed of 5 main chapters. It is structured as follows *chapter 1* is a general introduction on the objectives of the project/dissertation. *Chapter 2* is a consise development on the topic of synthesis imaging in radio astronomy; which should give an appropriate basis both to understand the results of observations, and also the underneath process involved before getting data that can be more easily interpreted from the scientific point of view. *Chapter 3* is about the DiFX software and its implementation in the scope of this project. *Chapter 4* is a discussion about application of GPGPU in software correlators. Finally *chapter 5* is a conclusion of the project/dissertation and hints at the enormous work which has yet to be done before an actual, highly performant, and operational, software correlator, can be implemented in the scope of the ongoing MITRA project.

Chapter 2

Synthesis Imaging

In the scope of the MITRA project, a concise development of the theoretical background related to imaging in radio astronomy in relation to digital systems is done.

2.1 Principles of Radio Interferometry

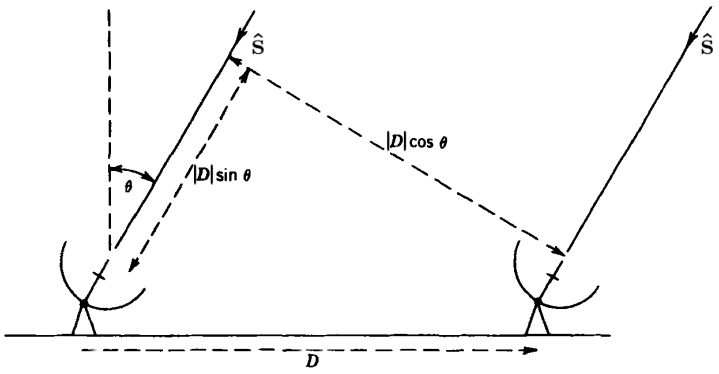


FIGURE 2.1:
Geometry of an elementary interferometer (Thompson et al., 2008, Pg. 51, Fig. 2.1)

Consider the figure 2.1 which shows an incoming electromagnetic planar wavefronts traveling in the direction, $\hat{\mathbf{s}}$, received by both antennas. Depending on the direction of the wavefront, relative to the direction the antennas point to, here characterised by the angle θ, the same wavefront will reach a particular antenna first. Here the right one, or both would get the same wavefront at the same time if $\theta = 0$. In the first case, it is only after a certain lapse of time that the wavefront reaches the other antenna. The extra distance the wavefront has to travel to reach the left antenna is the projection of the baseline direction on the unit vector in the source direction which is equal to $\mathbf{D} \cdot \hat{\mathbf{s}}$ (Thompson et al., 2008, Sec. 2.1).

$$\mathbf{D} \cdot \hat{\mathbf{s}} = |\mathbf{D}| \cdot \cos(90^\circ - \theta) = |\mathbf{D}| \sin(\theta) \tag{2.1}$$

This is also the extra time that the wavefront takes, at the speed of light, to reach the left antenna. It is known as the geometric delay or time delay. The geometric delay, τ_g is thus the following:

$$\tau_g = \frac{\mathbf{D} \cdot \hat{\mathbf{s}}}{\mathrm{c}} \tag{2.2}$$

where, c, is the speed of light in vacuum.

Thus a similar output is obtained by both antenna differing mainly by the geometric time delay.

Consider the figure 2.2. Suppose that the antennas track the source under observation, which

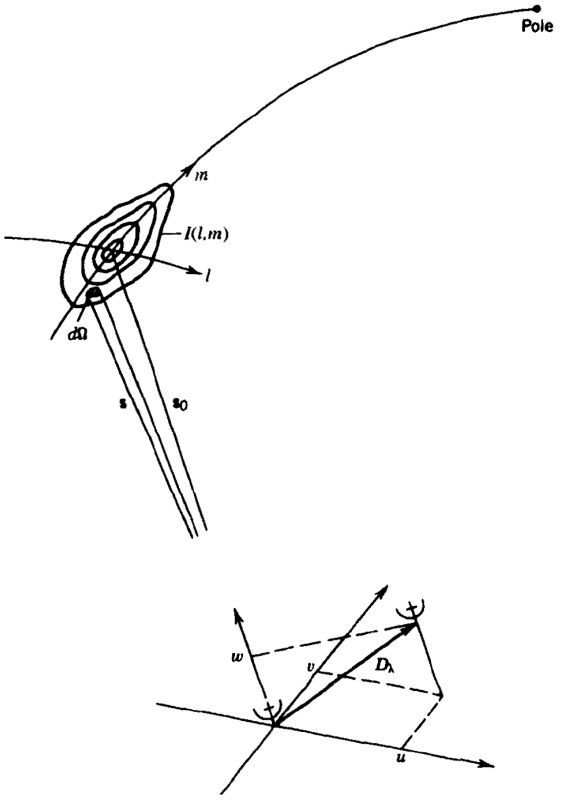


FIGURE 2.2:
Free coordinate system (Thompson et al., 2008, Pg. 70, Fig. 3.2)

is a common situation with parabolic dish antennas. Let the unit vector $\hat{\mathbf{s}}_\mathbf{0}$ indicate the phase reference position. This position, known as the phase-tracking centre, becomes the centre of the field to be mapped. The magnitude of the baseline vector, $|\mathbf{D}_\lambda| = \frac{|\mathbf{D}|}{\lambda}$, is measured in wavelengths at the centre frequency of the observing band, and the baseline direction, $\mathbf{D}_\lambda$, has components (u, v, w) in a right-handed coordinate system, where u and v are measured in a plane normal to the direction of the phase reference position (Thompson et al., 2008, CH 3). The spacing component v is measured toward the north as defined by the plane through the origin, the source, and the pole, and u toward the east (Thompson et al., 2008, CH 3). The component w is measured in the

direction $\hat{\mathbf{s}}_\mathbf{0}$ and so is defined as follows,

$$\mathbf{D}_\lambda \cdot \hat{\mathbf{s}}_0 = w \tag{2.3}$$

Source direction or position on the celestial sphere have the components (l, m, n) which are simply, respectively the direction cosines of the components of the particular direction/position $\hat{\mathbf{s}}$ in terms of the (u, v, w) components i.e.

$$\hat{\mathbf{s}} = (l, m, n) = (\hat{\mathbf{s}} \cdot \hat{\mathbf{u}}, \hat{\mathbf{s}} \cdot \hat{\mathbf{v}}, \hat{\mathbf{s}} \cdot \hat{\mathbf{w}}) \tag{2.4}$$

Then, since $l^2 + m^2 + n^2 = 1$ we can re-express n in terms of l and m as follows,

$$n = \sqrt{1 - l^2 - m^2} \tag{2.5}$$

leading us to re-express source directions with the following components i.e.

$$\hat{\mathbf{s}} = (l, m, \sqrt{1 - l^2 - m^2}) \tag{2.6}$$

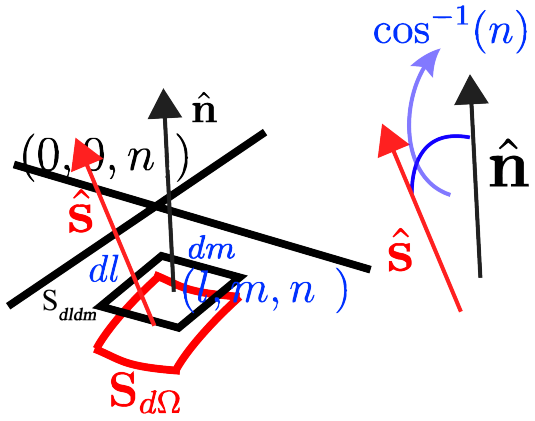


FIGURE 2.3:
Element of Solid Angle

One can also easily show that for an elementary patch of the sky $dl \cdot dm$, the solid angle, $d\Omega$, subtended from our reference is,

$$d\Omega = \frac{dldm}{n} \tag{2.7}$$

Refer to figure 2.3, note that the solid angle, $d\Omega$ is just the projection of the patch $dl \cdot dm$ on the unit sphere centered on the origin (Thompson et al., 2008, CH 3). Referring to the figure the elementary surface at coordinate $(u = l, v = m, n = n)$ is denoted as $\mathbf{S}_{dldm} = S_{dldm} \cdot \hat{\mathbf{n}}$ and its

projection on the unit sphere as, $\mathbf{S}_{d\Omega} = S_{d\Omega} \cdot \hat{\mathbf{s}}$. So we have the following relationship,

$$\mathbf{S}_{d\Omega} \cdot \hat{\mathbf{n}} = S_{dldm} \tag{2.8}$$

$$S_{d\Omega} \cdot \hat{\mathbf{s}} \cdot \hat{\mathbf{n}} = S_{dldm} \tag{2.9}$$

As the angle between $\hat{\mathbf{s}}$ and $\hat{\mathbf{n}}$ is $\cos^{-1}(n)$

$$S_{d\Omega} \cdot \cos(cos^{-1}(n)) = S_{dldm} \tag{2.10}$$

$$S_{d\Omega} \cdot n = S_{dldm}$$

$$S_{d\Omega} = \frac{S_{dldm}}{n} \tag{2.11}$$

$$d\Omega = \frac{dldm}{n}$$

$$d\Omega = \frac{dldm}{\sqrt{1 - l^2 - m^2}} \tag{2.12}$$

2.1.1 Visibility – Sky intensity distribution relationship

There is a direct relationship between the electrical signal received at the pair of antennas and the electromagnetic waves from the sky. Most importantly the visibility is a function of the baseline vector, $\mathbf{D}_\lambda$. It relates to the sky intensity distribution in the following way (Thompson et al., 2008, Sec. 3.1, Pg. 71-73) ,

$$V'(u,v,w) = \iint\limits_{\text{sky patch}} A_N I(l,m) e^{-j2\pi(\mathbf{D}_\lambda \cdot \hat{\mathbf{s}})} d\Omega \tag{2.13}$$

$$V'(u,v,w) = \iint\limits_{\text{sky patch}} \frac{A_N I(l,m) e^{-j2\pi(ul+vm+w(\sqrt{1-l^2-m^2}-1))}}{\sqrt{1 - l^2 - m^2}} dldm \tag{2.14}$$

where, $V'(u,v,w)$ is the measured visibility, A_N, the normalised product of the antenna beams, and $I(l,m)$, is the intensity distribution.

According to the paraxial approximation that is valid so long as the synthesised field is not too large. If l and m are small enough that the term

$$\left(\sqrt{(1-l^2-m^2)}-1\right) w \simeq -\frac{1}{2}(l^2+m^2)w \tag{2.15}$$

can be neglected then what is left is the following:

$$V'(u,v) = \iint\limits_{\text{sky patch}} \frac{A_N I(l,m) e^{-j2\pi(ul+vm)}}{\sqrt{1-l^2-m^2}} dldm \tag{2.16}$$

which is a direct Fourier transform relationship between visibility map and sky intensity distribution multiplied by the normalised antenna pattern.

2.2 The Van Cittert-Zernike theorem

The true visibility, $V(u,v)$ can be expressed as follows (Francesco, 2014, Slide 8):

$$V(u,v) = |V|e^{-j\phi} = \iint I(l,m) e^{-j2\pi(ul+vm)} dldm \tag{2.17}$$

absorbing terms from eq. 2.16 in the term for the intensity distribution, $I(l,m)$

The Fourier transform relationship between the true visibility, $V(u,v)$, and the sky intensity distribution, $I(l,m)$, is the Van Cittert-Zernike theorem on which synthesis imaging is based. This means that we can recover $I(l,m)$ from $V(u,v)$:

$$V(u,v) = \iint I(l,m) e^{-2j\pi(ul+vm)} dldm \tag{2.18}$$

$$I(l,m) = \iint V(u,v) e^{2j\pi(ul+vm)} dudv \tag{2.19}$$

In eq. 2.17, the visibility is expressed as, $|V|e^{-j\phi}$. The phase, ϕ, as portrayed by figure 2.5,

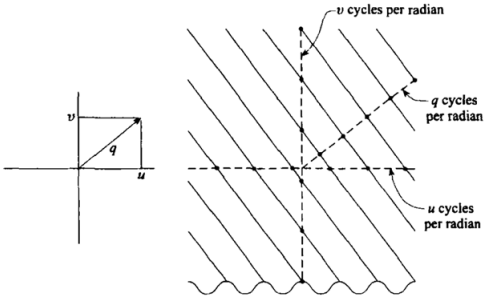


FIGURE 2.4: Fringe Visibility (Thompson et al., 2008, Pg. 64, Fig. 2.7)

contains information about the location of structure with spatial frequency (u, v) relative to the phase centre $(l = 0, m = 0)$, and the amplitude, $|V|$, gives information about how much of the spatial frequency component is present (Francesco, 2014, Slide 8) (Wilner, 2014, Slide 9).

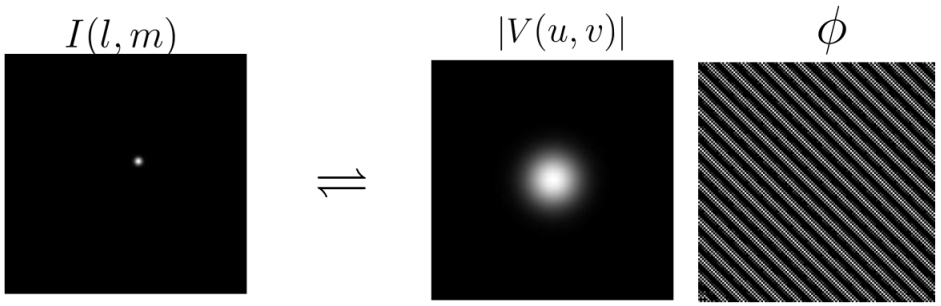


FIGURE 2.5: Amplitude and fringe of the Visibility function source offset from the phase centre in the upper-right direction (Wilner, 2014, Slide 9)

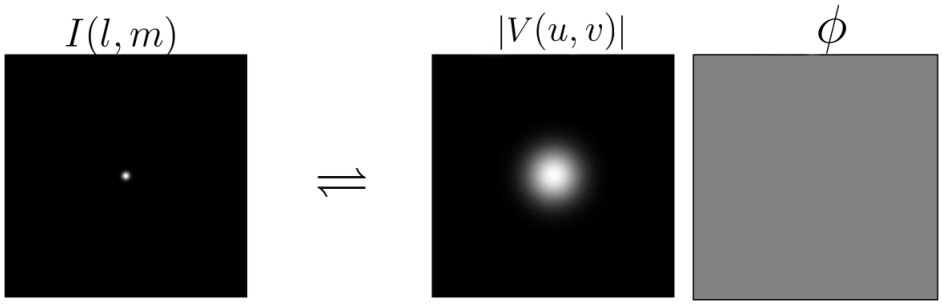


FIGURE 2.6: Amplitude and fringe of the visibility function, source at phase centre (Wilner, 2014, Slide 9)

2.3 The correlator

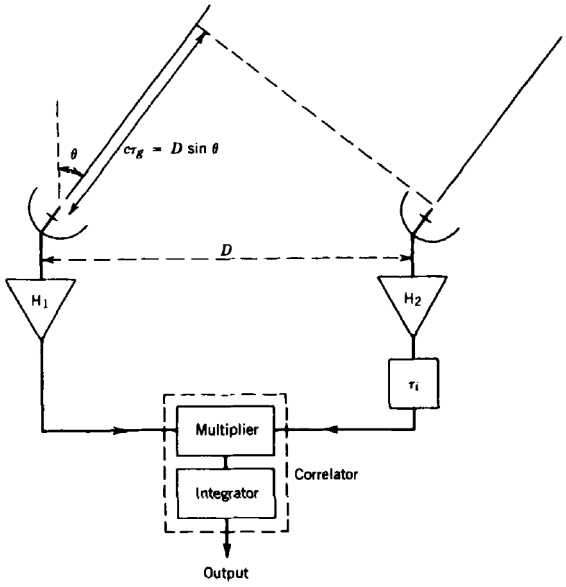


FIGURE 2.7:
A simplistic representation of a correlator (Thompson et al., 2008, Pg. 53, Fig. 2.3)

A correlator is needed to sample the visibility values. It does the following things as illustrated in figure 2.7 it multiplies the signals together and integrates over a certain interval of time called the integration time. Now assume that, for a point source, each antenna delivers the same signal. That is, the voltage, $V(t)$, to the correlator and that one lags the other by a time delay, $\tau - \tau_i$, the output of the correlator may be a voltage, a current, or a coded set of logic levels (for a digital system), but in any case it represents a physical quantity with the dimensions of voltage squared (Thompson et al., 2008, Sec 2.2). We can express it by using the following equation 2.20,

$$r(\tau) = \lim_{T\to\infty} \frac{1}{2T} \int_{-T}^{T} V(t)V(t-\tau)dt \tag{2.20}$$

This is actually an autocorrelation function. The signal from a natural cosmic source can be considered as a continuous random process that results in a broad spectrum, which has phases with a random function of frequency.

Assume that the time-averaged amplitude of the cosmic signal in any finite band, is constant, with frequency, ν, over the passband of the receiver. The squared amplitude of a frequency spectrum is known as the power density spectrum, or power spectrum. The power spectrum of a signal is the Fourier transform of the autocorrelation function of that signal (Thompson et al., 2008, Sec. 3.3). This statement is known as the Wiener-Khinchin relation (Thompson et al., 2008, Sec. 3.3). It can

be written as the following,

$$|H(\nu)|^2 = \int_{-\infty}^{\infty} r(\tau) e^{-j2\pi\nu\tau} d\tau \tag{2.21}$$

and its pair,

$$r(\tau) = \int_{-\infty}^{\infty} |H(\nu)|^2 e^{j2\pi\nu\tau} d\nu \tag{2.22}$$

It is also useful to examine the corresponding relation for the **cross-correlation function** of two different waveforms (Thompson et al., 2008, Sec. 3.3). The response of a correlator, as used in a radio interferometer, can thus be written as

$$r(\tau) = \lim_{T\to\infty} \frac{1}{2T} \int_{-T}^{T} V_1(t) V_2{}^*(t-\tau) dt \tag{2.23}$$

for which we have the pentagram as a short hand notation, as follows,

$$V_1(t) \star V_2(t) = \lim_{T\to\infty} \frac{1}{2T} \int_{-T}^{T} V_1(t) V_2{}^*(t-\tau) dt \tag{2.24}$$

From the convolution theorem one can very easily derive the following relationship (Thompson et al., 2008, Sec. 3.3),

$$V_1(t) \star V_2(t) \rightleftharpoons \widehat{V}_1(\nu) \cdot \widehat{V}_2^*(\nu) \tag{2.25}$$

In all the previous and following equations, the superscript asterisk * denotes complex conjugation and, $\rightleftharpoons$, denotes Fourier transform.

Now let's get back to equation 2.23, τ, is the time by which voltage V_2 is delayed with respect to voltage V_1. The functions V_1, and V_2 that represent the signals in equation 2.23 may be complex.

The output of a single multiplying device is a real voltage or number though (Thompson et al., 2008, Sec. 3.3). To obtain the complex cross-correlation, which represents both the amplitude and the phase of the visibility, one can record the fringe oscillations and measure their phase, or use a complex correlator which contains two multiplying circuits (Thompson et al., 2008, Sec. 3.3).

For the case where the antennas track the source, both the antenna beam center and the center of the source are at the (l, m) origin, the correlator output can thus also be expressed as the following,

$$r(\tau) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} I(l,m) A(l,m) |H(\nu)|^2 e^{j2\pi\nu\tau} dl dm d\nu \tag{2.26}$$

For a wavefront incident from the direction (l, m), the difference in propagation times through the two antennas to the correlator results from a difference in path lengths of $(ul + vm)$ wavelengths, where an approximation has been made (refer to the statement before and after Eq. 2.15). The corresponding time difference is $\frac{(ul+vn)}{\nu}$ (Thompson et al., 2008, Sec. 3.3). If we take as V_1, the signal from the antenna for which the path length is the greater (for positive l and m), then from equation 2.26, the correlator output becomes

$$r(\tau) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} I(l,m)A(l,m)|H(\nu)|^2 e^{-j2\pi(lu+mv)} dldmd\nu \tag{2.27}$$

Assuming that the intensity and the antenna pattern are constant over the bandpass range of the filters, and the width of the source is small compared with the antenna beam. The correlator output then becomes

$$\begin{aligned} r &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} I(l,m)A(l,m)dldm \int_{-\infty}^{\infty} |H(\nu)|^2 e^{-j2\pi(lu+mv)} d\nu & (2.28)\\ &= A_0 V(u,v) \int_{-\infty}^{\infty} |H(\nu)|^2 e^{-j2\pi(lu+mv)} d\nu & (2.29) \end{aligned}$$

where A_0 is the collecting area of the antennas in the direction of the maximum beam response and $V(u, v)$ the true visibility, the measured visibility, $V'(u, v)$ was introduced in sec. 2.1.1 in the equation 2.16 (Thompson et al., 2008, Sec. 3.3). The filter response $H(\nu)$ is a dimensionless (gain) quantity, note that we assume that the antennas are identical and that their filter response are identical such that $H_1(\nu) = H_2(\nu) = H(\nu)$ (Thompson et al., 2008, Sec. 3.3). If the filter response is essentially constant over a bandwidth, A_0, eq. 2.29 becomes

$$r = A_0 V(u,v)\Delta\nu \tag{2.30}$$

Thus we have here the visibility $V(u, v)$ which has units of W m^2 Hz^{-1}, A_0 has units of m^2, and $\Delta\nu$ has units of Hz. This is consistent with r, the output of the correlator, which is proportional to the correlated component of the received power (Thompson et al., 2008, Sec. 3.3).

2.4 Digital Correlators

2.4.1 FX Correlators

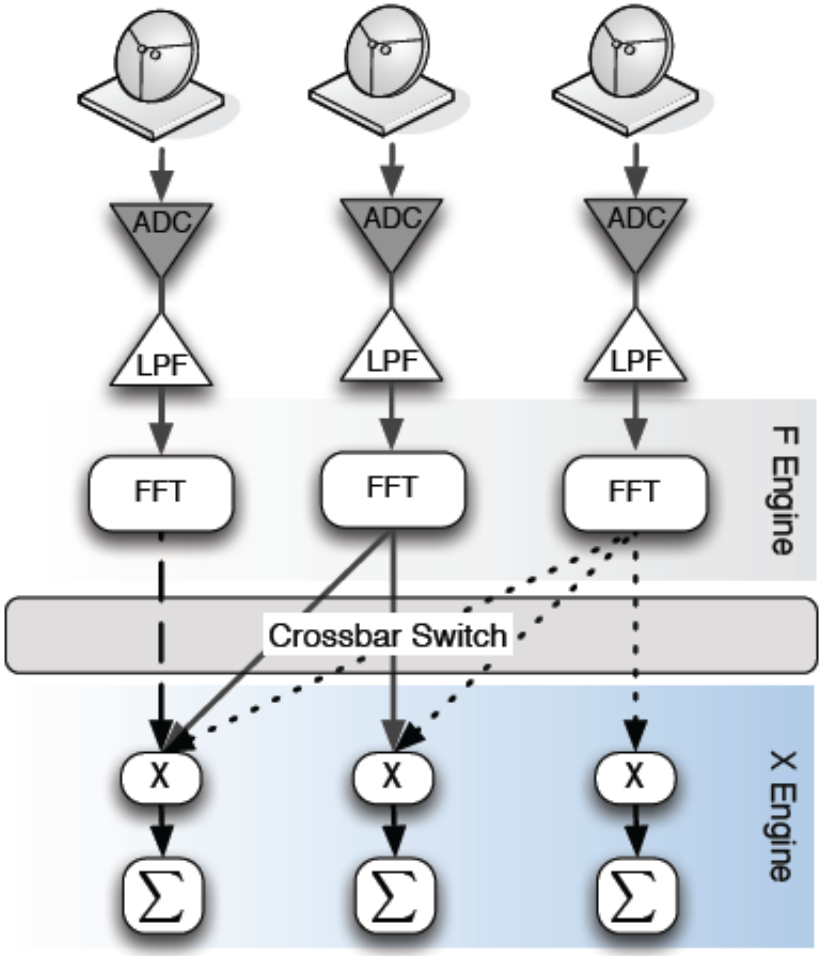


FIGURE 2.8:
FX correlator (Woods et al., 2010, Sec. 2.3.1)

A digital FX complex correlator is one in which the Fourier transformation to frequency domain is performed before cross multiplication of data from different antennas (Thompson et al., 2008, Sec 8.7),. In such a correlator the input bit stream from each antenna here with index k, $V_k(t)$, is converted to a frequency spectrum,$\widehat{V}_k(\nu)$, by a real time FFT (Thompson et al., 2008, Sec 8.7),

$$\widehat{V}_k[a_n, \nu] = \sum_{l=0}^{L-1} V_k[l] \mathrm{e}^{-i2\pi\nu l/L} \tag{2.31}$$

and then for each antenna pair (i,j), the complex amplitudes for each frequency, ν_m, at a time index, a_n, are multiplied to produce the cross power spectrum (Thompson et al., 2008, Sec

8.7) (Woods et al., 2010),

$$c_{ij}[a_n, \nu_m] = \widehat{V}_i[a_n, \nu_m]\widehat{V}_j^*[a_n, \nu_m] \tag{2.32}$$

and accumulated over all the, A time steps, to improve the signal to noise ratio (SNR) (Woods et al., 2010),

$$C_{ij}[A, v_m] = \sum_{a=0}^{A-1} \widehat{V}_i[a_n, \nu_m]\widehat{V}_j^*[a_n, \nu_m] \tag{2.33}$$

Chapter 3

Distributed FX Correlation - The DiFX software

In this chapter, the work in relation to the DiFX program is explained, and the results analysed.

3.1 Introduction

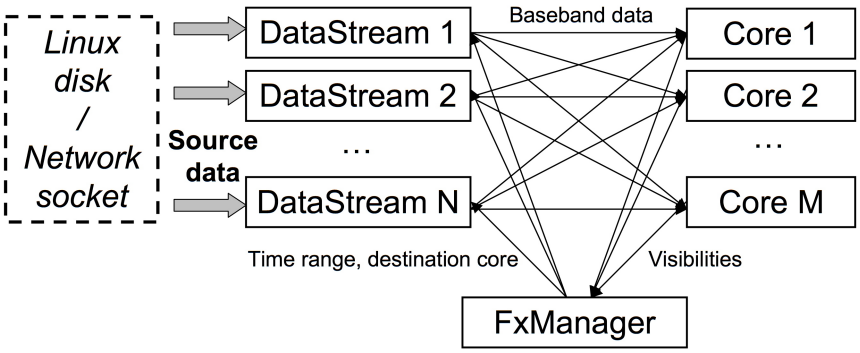


FIGURE 3.1: DiFX a distributed FX correlator (Deller et al., 2007)

The DiFX software by Deller et al. (2007), is a software FX correlator written in C++, which functions using the Fourier transform and correlation, as explained in sec. 2.4.1. Its main specificity is distributed computing, which is the use of multiple nodes/general purpose computing machines such as conventional CPUs to perform correlation tasks. As the computation of cross-correlation values, C_{ij} as in eq. 2.33 can be performed on a baseline independent scheme, this can be adapted to distributed computing approaches. However, even though the computation is the same, this implies data communication, of the different stations baseband data, between the nodes. This

communication is done using a Message Parsing Interface (MPI), through which data communication can be managed. As described by Deller et al. (2007), and as illustrated in fig. 3.1, MPI allows us to set up a main node, called the FxManager. The latter directs specific time range of streamed station data to processing cores. The data processed is then sent back after computation to the Master node for accumulation.

In this project, DiFX version 2.4 was used. It requires a GNU/Linux operating system. It also needs an MPI installation on the Ubuntu 14.04 LTS OS distribution. It works with both the 32-bit and 64-bit OS. different configurations, with Intel CPUs, is listed in table 3.1. As already implemented in the software, for optimised computation of vector operations, the Intel Performance Primitive library was used. The main investigation turned around exploring the possibilities offered by the software and its performance.

TABLE 3.1: Setups used

Index	Setup info	No. of Cores	Thermal Design Power[a]	Estimated Cpu Cost[a]
1	i7-2600K 3.4 GHz Ubuntu 14.04 LTS VMware 32-bit	4	95 W	$ 317
2	i3-3240 3.4 GHz Ubuntu 14.04 LTS 64-bit	2	55 W	$ 147
3	C2Quad-Q8200 2.33 GHz Ubuntu 14.04 LTS 64-bit	4	95 W	$ 66
4	Core2Duo E4600 2.4 GHz Ubuntu 14.04 LTS 64-bit	2	65 W	$ 40
5	Pentium 4 HT 630 3.0 GHz Ubuntu 14.04 LTS 32-bit	1	84 W	$ 12

[a]From the ark.intel.com database

3.2 Working with datasets

The approach taken in this project was to work with the datasets provided and familiarise with the workflow used. The basic procedure for software installation is explained in the appendix A.1. Available data, at http://cira.ivec.org/dokuwiki/doku.php/difx/datasets, was used to test the installation. Following on table 3.2 is the info provided about the data set.

TABLE 3.2: Data sets

Name	Size	Data info	Observed source name
rdv70	5.8 GB	6 Stations - RCP(IEEE) polarised 2333MHz - 8903MHZ (8x8MHz bands) 50 secs of Observertation 128 Mbps data 3 VLBA (.vlba) + 3 Mark4 (.mark4) data format	4C39.25
v252f	3.8 GB	6 Stations RCP(IEEE) polarised and LCP(IEEE) polarised 8425MHz - 8441MHZ (2POLx16MHz bands) 8409MHz - 8425MHZ (2POLx16MHz bands) 50 secs of Observertation 256 Mbps data 5 LBA style (.lba) + 1 Mk5a (.m5a) data format	0208-512
K08161	2.6 GB	2 Stations - RCP(IEEE) polarised 2.3GHz - 8.2GHZ (16x16MHz bands) 40 secs of Observertation 256 Mbps data 1 EVN (.evn) + 1 Mk5a (.m5a) data format	0955+476
tc016a.pulsar	5.78 GB	3 Stations RCP(IEEE) polarised and LCP(IEEE) polarised 1642MHz - 1674MHZ (2POLx4x8MHz bands) 60 secs of Observertation 256 Mbps data 3 Mk5a (.m5a) data format	J1645-0317

3.2.1 The workflow

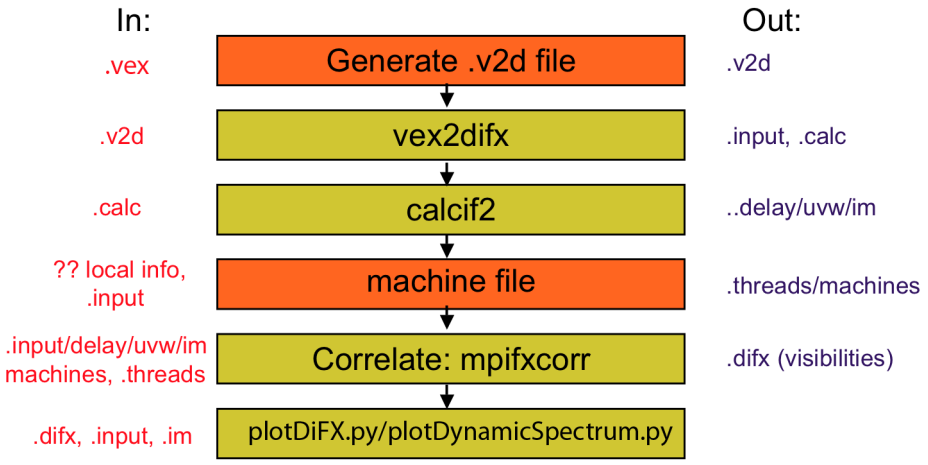


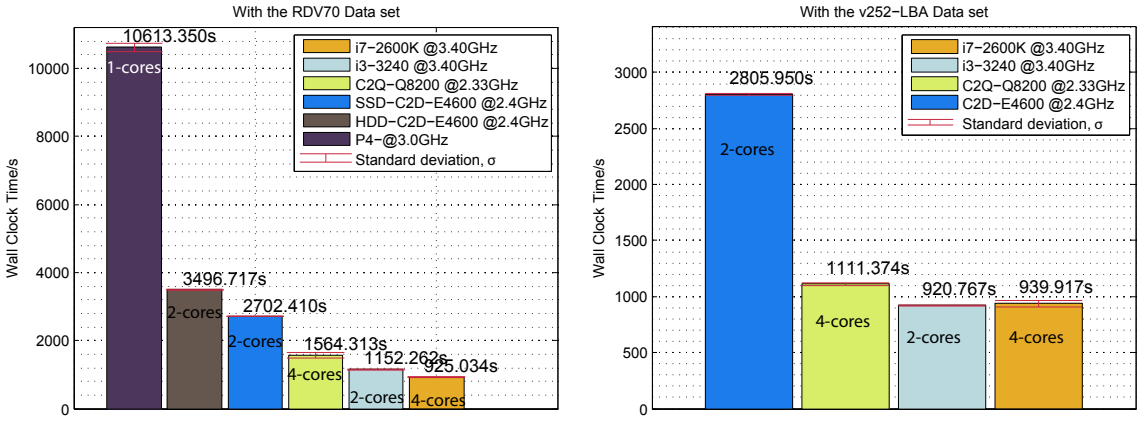
FIGURE 3.2: DiFX the correlation workflow[1] (Deller, 2014)

Generally most of the datasets are provided with configuration files .v2d files and .vex files (Deller, 2014) . Fig. 3.2 shows the main steps to be performed from a working setup to perform a correlation job. In the .v2d files (vex to difx) the telescopes used and their data file directories is specified, along with the .vex file. Inside the .vex are various metadata, about the telescopes, the observations done, the sources information, and the way the data is recorded, are specified (Deller, 2014). Using the .v2d file as input to vex2difx, generates .calc and .input files (Deller, 2014). In the .calc file information about the earth orientation parameters and the antenna configuration is generated (Deller, 2014). This is processed by the calcif2 routine which models the antenna delay, and baseline information in .im files (Deller, 2014). If the the correlation is to be done with separate nodes, the usual machine files, as used in MPI, must be created, for a two node setup the content can be as simple as follows:

```
mrt@172.37.22.12 slots=4
mrt@172.37.22.13 slots=4
```

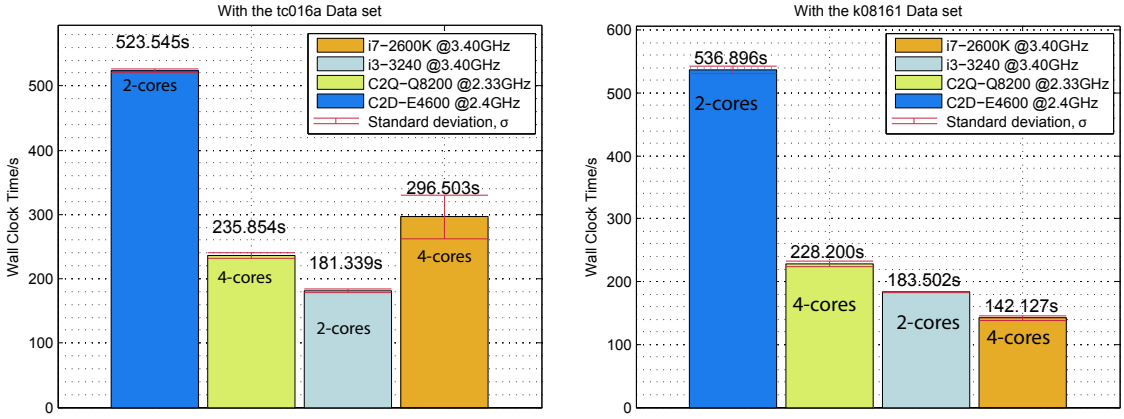
The login name, the IP address of the nodes is specified and there the "slots" indicate the number of processes to be used. Then, the mpifxcorr program can be used by the appropriate mpirun command. It is the mpifxcorr program which is actually performing the correlation task, as illustrated in fig 3.1. To obtain useful log information, the errormon2 program is used. It outputs useful information about the stage of the correlation reached. It also outputs a wall clock time, for which the correlation task has been run (Deller, 2014). As main output, raw visibility files are obtained in a folder with the .difx extension (Deller, 2014).

To evaluate how well the program performs, the correlation tasks on the different data sets has been run on the configurations displayed in fig. 3.3 and fig. 3.4. A notable observation when we take as reference the time run of the single-core setup the Pentium 4 HT 630 @3GHz (fig. 3.5a), is that the data sets with the higher number of baselines such as the RDV70 data is processed faster on the 4-cores CPUs, the i7-2600K ($\sim$11.5x speed up), and C2Q-Q8200 ($\sim$6.8x speed up) or the latest generation CPUs such as the i3-3240 ($\sim$9.2x speed up) which outperforms the C2Q-Q8200 of a much older generation.



(A) DiFX - RDV70 runtime (B) DiFX - V252-LBA runtime

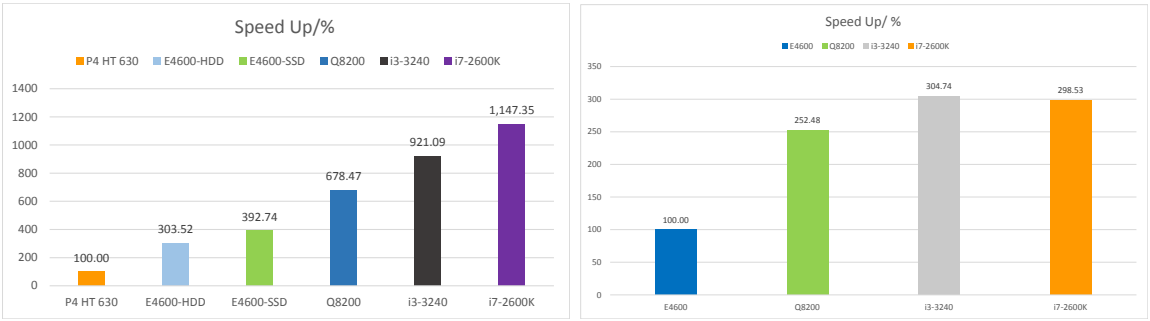
FIGURE 3.3: DiFX Run Time



(A) DiFX - TC016A runtime (B) DiFX - Geodesy K08161 runtime

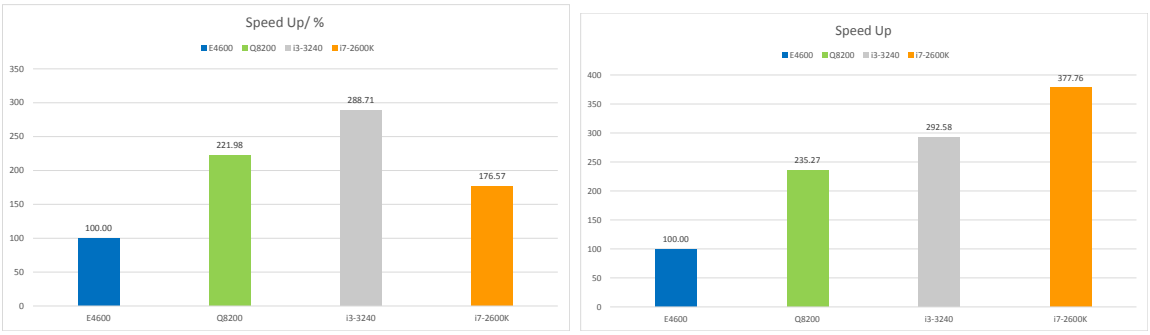
FIGURE 3.4: DiFX Run Time

Also on the same figure (fig. 3.5a) for the RDV70 data set the C2D-E4600 was run with a regular 7200 rpm hard disk drive (HDD) and also a Solid State Drive (SSD), the Kingston V300 120 GB, where the read/write speed as per the specifications is roughly 3 times that of the HDD as concerns the SSD. The higher speedup of the configuration (C2D-E4600-SSD $\sim$ 3.9x) compared with the one with a regular HDD (C2D-E4600-HDD $\sim$ 3.0x) suggests that data read and write speeds can effectively influence the performance.



(A) DiFX - RDV70 (B) DiFX - V252-LBA

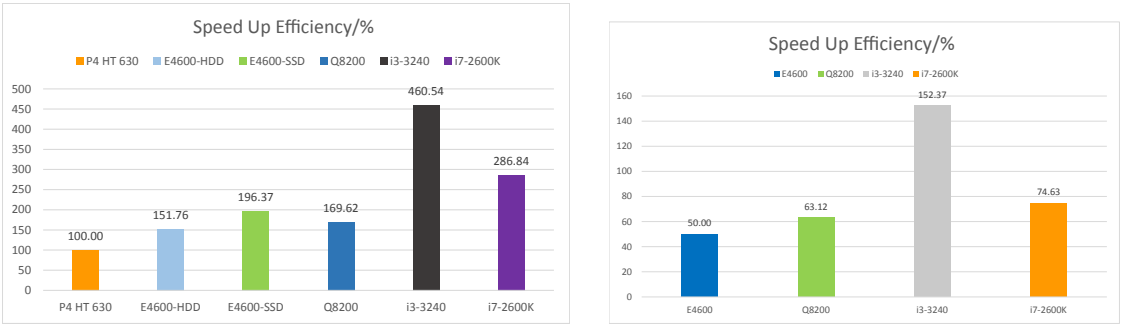
FIGURE 3.5: DiFX Speed Up



(A) DiFX - TC016A (B) DiFX - Geodesy K08161

FIGURE 3.6: DiFX Speed Up

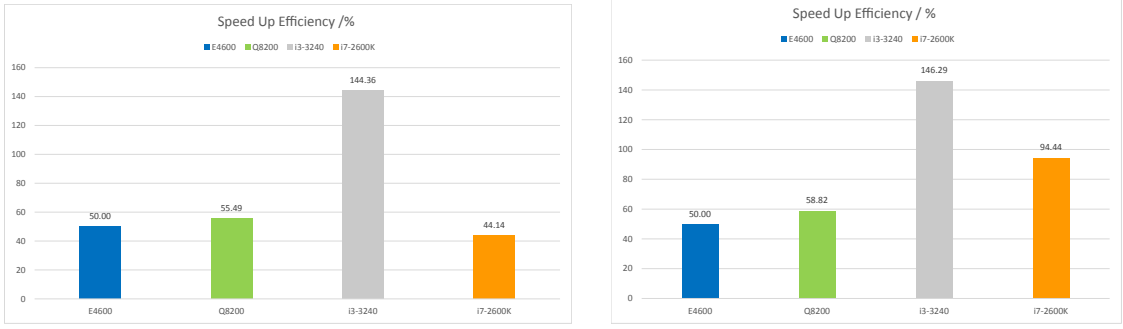
One would expect this trend to to maintained for the V252-LBA data (fig. 3.5b) consisting of 6 stations, but this is actually not the case, possibly due to the lower number of frequency channels. For the other data sets with the C2D-E4600 @2.4 GHz as reference, the hierarchy between higher end CPU is more evident for the K08161 data set (fig 3.4b), which could be accounted for the greater number of channels. For the tc016a data set (fig. 3.6a) the performance of the higher end CPU would not be evident if we had not considered the fact that it was running on a virtual machine (VMWare 7.0 software).



(A) DiFX - RDV70 (B) DiFX - V252-LBA

FIGURE 3.7: DiFX Speed Up efficiency

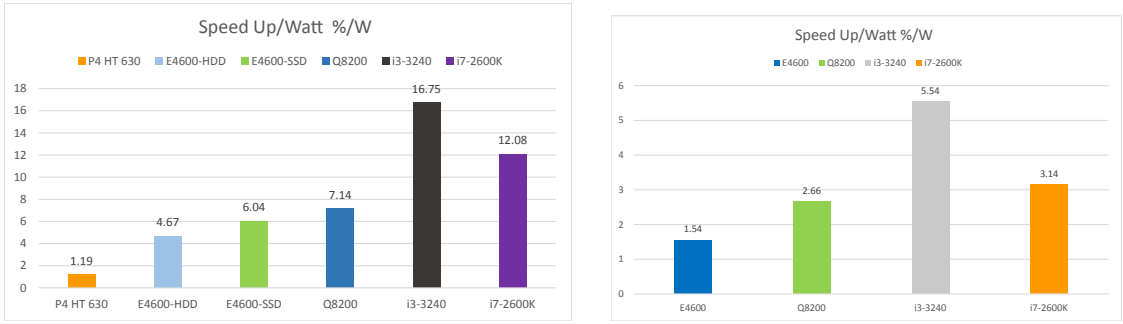
When we consider the relative speed up on a per core basis (fig. 3.7 and fig. 3.8), the 2-core i3-3240 @3.4GHz, distinguishes from the other setups, mainly due to its more recent generation architecture, which have a greater power efficiency.



(A) DiFX - TC016A (B) DiFX - Geodesy K08161

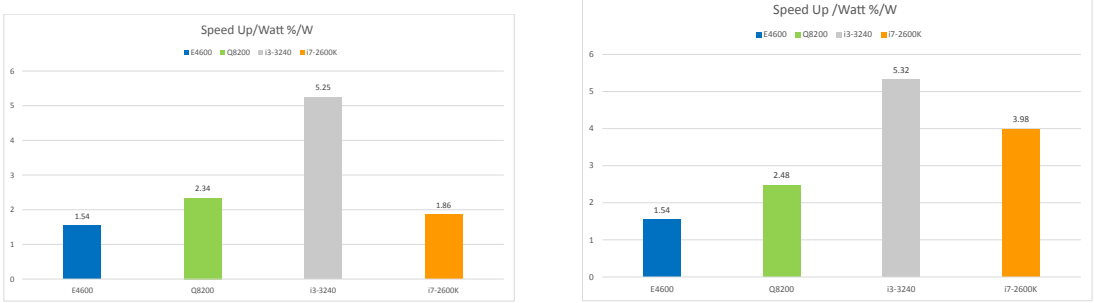
FIGURE 3.8: DiFX Speed Up efficiency

The values of estimated speed up per watt (fig. 3.9, fig. 3.10) which actually uses only the TDP values as given per the specification, confirms the greater power efficiency of the newer generation CPUs.



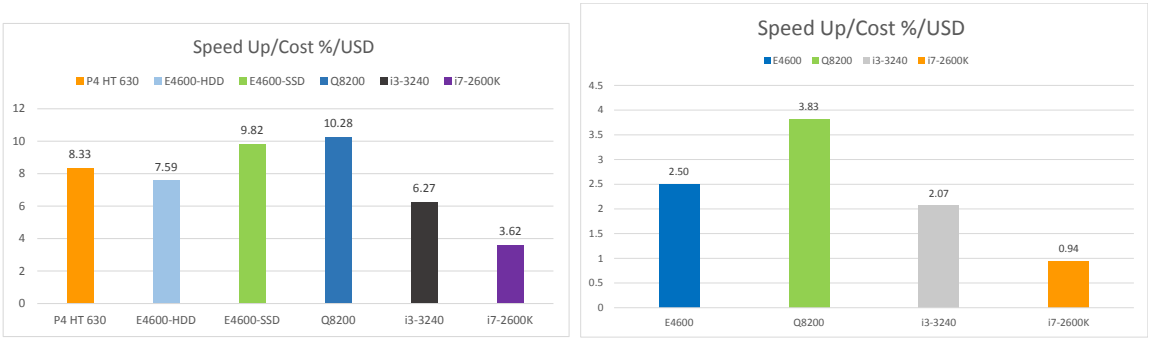
(A) DiFX - RDV70 (B) DiFX - V252-LBA

FIGURE 3.9: DiFX Speed Up per watt



(A) DiFX - TC016A (B) DiFX - Geodesy K08161

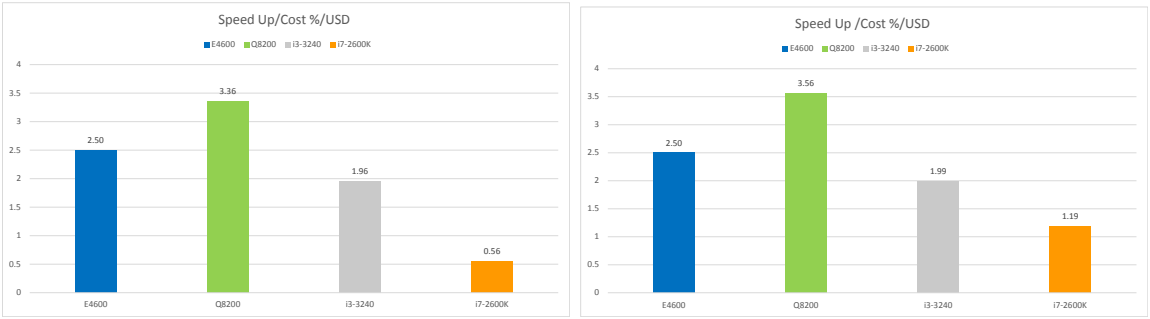
FIGURE 3.10: DiFX Speed Up per watt



(A) DiFX - RDV70 (B) DiFX - V252-LBA

FIGURE 3.11: DiFX Speed Up per cost

Lastly if we analyse the speed up per estimated cost (fig. 3.11, fig. 3.12), we find out that the C2Q-Q8200 @2.333GHz is the most sensible option as concerns the cost.



(A) DiFX - TC016A (B) DiFX - Geodesy K08161

FIGURE 3.12: DiFX Speed Up per cost

3.2.2 Visualising the output

After the lengthy process of analysing the performance of different configurations, the focus was put on the means to visualise the data. DiFX is provided with specific programs to convert the raw visibility files and other information e.g. (baselines u, v, w cordinates) to common files datatypes, used in astronomy, such as the FITS format, through the difx2fits program (Deller, 2014). However it generates a specific FITS-IDI that is to be used with AIPS that was not tested in the scope of this project. Information about the array geometry and baselines configuration are viewed with the well known "fv" fits viewer, but not the visibility. The ease to obtain the visibility information is something that can thus surely be easily improved for the DiFX software.

Fortunately though, the DiFX software comes with a set of Python scripts and modules. Two of them, plotDiFX.py and plotDynamicSpectrum.py are the most useful ones to visualise the visibility information.

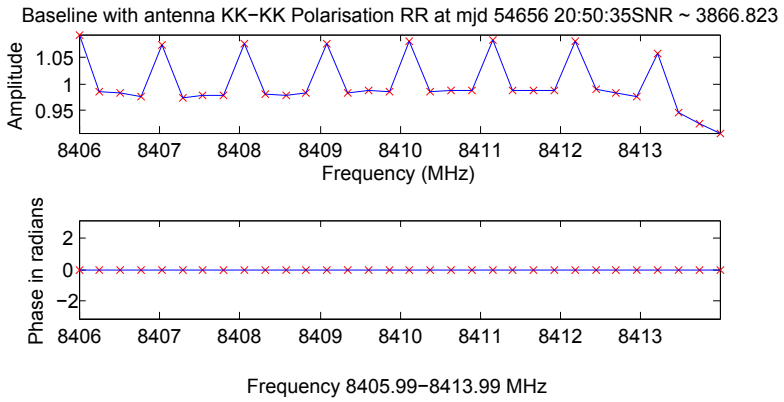


FIGURE 3.13: DiFX - RDV70

The visibility values are displayed on a per baseline basis. The plotDiFX.py script actually uses the complex values of data found inside the raw visibility files. It then computes the absolute value. Finally, it obtains the phase value from the argument of the complex values of visibility. In the scope of this project, ease of accessibility to the visibility values was of a crucial importance. The python scripts were modified to use general Python modules to output the data. The latter output the visibility values already fetched by the scripts to the ".MAT" format (MATLAB®). Shown in fig. 3.13, is a plot of those information fetched using the plotDiFX.py script, for the RDV70 data set, at a selected frequency band, and baseline.

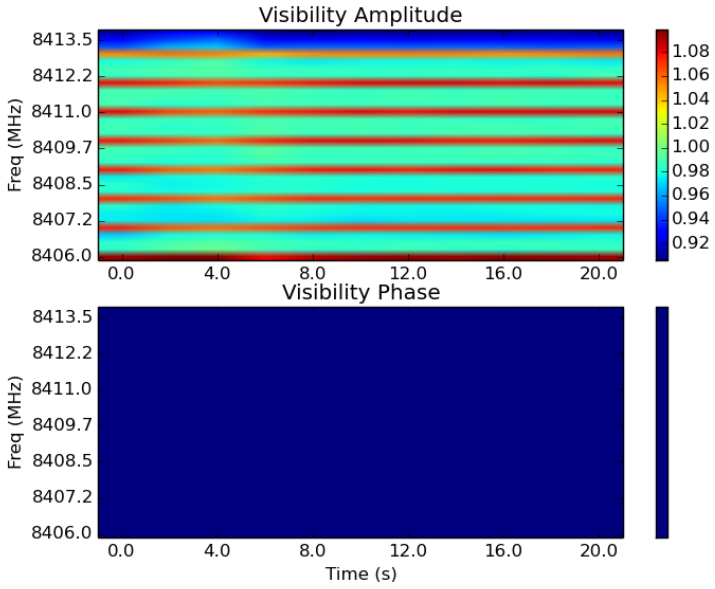


FIGURE 3.14: DiFX - RDV70 - Power Spectrum for the band 8406-8414 MHz Baseline KK-KK

The plotDiFX.py module gives the visibility info at a specific time. It also collects the information over the whole selected observation time length and over all polarisations by default. It outputs a spectrum of the visibility amplitude and phase.

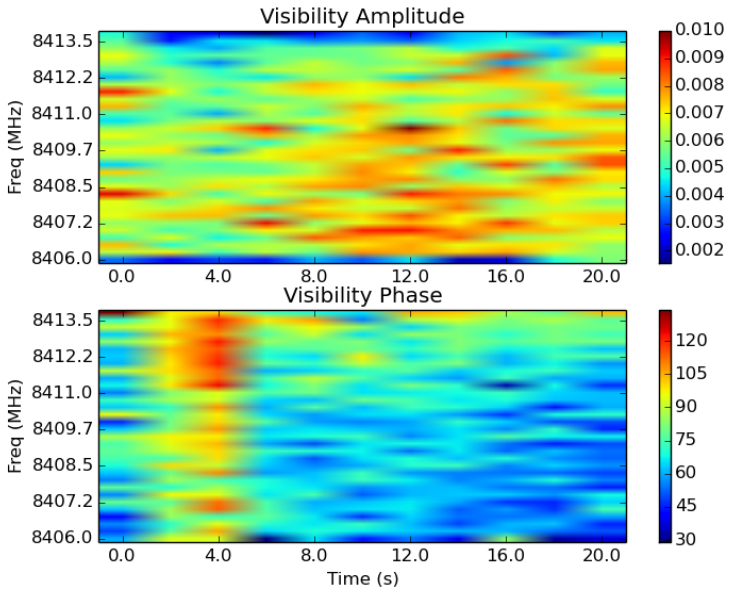
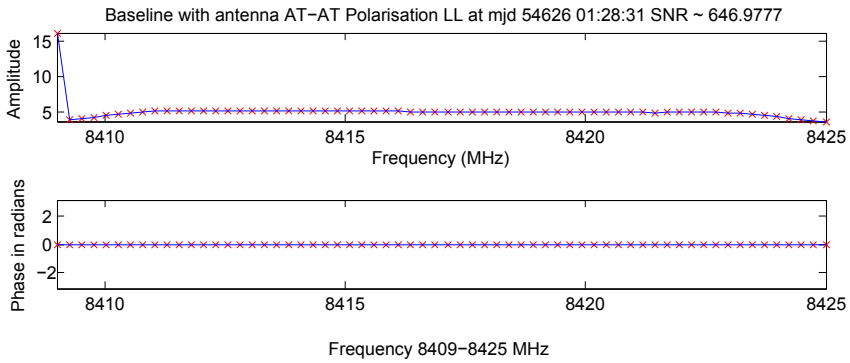
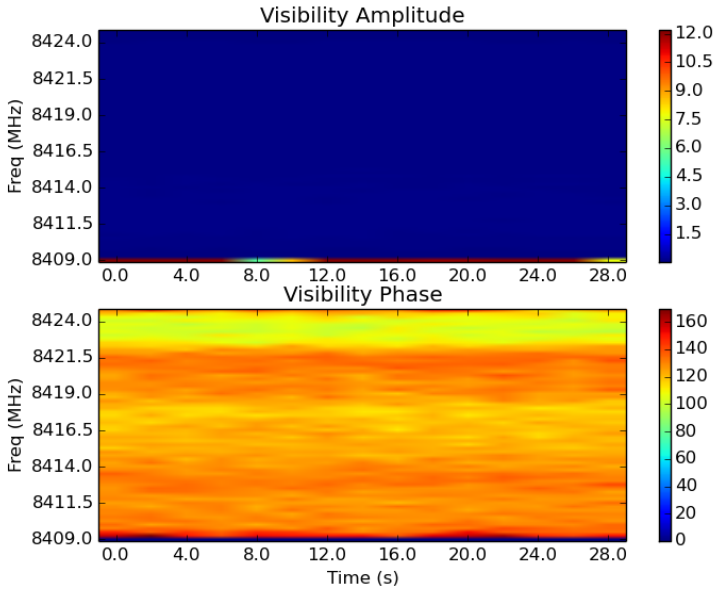


FIGURE 3.15: DiFX - RDV70 - Power Spectrum for the band 8406-8414 MHz Baseline KK-KP

3.3 Experiment - Pseudo Data



(A) DiFX - v252-LBA Data set - Output Visualisation - baseline AT-AT



(B) DiFX - v252-LBA Data set - Power Spectrum for the band 8409-8425 MHz - baseline AT-AT

FIGURE 3.16: DiFX - V252-LBA Dataset - Baseline AT-AT data

A main restriction for the general implementation of the DiFX software is that it uses data formats which are specific to specialised data recorder formats and hardware units. These formats are used by the radio astronomy observation centres. Examples are the LBA format, (2-bit magnitude sign or 2-bit offset quantisation), the MarkIV, MarkV, VLBA (8 bit linear quantisation)[2] (Deller, 2014). Ideally, one would want an implementation of the DiFX software for reading and writing in other formats. However the time frame of the project did not allow for such an implementation, due to the non-trivial nature of the software development involved. However, though, the LBA (.lba) data

[2] http://cira.ivec.org/dokuwiki/doku.php/difx/files

format provided a means to test hypothetic generated data.

The main useful information is provided by Phillips (2005) is that the LBA consists of a 4KB ASCII header. This is followed by a raw baseband data which can be 2-bit data, 8-bit or 10-bit. However, only 2-bit LBA data can be used with DiFX. The only information lacking was how the values to different channels was to be assigned, as the .lba file can contain up to 8 channelsy Phillips (2005).

To investigate the output I used the V252-LBA dataset. The 4KB ASCII header was buffered using raw read operations in MATLAB® by a simple code see the listing A.3. Then, new .lba files with the same file name of specific baselines are generated with the same header and additional 2-bit data written. The sampling rate information of the dataset is provided in the .vex file, and for the V252-LBA dataset is 32 MegaSample/s*(2bits/sample). For the tests 10000x1024x2bit values were generated. The data files generated have the same content, and thus the output produced is related to the other configuration file parameters, which set up specific information about the telescopes, which influence the calculated values of visibility.

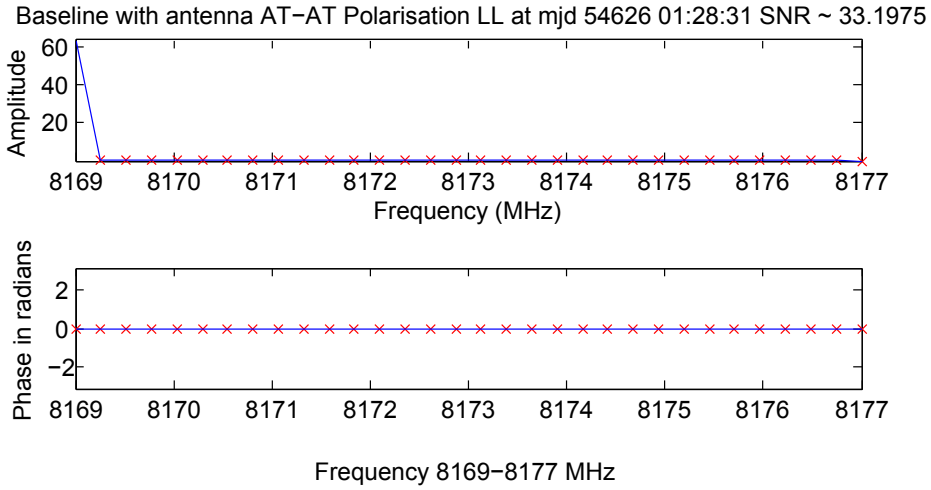


FIGURE 3.17: DiFX - v252-LBA - 2-bit +ve one - baseline AT-AT

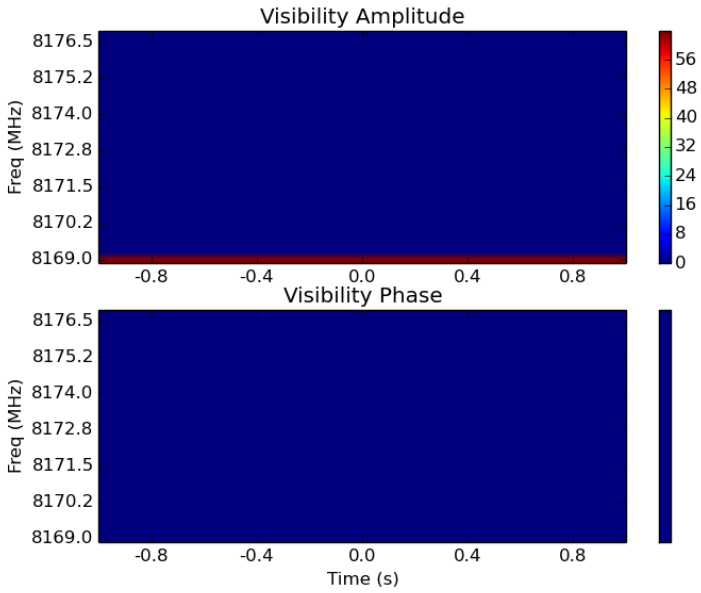


FIGURE 3.18: DiFX - v252-LBA - 2-bit +ve one - Power Spectrum - baseline AT-AT

For the constant highest positive (fig. 3.18) and highest negative 2-bit (fig. 3.22) test data, the output is found be very similar. Fringe like figures appear in the phase spectrum for the AT-CD baseline. With a constant zero 2-bit test data (fig. 3.28) this fringe like shape does not appear, we have instead a smoother distribution if we refer to the spectrum plot.

For another test, instead of constant values, random numbers which could take all the 2-bits values were used (fig. 3.30)[3]. With the AT-CD baseline, there is a prominent variation in the Amplitude Spectrum. This happened even though the magnitude of the values were of comparable order. Also, the values of the phases showed much greater variation. Lastly, as test data, a periodic square pulse, of a period of 64, 2-bit values, was used, where one period consists of 32 values with the highest positive 2-bit value and for the next 32, 2-bit values it consisted of zero 2-bit values. The effect observed (fig. 3.34) in both the autocorrelation AT-AT baseline and the AT-CD baseline is the appearance of 4 discrete spectral lines.

[3]To allow the test to be repeated with the same numbers the value of the seed of the random number generator was stored and reused

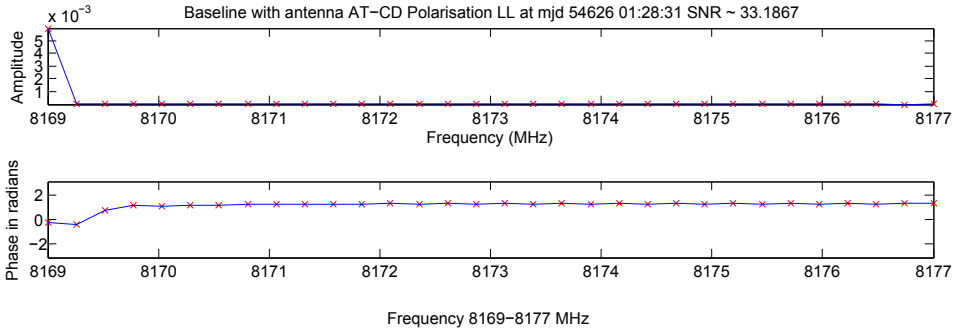


FIGURE 3.19: DiFX - v252-LBA - 2-bit +ve one - baseline AT-CD

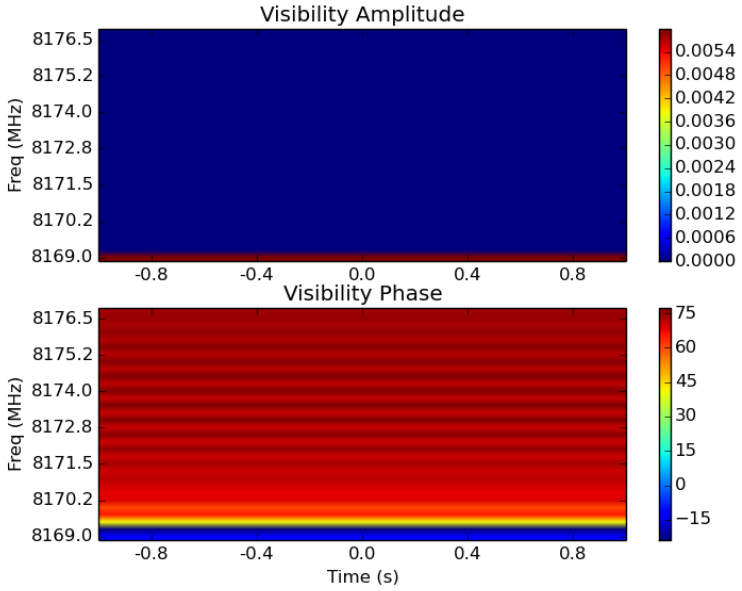


FIGURE 3.20: DiFX - v252-LBA - 2-bit +ve one - Power Spectrum - baseline AT-CD

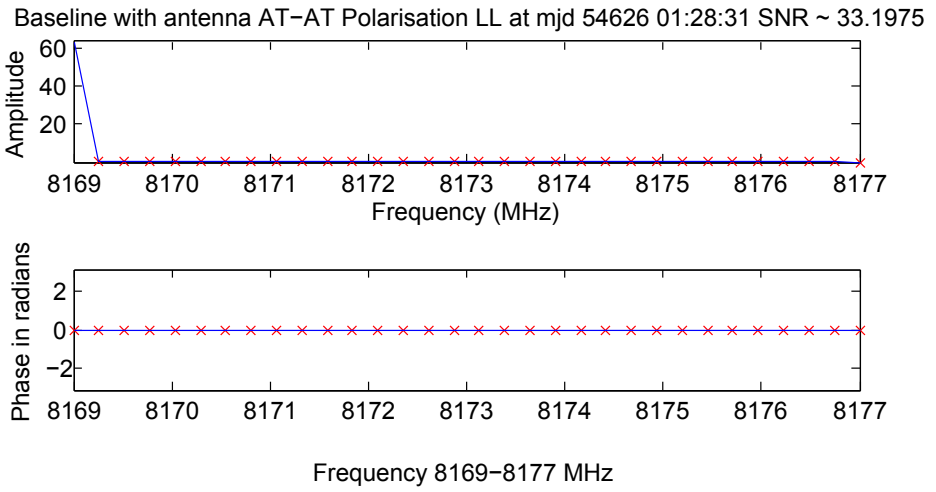


FIGURE 3.21: DiFX - v252-LBA - 2-bit -ve one - baseline AT-AT

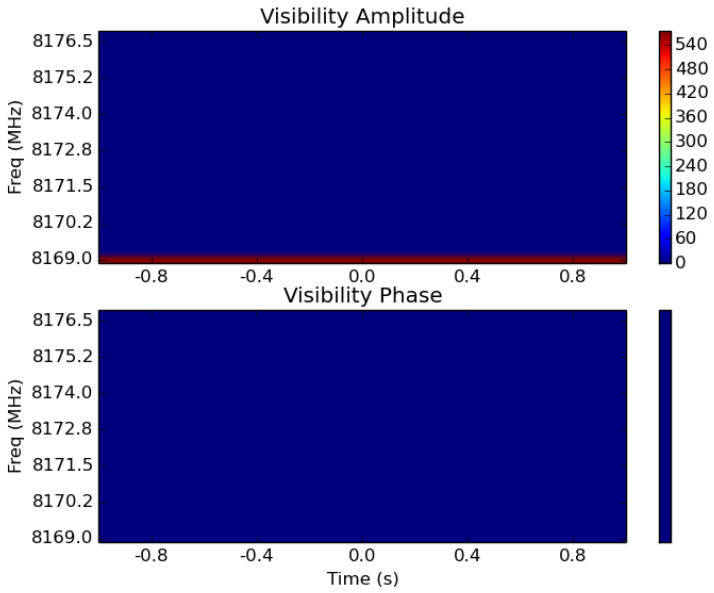


FIGURE 3.22: DiFX - v252-LBA - 2-bit -ve one - Power Spectrum - baseline AT-AT

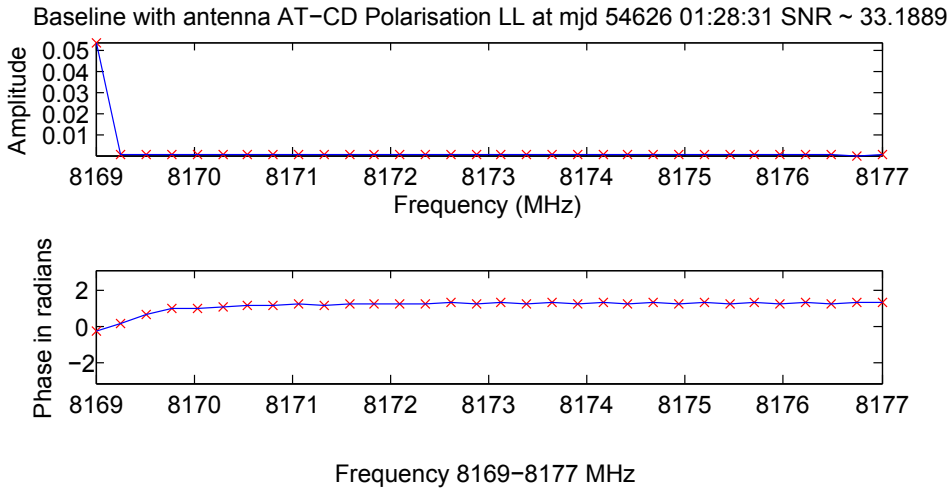


FIGURE 3.23: DiFX - v252-LBA - 2-bit -ve one - baseline AT-CD

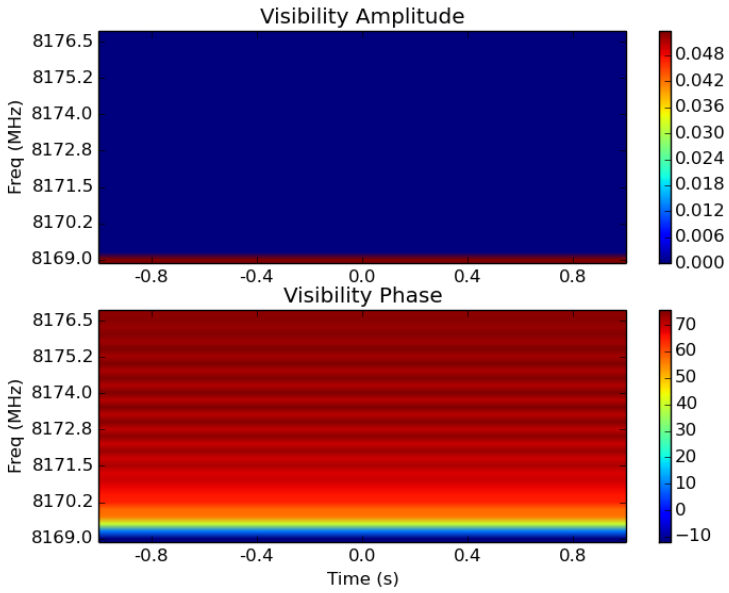


FIGURE 3.24: DiFX - v252-LBA - 2-bit -ve one - Power Spectrum - baseline AT-CD

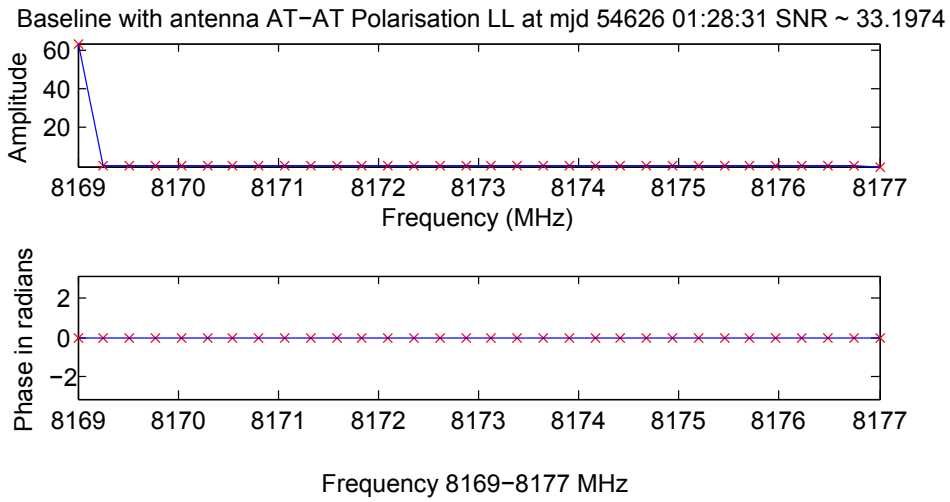


FIGURE 3.25: DiFX - v252-LBA - 2-bit zeros - baseline AT-AT

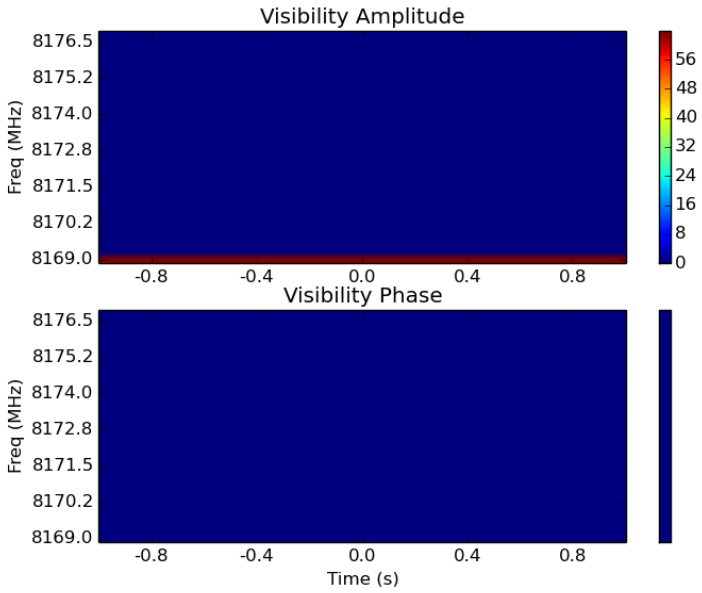


FIGURE 3.26: DiFX - v252-LBA - 2-bit zeros - Power Spectrum - baseline AT-AT

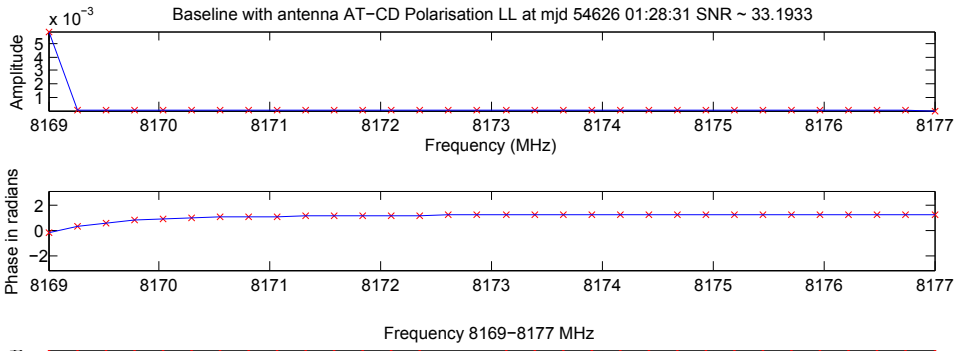


FIGURE 3.27: DiFX - v252-LBA - 2-bit zeros - baseline AT-CD

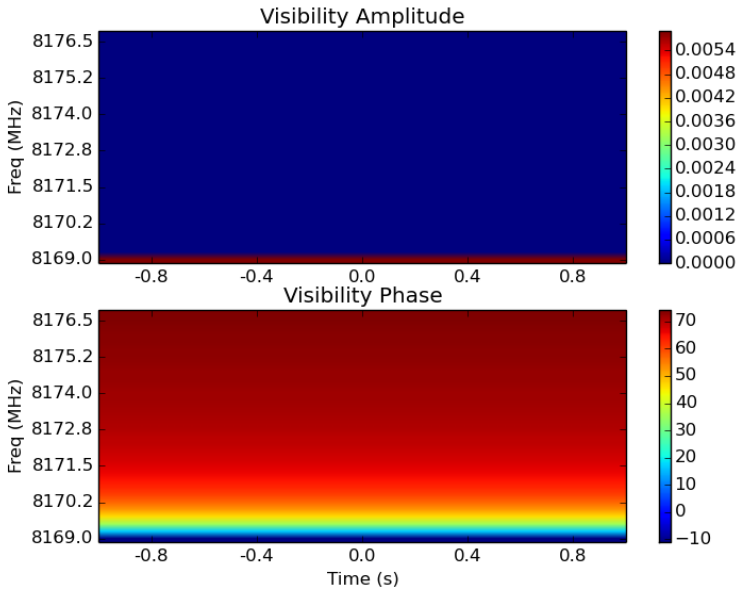


FIGURE 3.28: DiFX - v252-LBA - 2-bit zeros - Power Spectrum - baseline AT-CD

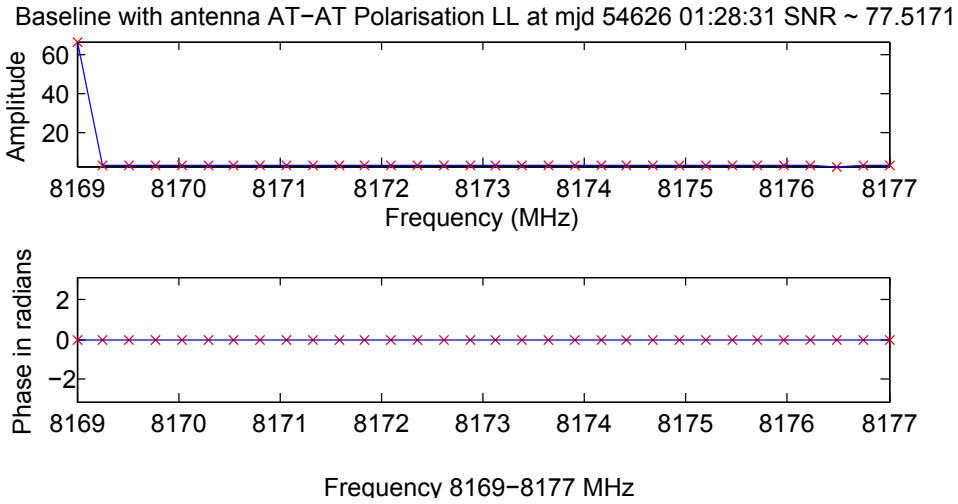


FIGURE 3.29: DiFX - v252-LBA - 2-bit random values - baseline AT-AT

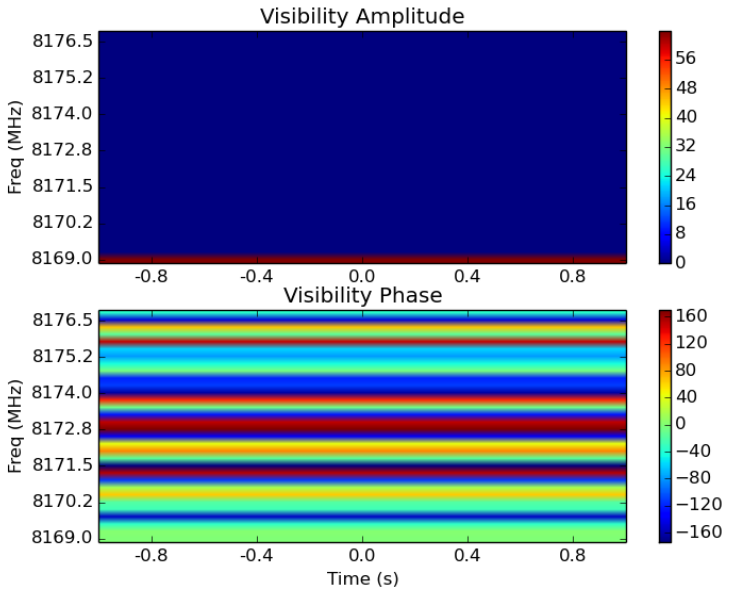


FIGURE 3.30: DiFX - v252-LBA - 2-bit random values - Power Spectrum - baseline AT-AT

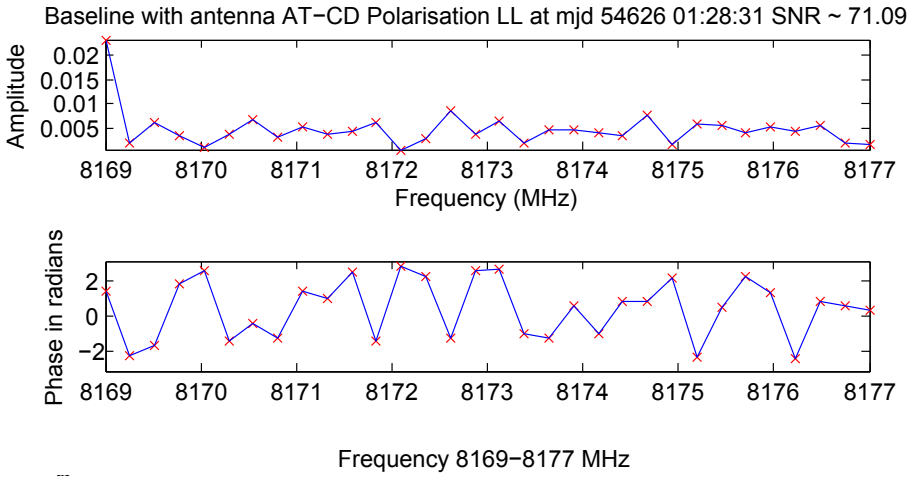


FIGURE 3.31: DiFX - v252-LBA - 2-bit random values - baseline AT-CD

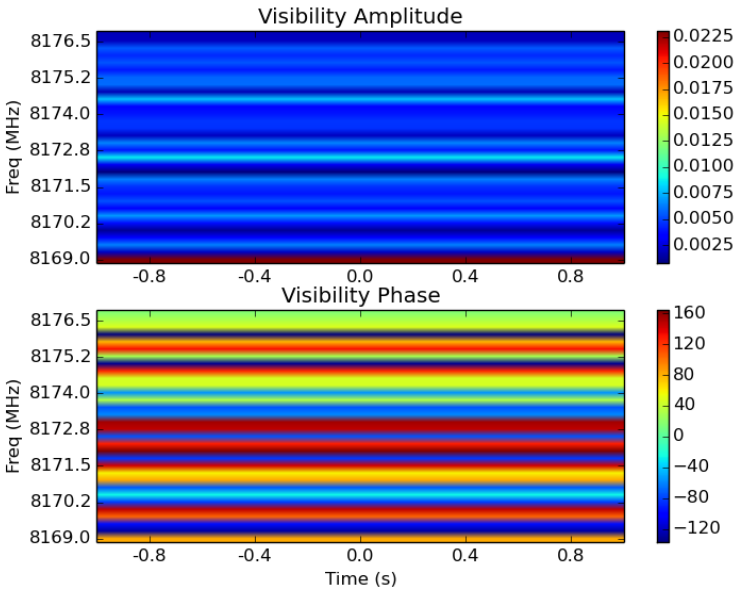


FIGURE 3.32: DiFX - v252-LBA - 2-bit random values - Power Spectrum - baseline AT-CD

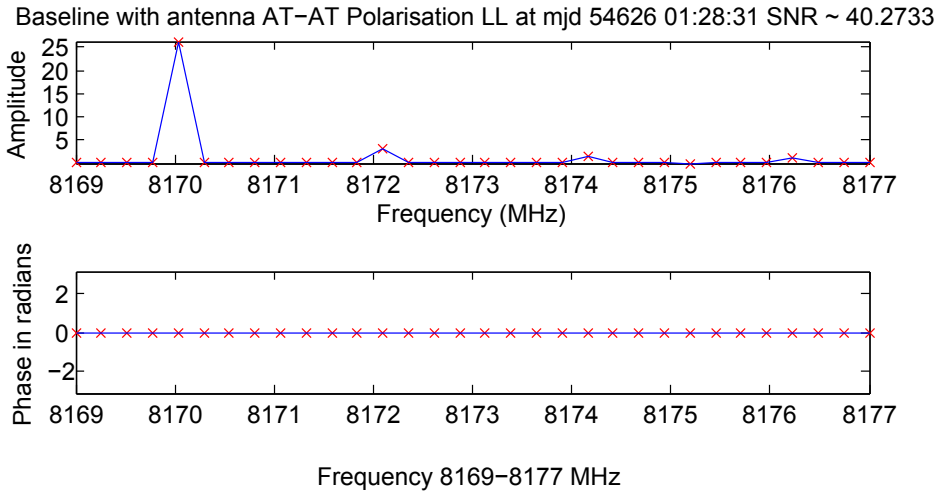


FIGURE 3.33: DiFX - v252-LBA - 2-bit +ve square pulse - baseline AT-AT

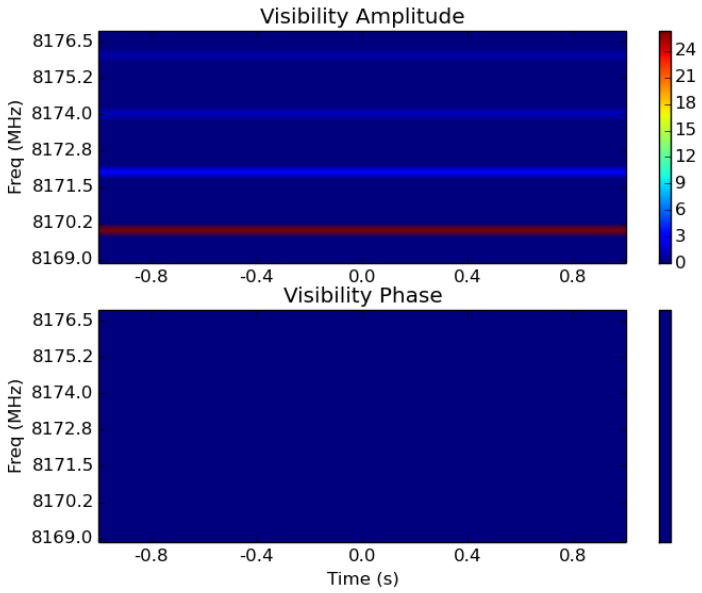


FIGURE 3.34: DiFX - v252-LBA - 2-bit +ve square pulse - Power Spectrum - baseline AT-AT

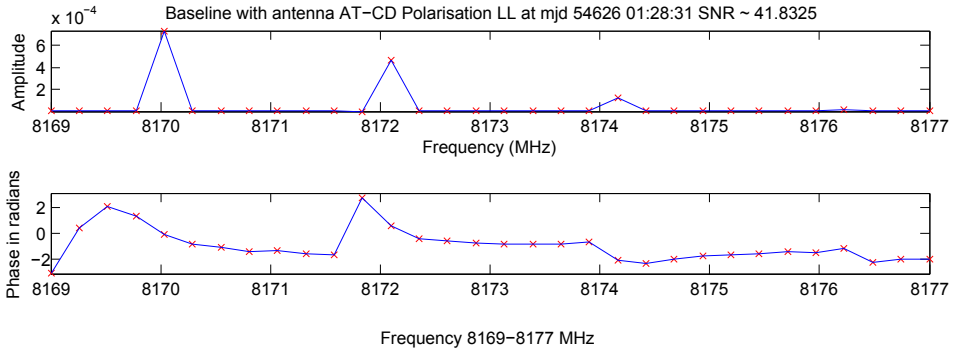


FIGURE 3.35: DiFX - v252-LBA - 2-bit +ve square pulse - baseline AT-CD

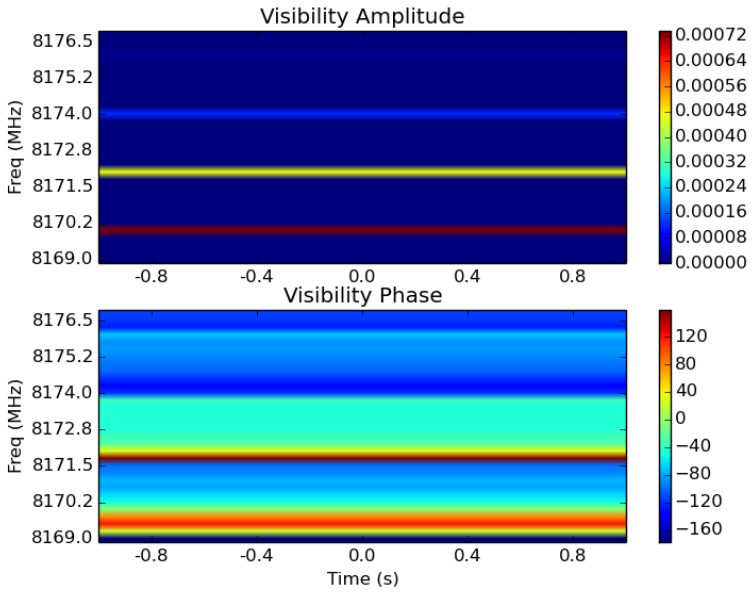


FIGURE 3.36: DiFX - v252-LBA - 2-bit +ve square pulse - baseline AT-CD

Chapter 4

Alternative General Purpose Computing Hardware - GPGPU prospects

The chapter 3 developed around the implementation on CPUs as General Purpose Computing Hardware. Here in this brief chapter, the alternative on Graphics Processing Units is discussed.

4.1 General Purpose Computing on Graphics Processing Units (GPGPU)

The present work done in this project with DiFX with CPUs is mostly a feasibility study, as developed in section 3.2, use of more processing core scaled with performance increase, where the data favored computation, and it was estimated that the more power efficient setups performed better on a performance per watt basis. However on a performance basis we shall see that an important alternative which has recently gained much momentum is that of going to more massively parallel schemes where for this instance Graphics Processing Units (GPUs) are used.

4.1.1 The GPU and Software correlators

GPUs are a type of general purpose computing hardware, which were initially mainly aimed to accelerate tasks related to the display of 3D graphics and perform vector calculations involved in consumer industry 3D games. For the past 2 decades however its application has spread out to many new areas due to the following consideration.

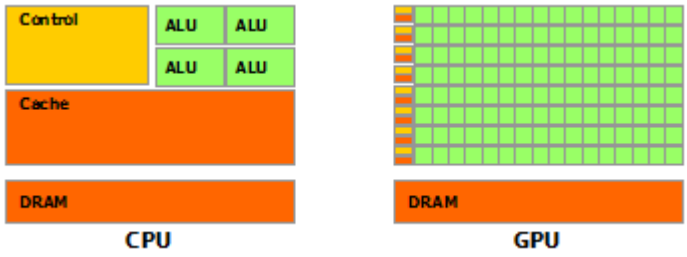


FIGURE 4.1: (CUDA, 2015) CPU vs GPU

As illustrated by fig. 4.1 on CPUs a higher area on the processor chip is allocated to hardware related to data control, and caching memory and actually the processing units displayed in green and known as the ALUs (Arithmetic Logic Units) are allocated to a much smaller area. In constrast GPUs allocate a much greater chip area to ALUs. The logical reasons for both consideration is that CPUs are aimed at doing very general work, mostly serial programs and thus relies on caching, which can be described as the process of using prediction algorithms which store the most frequently used data on very high speed memory caches so as to favorise context switching. GPUs on the other hand are massively parallel devices and are mostly aimed at working on applications suited to parallel workflow which is the case in 3D graphics which relies mostly on matrix operations which in terms of data processing suit perfectly to parallel schemes.

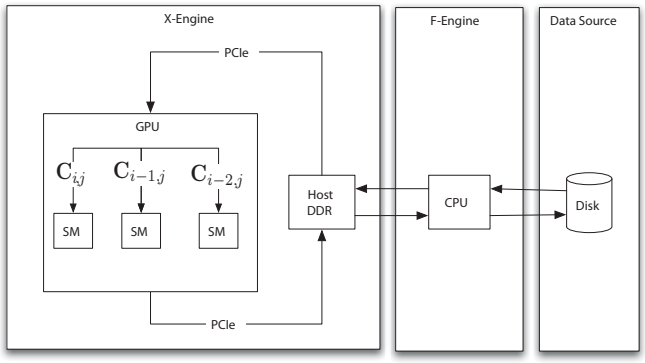


FIGURE 4.2: (Woods et al., 2010) GPU correlator implementation

Cross correlation is one of those application and in fact Woods et al. (2009) addressed the question by isolating part of the DiFX software in a MSc thesis research, and implemented the scheme shown in fig. 4.2, where the most intensive calculation is the X-part (calculation of $\mathbf{C}_{ij}$), i.e cross-multiplication as discussed in 2.4.1, was implemented on GPU. The SM (Streaming Multiprocessors (CUDA, 2015)) as shown on fig. 4.2 actually represents groups of independent GPU-cores.Woods et al. (2009) used the NVIDIA 9800GT GPU which had 14 SM and 112 gpu-cores, and obtained a speedup of 12.5x over implementation on a 4-core CPU, the Intel Xeon Harpertown

X5450.

Most of the other works address the X-engine, as the computation scales as discussed in the introduction with the number of baselines, as $N(N+1)/2$. A notable work is done by Clark et al. (2012) where their CUDA™-C source code is available at `https://github.com/GPU-correlators/xGPU`, where an extensive work was done on optimising the arithmetic occupancy with specific GPU architectures. An explicit data organisation strategy which relate to the bit-precision and how the actual digital data is processed and stored for intermediate calculation on the GPU is used (Clark et al., 2012). They claim to be able to achieve an occupancy of 79% with the strategy used (Clark et al., 2012).

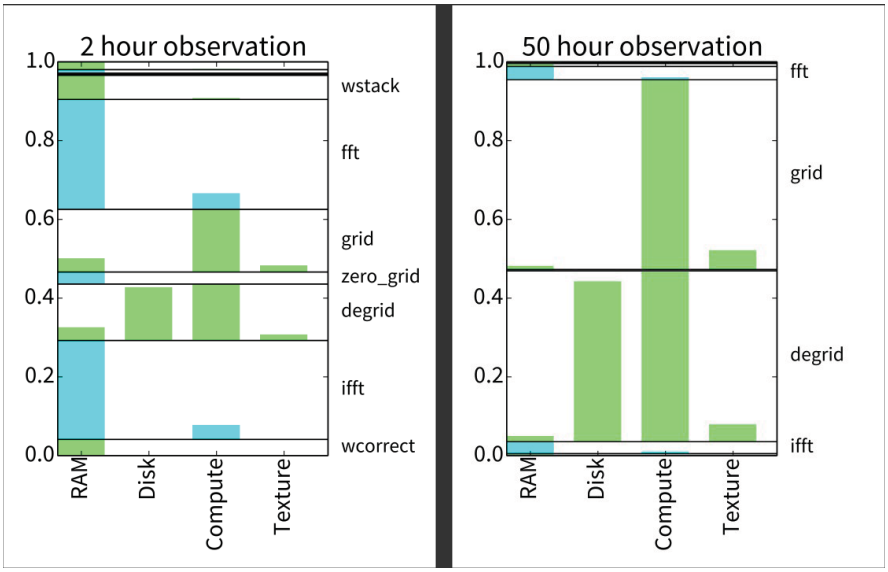


FIGURE 4.3: Ratcliffe (2015) Process Load

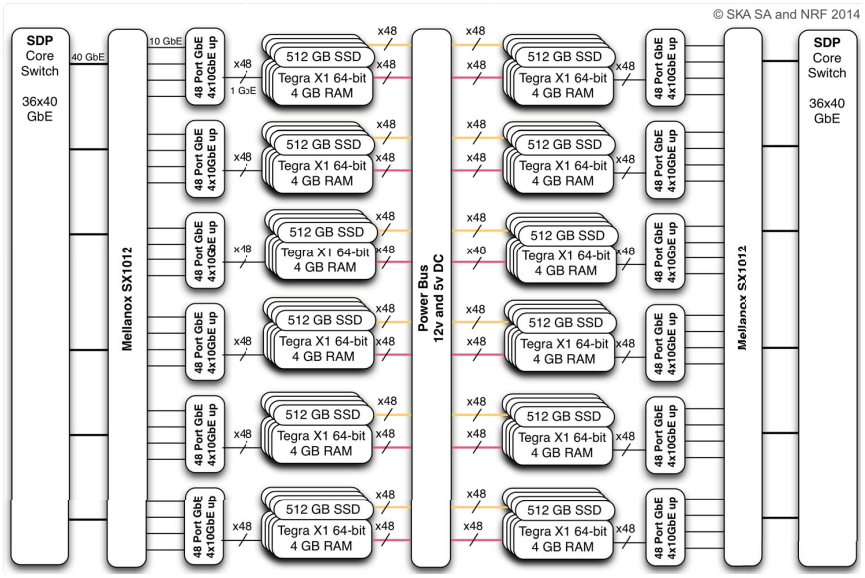


FIGURE 4.4: Ratcliffe (2015) Embedded GPU SoC Design

The research in this field is very active and there has been a recent presentation by Ratcliffe (2015) at the NVIDIA GPU Technology Conference. He points out that for higher scale observations, of longer periods of time (50 hours) with a buffer data of the order of 1 petabyte (1×10^6 GB), as shown in fig. 4.3, the processing involved in the image synthesis process is dominated by the computation of cross-correlation. Ratcliffe (2015) and coworkers are actually prototyping designs using a distributed computing array with 1056 nodes with low power energy efficient System on Chip (SoC) GPUs, the Nvidia Tegra X1, which comprise of a power-efficient 256-cores GPU architecture with a 64-bit 8-core ARM processor, of a TDP of about 10 W, with a floating point performance of about 500 GFlops, shown in fig. 4.4. Specific computation task related to cross correlation are to be offloaded from already in operation High Performance Computing servers (Ratcliffe, 2015).

Thus in conclusion the use of highly parallel architectures and distributed computing, in the field of synthesis imaging shows great prospects.

Chapter 5

Conclusions and future work

5.1 Conclusions

The work done with DiFX shows that the use of completely general purpose hardware for doing software correlation, which take into account non-trivial configuration parameters in relation to the observation and instruments used is feasible. The performance scales well with the use of a higher number of cores or nodes as long as the data content favors computation in constrast to low occupancy and data transfer.

5.2 Future work

The main barrier as concerns the implementation of DiFX is the unavailability of a more flexible data format, which would allow general hardware to be used without specialised data units. The possibility to simulate data should be made possible, and this is an extensive process which requires more research and development on an appropriate way to encode data in the formats accepted by DiFX or simply implement a custom flexible format for the software.

Work towards the accessibility of the output data through investigation of the means used actually to visualise data on the formats output by DiFX such as the FITS-IDI format and output to Mark4 formats along with a custom user friendly documentation is still to be done.

As concerns the hardware implementation the whole approach taken by DiFX, would be worth to be implemented on more energy efficient general purpose hardware such as low power and high performance graphics processing units which are actually gaining momentum in terms of application worldwide, particularly in astronomy.

Appendix A

DiFX - Custom documentation

A.1 DiFX - Setup

The following follows the installation instructions given, on http://cira.ivec.org/dokuwiki/doku.php/difx/installation .
To install the DiFX-2.4 software correlator on Ubuntu 14.04 LTS we do the following things. You have to first apply for an account to get access to the svn repository of the DiFX software, at the following url : http://svn.atnf.csiro.au/cgi-bin/svnpasswd

1. First we install the packages on which the software depends such as MPI and other stuffs

```
sudo apt-get install build-essential subversion
sudo apt-get install libopenmpi-dev libfftw3-dev libtool bison
sudo apt-get install flex pkg-config automake libexpat1-dev
sudo apt-get install gfortran openmpi-bin
sudo apt-get install rpcbind
sudo apt-get install python-numpy python-scipy python-matplotlib
```

2. Once you have the dependencies installed and a SVN account ready you'll be able to download the source of the software somewhere using the following command.

```
sudo svn co https://svn.atnf.csiro.au/difx/master_tags/DiFX-2.3
```

   At a certain time you'll be asked for your login information and password.

3. Install the IPP software availble, i.e.

```
>> cd $IPP_PATH
>> sudo chmod +x install.sh
>> sudo bash install.sh
```

where $ IPP_PATH means the directory of the IPP installation files

4. Install PGPLOT, following the instructions of the following website:
   http://pendientedemigracion.ucm.es/info/Astrof/software/howto/howto-pgplot.html
   Do the following,

```
>> cd /usr/local/src
>> sudo mv ~/Downloads/pgplot5.2.tar.gz .
>> sudo tar zxvf pgplot5.2.tar.gz
>> sudo mkdir /usr/local/pgplot
>> cd /usr/local/pgplot
>> sudo cp /usr/local/src/pgplot/drivers.list .
>> sudo /usr/local/src/pgplot/makemake
/usr/local/src/pgplot linux g77_gcc_aout
```

   Edit the file makefile

```
>> sudo gedit makefile &
```

   Change

```
FCOMPL=g77
# to
FCOMPL=gfortran
```

   Then save, and compile

```
>> sudo make
>> sudo make cpg
>> sudo make clean
```

   Export the paths

```
>> export PGPLOT_DIR=/usr/local/pgplot
>> export PGPLOT_DEV=/Xserve
```

5. Then go back to install DiFX-2.4, edit the setup.bash file,

```
>> cd $DIFX_ROOT
>> sudo gedit setup.bash &
```

6. In the setup.bash thus change the following paths, so that they match the directory of DiFX and that of PGPLOT and also of IPP, and MPI.
   Also uncomment the lines which would allow log messages to be parsed, here following is an example for my setup,

```
####### ROOT PATHS ############################
export DIFXROOT=/home/ruben/Final_Year_Project/DiFX/DiFX-2.3
export DIFX_PREFIX=$DIFXROOT
export PGPLOTDIR=/usr/local/pgplot/
export IPPROOT=/opt/intel/ipp/bin/

####### COMPILER ##############################
export MPICXX=/usr/local/bin/mpicxx

# ALSO UNCOMMENT THESE THINGS ALSO

####### PORTS FOR DIFXMESSAGE ###############
# Uncomment these to enable DIFX_MESSAGES
export DIFX_MESSAGE_GROUP=224.2.2.1
export DIFX_MESSAGE_PORT=50201
export DIFX_BINARY_GROUP=224.2.2.1
export DIFX_BINARY_PORT=50202
```

7. In the terminal from the root folder of DiFX, use the geniepc script with input the path to */opt/intel*

```
>> ./geniepc /opt/intel
```

8. Then we can start the installation of DiFX-2.4, in the root folder of DiFX, do the following,

```
>> source setup.bash
>> ./install-difx
```

9. There is a step to make rpcbind work correctly,

```
>> sudo -i service portmap stop
>> sudo -i rpcbind -i -w
>> sudo -i service portmap start
>>echo 'OPTIONS="-w -i"' | sudo tee /etc/default/rpcbind
>>sudo service portmap restart
```

10. This completes DiFX installation

A.2 Using DiFX

1. To use DiFX we have to start the calculation server and check it for the host,

```
>> startCalcServer
>> checkCalcServer 127.0.0.1
>> export CALC_SERVER=127.0.0.1
```

   To see the processes, open another terminal

```
>> errormon2
```

2. When all this is done, one can try the example files, from the root folder of the RDV70 data, we generate the configuration files for DiFX

```
>> vex2difx example.v2d
>> calcif2 example_1.calc
```

3. And for a single node we can launch the correlation task as follows

```
>> mpirun -np 8 mpifxcorr example_1.input
```

4. When the correlation is done, you can get the plot in the following way,

```
>> plotDiFX.py *.difx/* -i *.input --toscreen
```

   There are other options available such as choosing the frequency, specific polarisations, baselines, etc. These can be viewed using the help option

```
>> plotDiFX.py -h
```

5. To view the dynamic spectrum plots we use plotDynamicSpectrum.py

```
>> plotDynamicSpectrum.py *.difx/* -i *.input -b 257 --toscreen
```

Here we plotted the specific baseline with id 257 as specified by the option -b. To view the other options available such as choosing the frequency, specific polarisations, baselines, etc. Use the help option

```
>> plotDynamicSpectrum.py -h
```

A.3 Custom Modification

The following listing A.1 is just the original plotDiFX.py code by Deller (2014) with the addition of line 293 to 312 to produce .MAT files , which export the values used by plotDiFX.py to calculate values for its plots, and the plots values also.

```
1  #!/usr/bin/python
2  import sys, os, struct, time, pylab, math, numpy, scipy.io
3  import parseDiFX
4  from optparse import OptionParser
5  from numpy import fft
6
7  helpstr = "plotDiFX.py [options] <difx file 1> <difx file 2>\
8   ... [difx file N]\n\n"
9  helpstr += "Flashes bandpasses of selected bands overlaid"
10 parser = OptionParser(helpstr)
11 parser.add_option("-f", "--freq", dest="freq", \
12 metavar="targetfreq", default="-1",
13                   help="Only display visibilities \
14 from this frequency index")
15 parser.add_option("-b", "--baseline", dest="baseline",\
16  metavar="targetbaseline", default="-1",
17                   help="Only display visibilities from\
18  this baseline num")
19 parser.add_option("-c", "--maxchannels", dest="maxchannels",\
20  metavar="MAXCHANNELS",
21                   default="33000",
22                   help="The length of the array that will\
23           be allocated to hold vis results")
24 parser.add_option("-p", "--pols", dest="pollist",\
```

```
25  default="RR,RL,LR,LL,YY,YX,XY,XX",
26                    help="Only display polarization pairs\
27  from this comma-separated list")
28 parser.add_option("-v", "--verbose", dest="verbose", \
29 action="store_true", default=False,
30                    help="Turn verbose printing on")
31 parser.add_option("-i", "--inputfile", dest="inputfile", default="dy9-=",
32                    help="An input file to use as guide for number\
33  of channels for each freq")
34 parser.add_option("--toscreen", dest="toscreen", default=False,\
35  action="store_true",
36                    help="Plot to the screen, otherwise to png files")
37 parser.add_option("--singlevis", dest="singlevis", default=False,\
38  action="store_true",
39                    help="Stop plotting as soon as there\
40            is a time change")
41 parser.add_option("--firstpermatch", dest="firstpermatch",\
42  default=False, action="store_true",
43                    help="For each baseline plot only\
44          the first matching entry")
45 parser.add_option("--singleplot", dest="singleplot",\
46  default=False, action="store_true",
47                    help="Plot everything on one axis")
48 parser.add_option("--unwrap", dest="unwrap",\
49  default=False, action="store_true",
50                    help="Unwrap the phase")
51 parser.add_option("--noauto", dest="noauto",\
52  default=False, action="store_true",
53                    help="Exclude autocorrelation data")
54 parser.add_option("--amprange", dest="amprange",\
55  default="-1,-1",
56                    help="Range for the y axis for\
57  amplitude subplot in form min,max")
58 (options, args) = parser.parse_args()
59
60 if len(args) < 1:
61     parser.error("You must supply at least \
62       one DiFX output file!")
63
64 numfiles = len(args)
65
66 targetbaseline = int(options.baseline)
67 targetfreq = int(options.freq)
```

```
68  maxchannels    = int(options.maxchannels)
69  pollist        = options.pollist.upper().split(",")
70  verbose        = options.verbose
71  inputfile      = options.inputfile
72  toscreen       = options.toscreen
73  singlevis      = options.singlevis
74  singleplot     = options.singleplot
75  firstpermatch  = options.firstpermatch
76  amprange       = options.amprange.split(',')
77  unwrap         = options.unwrap
78  noauto         = options.noauto
79
80  if inputfile == "":
81      parser.error("You must supply an input file!")
82  if len(amprange) != 2:
83      parser.error("Supply amprange in the form min,max!")
84
85  (numfreqs, freqs) = parseDiFX.get_freqtable_info(inputfile)
86  (numtelescopes, telescopes) = \
87  parseDiFX.get_telescopetable_info(inputfile)
88  (numdatastreams, datastreams) = \
89  parseDiFX.get_datastreamtable_info(inputfile)
90  (numbaselines, baselines) = \
91  parseDiFX.get_baselinetable_info(inputfile)
92  amprange[0] = float(amprange[0])
93  amprange[1] = float(amprange[1])
94
95  if numfreqs == 0 or numtelescopes == 0 or numdatastreams\
96   == 0 or numbaselines == 0:
97      parser.error("Couldn't parse input file " + inputfile + " correctly")
98
99  chans = []
100 amp = []
101 phase = []
102 vis = []
103 lag = []
104 lagamp = []
105 #Initialise the arrays
106 for i in range(numfiles):
107     amp.append([])
108     phase.append([])
109     vis.append([])
110     lag.append([])
```

```
111     lagamp.append([])
112     for j in range(maxchannels):
113         amp[i].append(0.0)
114         phase[i].append(0.0)
115         vis[i].append(0.0)
116         lag[i].append(0.0)
117         lagamp[i].append(0.0)
118
119 for i in range(maxchannels):
120     chans.append(i)
121
122 pylab.xlabel("Channel")
123 pylab.ylabel("Amplitude")
124 linestyles = ['b', 'r', 'g', 'k', 'y']
125 difxinputs  = []
126 nextheader  = []
127 mjd         = []
128 seconds     = []
129 freqindex   = []
130 baseline    = []
131 polpair     = []
132 nchan       = []
133 med         = []
134 for filename in args:
135     difxinputs.append(open(filename))
136     freqindex.append(0)
137     nchan.append(0)
138     baseline.append(0)
139     mjd.append(0)
140     seconds.append(0.0)
141     med.append(0.0)
142     polpair.append("")
143     nextheader.append([])
144 unplottedbaselines=[]
145 for i in range(16):
146     for j in range(16):
147         unplottedbaselines.append(j*256+i)
148
149 for i in range(numfiles):
150     nextheader[i] = parseDiFX.parse_output_header(difxinputs[i])
151
152 count = 0
153 keeplooping = True
```

```
154 if not len(nextheader[0]) == 0:
155     startseconds = nextheader[0][2]
156 while not len(nextheader[0]) == 0 and keeplooping:
157     print "Looping..."
158     for i in range(numfiles):
159         snr = 0
160         delayoffsetus = 0
161         if len(nextheader[i]) == 0:
162             nextheader[0] = [] #Will cause an exit
163             break
164         baseline[i] = nextheader[i][0]
165         mjd[i] = nextheader[i][1]
166         seconds[i] = nextheader[i][2]
167         if singlevis and seconds[i] > startseconds:
168             print "Exiting since singlevis was specified"
169         keeplooping = False
170             break
171         freqindex[i] = nextheader[i][5]
172         polpair[i] = nextheader[i][6]
173         nchan[i] = freqs[freqindex[i]].numchan/freqs[freqindex[i]].specavg
174         buffer = difxinputs[i].read(8*nchan[i])
175         if nchan[i] > maxchannels:
176             print "How embarrassing - you have tried to read files \
177         with more than " + str(maxchannels) + " channels.  Please\
178         rerun with --maxchannels=<bigger number>!"
179             sys.exit()
180     if firstpermatch:
181         match_freq = (targetfreq < 0) or (freqindex[i] == targetfreq)
182         match_pol = (polpair[i]) in pollist
183         match_new = baseline[i] in unplottedbaselines
184         if match_new and match_freq and match_pol:
185             targetbaseline = baseline[i]
186             try:
187                 unplottedbaselines.remove(baseline[i])
188                 unplottedbaselines.remove(baseline[i])
189             except:
190                 pass
191             print 'Got new baseline %s, freq %d' %\
192              (str(targetbaseline),freqindex[i])
193         else:
194             if match_new:
195                 print 'Skip new baseline %s with wrong\
196                  freq %d != %d' % (str(baseline[i]),\
```

```
197                 freqindex[i],targetfreq)
198             if match_freq:
199                 print 'Skip old baseline %s with desired\
200                  freq %d = %d' % (str(baseline[i]),freqindex[i]\
201                 ,targetfreq)
202             targetbaseline = 13
203
204         for j in range(nchan[i]):
205             cvis = struct.unpack("ff", buffer[8*j:8*(j+1)])
206             vis[i][j] = complex(cvis[0], cvis[1])
207             amp[i][j] = math.sqrt(cvis[0]*cvis[0] + cvis[1]*cvis[1])
208             phase[i][j] = math.atan2(cvis[1], cvis[0])
209
210         if unwrap:
211         phase[i] = (numpy.unwrap(phase[i]))*180.0/math.pi
212
213     if (targetbaseline < 0 or targetbaseline == baseline[i]) and \
214         (targetfreq < 0 or targetfreq == freqindex[i]) and \
215         (polpair[i] in pollist) and not (noauto and (baseline[i]\
216         % 257) == 0):
217             lag[i] = fft.ifft(vis[i], nchan[i])
218             for j in range(nchan[i]/2):
219                 lagamp[i][j+nchan[i]/2] = abs(lag[i][j])
220             for j in range(nchan[i]/2):
221                 lagamp[i][j] = abs(lag[i][j+nchan[i]/2])
222             med[i] = numpy.median(amp[i][:nchan[i]])
223             if i > 0:
224                 for j in range(len(nextheader[0])):
225                     if nextheader[i][j] != nextheader[0][j]:
226                         print "Headers disagree!"
227                         if verbose:
228                             print nextheader[0]
229                             print nextheader[i]
230                         break
231             if (targetbaseline < 0 or baseline[0] == targetbaseline) and \
232                (targetfreq < 0 or freqindex[0] == targetfreq) and\
233            (polpair[0] in pollist):
234                 maxlag = max(lagamp[i])
235                 maxindex = lagamp[i].index(maxlag)
236                 delayoffsetus = (maxindex - nchan[i]/2) *\
237         1.0/(freqs[freqindex[0]].bandwidth*2)
238         if singleplot:
239             ls = linestyles[count%len(linestyles)]
```

```
240             print "Setting linestyle to " + ls
241             count += 1
242         else:
243             ls = linestyles[i]
244                 pylab.subplot(311)
245         if amprange[1] > 0:
246
247
248             pylab.ylim(amprange)
249                 pylab.plot(chans[:nchan[i]], amp[i][:nchan[i]], ls)
250                 pylab.subplot(312)
251                 pylab.plot(chans[:nchan[i]], phase[i][:nchan[i]], ls+'+')
252                 pylab.subplot(313)
253                 pylab.plot(chans[:nchan[i]], lagamp[i][:nchan[i]], ls)
254                 lagamp[i][maxindex] = 0
255                 if maxindex > 0:
256                     lagamp[i][maxindex-1] = 0
257                 if maxindex < len(lagamp)-1:
258                     lagamp[i][maxindex+1] = 0
259                 rms = numpy.std(lagamp[i])
260                 snr = maxlag/rms
261         #print lagamp[i][:nchan[i]+1]
262                 print snr
263                 print maxlag
264
265     if (targetbaseline < 0 or baseline[0] == targetbaseline) and \
266        (targetfreq < 0 or freqindex[0] == targetfreq) and \
267        (polpair[0] in pollist) and \
268         not (noauto and (baseline[i] % 257) == 0):
269         pylab.subplot(311)
270         ant1index = baseline[0] / 256 - 1
271         ant2index = baseline[0] % 256 - 1
272         ant1name  = telescopes[ant1index].name
273         ant2name  = telescopes[ant2index].name
274         lowfreq   = freqs[freqindex[0]].freq
275         hifreq    = freqs[freqindex[0]].freq + freqs[freqindex[0]].bandwidth
276         if freqs[freqindex[0]].lsb:
277             lowfreq -= freqs[freqindex[0]].bandwidth
278             hifreq -= freqs[freqindex[0]].bandwidth
279         print seconds[0]
280         hour      = int(seconds[0]/3600)
281         minute    = int(seconds[0]/60 - 60*hour)
282         second    = int(seconds[0] - (3600*hour + 60*minute))
```

```
283         mjdstring = "%d %02d:%02d:%02d" % (mjd[0], hour, minute, second)
284         titlestr = "Baseline %s-%s, Freq %.2f-%.2f, pol %s, date %s" % \
285                     (ant1name, ant2name, lowfreq, hifreq,\
286      polpair[0], mjdstring)
287         pylab.ylabel("Amplitude")
288         pylab.subplot(312)
289         pylab.ylabel("Phase (deg)")
290         pylab.subplot(313)
291         pylab.ylabel("Lag")
292         pylab.xlabel("Channel")
293     #Save to .MAT files
294     scipy.io.savemat("%s_baseline%03d_freq_%02d_pol_%sPlotDiFX.mat" %\
295     (inputfile, baseline[i], freqindex[i], polpair[i]),\
296      mdict={"Visibility": vis[0][:j+1],\
297     "nchan": nchan[i],\
298     "chan": chans[:nchan[i]],\
299     "lag":lag[i],\
300     "amp": amp[i][:nchan[i]],\
301     "phase":phase[i][:nchan[i]],\
302     "baseline":baseline[i],\
303     "freqindex":freqindex[i],\
304     "ant1name":ant1name,\
305     "ant2name":ant2name,\
306     "lowfreq":lowfreq,\
307     "hifreq":hifreq,\
308     "mjdstring":mjdstring,\
309     "snr":snr,\
310     "csnr":((snr-3)/2),\
311     "delayOffSet":delayoffsetus,\
312     "polpair":polpair[i]})
313     if not singleplot:
314         pylab.subplot(311)
315             pylab.title(titlestr)
316         pylab.subplot(313)
317             pylab.figtext(0.0,0.0,"Fringe S/N %0.2f \
318         @ offset %0.2f us (%s)" % \
319                             (snr, delayoffsetus,\
320         "raw S/N is overestimated - corrected value ~%0.2f" % ((snr-3)/2)))
321             if toscreen:
322                 pylab.show()
323             else:
324             pylab.savefig("%s_baseline%03d_freq_%02d_pol_%s.png" %\
325         (inputfile, baseline[i], freqindex[i], polpair[i]))
```

```
326             pylab.clf()
327         print "Median values were:"
328         for i in range(numfiles):
329             print "File %d: %.6f" % (i, med[i])
330         if numfiles == 2:
331             print "Ratio of medians was " + str(med[1]/med[0])
332     if keeplooping:
333         for i in range(numfiles):
334             nextheader[i] = parseDiFX.parse_output_header(difxinputs[i])
335 if singleplot:
336     if toscreen:
337         pylab.show()
338     else:
339         pylab.savefig("%s_baselineALL_freq_ALL_pol_ALL.png" % (inputfile))
```

LISTING A.1: PlotDiFX.py by Deller et al. (2007) with additional code (line 293 to 312) to output to MATLAB® .MAT files

The following listing A.2 is used to plot the values in the .MAT files, which have to be loaded before in MATLAB®. Then the following command can be used to obtain the plot.

```
>> DiFXplot(lowfreq,hifreq,nchan,phase,amp,ant1name,...
ant2name,polpair,mjdstring,snr,csnr)
```

```
1  function DiFXplot(lowfreq,hifreq,nchan,phase,...
2      amp,ant1name,ant2name,polpair,mjdstring,snr,csnr)
3  close all
4  y = (lowfreq:(hifreq-lowfreq)/double(nchan-1):hifreq);
5  subplot(2,1,1)
6  plot(y,amp,'rx',y,amp,'b-')
7  ylabel('Amplitude');
8  xlabel('Frequency (MHz)');
9    title(strcat({'Baseline with antenna '},ant1name,'-',...
10       ant2name,{' Polarisation '},polpair,{' at mjd '},...
11       mjdstring,{' SNR ~ '},num2str(snr)));
12 axis([lowfreq hifreq -Inf Inf])
13
14 subplot(2,1,2)
15 plot(y,phase,'rx',y,phase,'b-')
16 ylabel('Phase in radians');
17 xlabel(strcat({'Frequency '},num2str(lowfreq),'-',...
18      num2str(hifreq),{' MHz '}))
```

```
19 axis([lowfreq hifreq -pi pi])
```

LISTING A.2: MATLAB® Function file DiFXplot.m

A.4 Experiments

The following code would work within the directory where the *.lba* files for the v252-LBA files are found with the addition of a folder *DATA*, created. Thus with the MATLAB® directory at the root of the original *.lba* files we can use the function pulse2bit in the following way

```
>> pulse2bit(64,32,1,1)
```

Which will create the generated numbers for the last example in section 3.3,

```
1  function pulse2bit(l,w,tsize,sign)
2  close all
3  load dataSeed.mat
4  % The reference used are the MathWorks information
5  % about fseek and fread and the examples,
6  % http://www.mathworks.com/help/matlab/ref/fseek.html
7  % http://www.mathworks.com/help/matlab/ref/fread.html
8  % http://www.mathworks.com/help/matlab/ref/fwrite.html
9
10 % Open an the original .lba file in read mode
11 fid1 = fopen('v252f_Cd_161_012830_2bit.lba', 'r');
12 % Create 2 new files in an already created folder Data in write mode
13 fid2 = fopen('Data/v252f_Cd_161_012830.lba','w');
14 fid3 = fopen('Data/v252f_At_161_012830.lba', 'w');
15
16 % Buffer the first 4096 bytes which contain ASCII character
17 A = fread(fid1,4096,'uint8=>char');
18 %%%%%
19
20 % o is the length of the period
21 % l is for the number of zero 2-bit values
22 % w is for the number of the highest 2-bit value
23 o = l-w;
24 % pattern generates the 1 period of the pattern and
25 % sign is to be used to change the sign of the highest 2-bit value
26 pattern =  [sign*ones(1,w),zeros(1,o)];
27
28 %Copy first the buffered header of the original LBA file to the new file
```

```
29 fwrite(fid2, A);
30 fwrite(fid3, A);
31
32 % Then hopefully start injecting the generated data in the new files
33 tic
34  r = int8(zeros(1,1024*1000*10));
35  r = int8(repmat(pattern, 1, (length(r)/length(pattern))));
36  for i = 1:tsize
37  fwrite(fid2, r,'bit2');
38  fwrite(fid3, r,'bit2');
39  end
40  toc
41 % A loop is used to avoid having to deal with memory problems which will
42 % Sure arise if the size written in bytes is too large
43
44 % When the operation is finished close the files which are then ready
45 % To be used as pseudo files for the v252-LBA data set
46  fclose(fid2);
47  fclose(fid3);
48  fclose(fid1);
```

LISTING A.3: MATLAB® Function file pulse2bit to produce a .lba file with a square data pulse

To generate a random sequence of random numbers, with reference to the Mathworks documentation http://www.mathworks.com/help/matlab/ref/rng.html. A seed was produced, with the following command inside the MATLAB® command window.

```
>> dataSeed = rng
```

And the value of the seed was store in the same directory with the file name *dataSeed.mat* Thus the following code in listing A.4 could be used in the following way

```
>> random_2bit(1)
```

To generate the values used for the example with random 2-bit values in section 3.3.

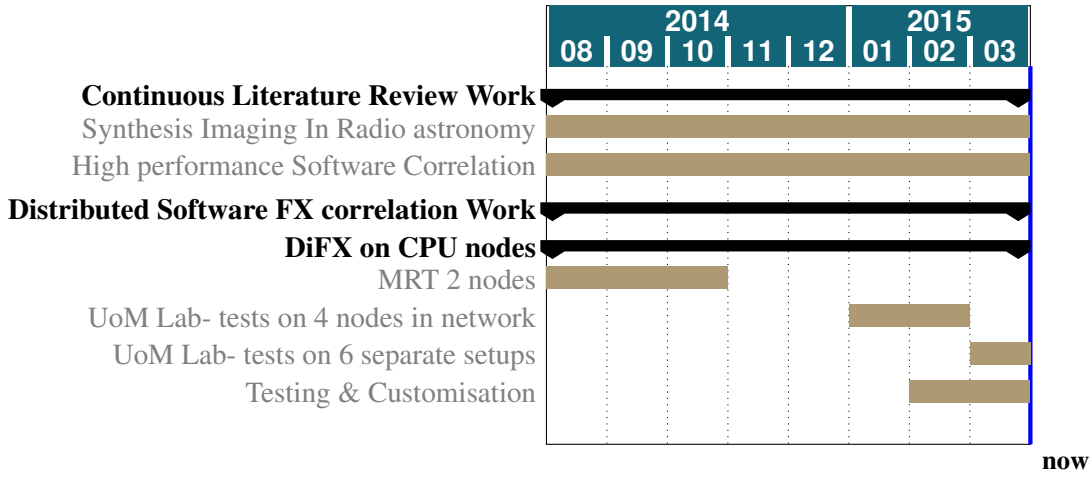
```
1 function random_2bit(tsize)
2 close all
3 load dataSeed.mat
4
5 fid1 = fopen('v252f_Cd_161_012830_2bit.lba', 'r');
```

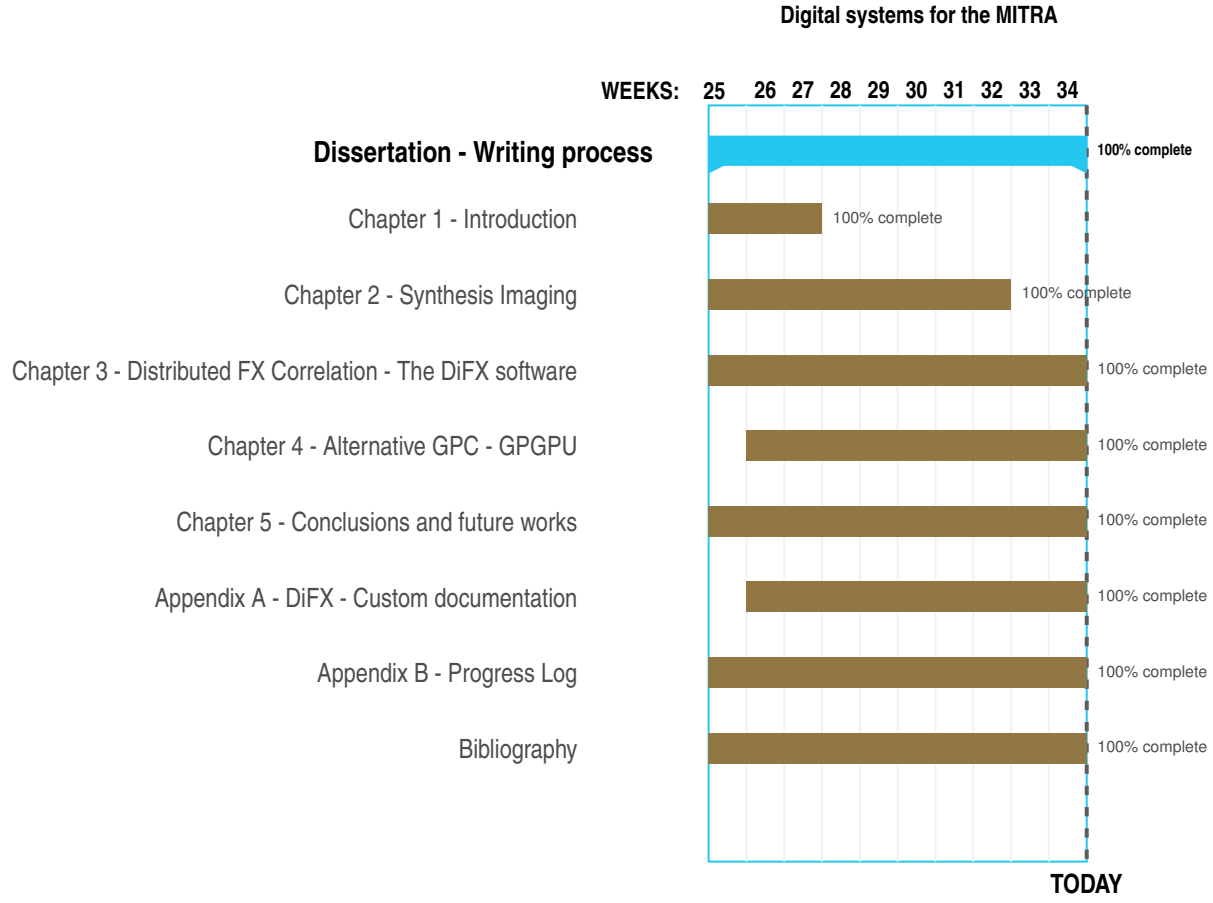
```
6
7  fid2 = fopen('Data/v252f_Cd_161_012830.lba','w');
8  fid2 = fopen('Data/v252f_Cd_161_012830.lba','w');
9  fid3 = fopen('Data/v252f_At_161_012830.lba', 'w');
10
11 A = fread(fid1,4096,'uint8=>char');
12 %Copy the header of an original LBA file to the new file
13 fwrite(fid2, A);
14 fwrite(fid3, A);
15  tic
16 rng(dataSeed); %Reset the Random Number Generator seed once
17  for i = 1:tsize
18  %Random number which can take the value 1, 0 or -1
19  r = randi([-1 1],1,1024*1000*10);
20  fwrite(fid2, r,'bit2');
21  fwrite(fid3, r,'bit2');
22  end
23  toc
24  fclose(fid2);
25  fclose(fid3);
26  fclose(fid1);
```

LISTING A.4: MATLAB® Function file random_2bit to produce a .lba file with a ramdom signed 2-bit data

Appendix B

Progress Logs




Digital systems for the MITRA
WEEKS:
25
26
27
28
29
30
31
32
33
34
Dissertation - Writing process
100% complete
Chapter 1 - Introduction
100% complete
Chapter 2 - Synthesis Imaging
100% complete
Chapter 3 - Distributed FX Correlation - The DiFX software
100% complete
Chapter 4 - Alternative GPC - GPGPU
100% complete
Chapter 5 - Conclusions and future works
100% complete
Appendix A - DiFX - Custom documentation
100% complete
Appendix B - Progress Log
100% complete
Bibliography
100% complete
TODAY

UNIVERSITY OF MAURITIUS

FACULTY ……OF SCIENCE……..

PROGRESS LOG

Student Name : LOUIS, Ruben Anderson

Student ID : 1210449

Department : Physics

Programme : BSc (Hons) Physics with Computing

Title of Dissertation: Digital Systems for the MITRA

Supervisor : Dr. Beeharry Girish Kumar

Project Coordinator: Dr. Bunwaree Mono Runjun

Your Progress Log serves as a record of your transferable skills and participation and attainment as a student for dissertation purposes.

Its purpose is to help you to plan your own dissertation and to record the outcomes.

As well as gaining valuable skills, you will find that the information accumulated in this Log will prove helpful during the write up of the dissertation.

The document belongs to you and it is your responsibility to keep it up to date.

It is your responsibility to ensure your supervisor is aware of the dissertation activities you have undertaken.

You should sign the appropriate statement below when you submit your Progress Log:

I confirm that the information I have given in this Log is a true and accurate record:

Signed: ……  …… Date: ..03/04/2015…..

RECORD OF STRATEGIC MEETINGS WITH SUPERVISOR

Meetings	Date	Topics/Themes Discussed	Comments (If any)	Supervisor's Initials	Student's Initials
1	12.08.14	General Purpose Computing in Radio Astronomy		GKB	R.L
2	13.08.14	Performance Scaling of General Purpose hardware		GKB	R.L
3	18.08.14	Distributed FX Correlation (DiFX)		GKB	R.L
4	18.09.14	DiFx on 2 nodes	Troubleshooting Setup	GKB	R.L
5	09.10.14	Synthesis Imaging	Radio Interferometry	GKB	R.L
6	09.02.15	Wide Field Polarised Interferometry		GKB	R.L
7	13.02.15	DiFX Data Recording Format Issue		GKB	R.L
8	03.03.15	DiFX test Setups and Performance		GKB	R.L

Supervisor Girishkumar Beeharry ..... Signature ..... Date 01.04.2015

N.B: Both the supervisor and the student should retain a copy of this Project Progress Log.
A copy of the duly filled and signed Progress Log should be **included and submitted in the section** 'Appendices' of the Dissertation.

Bibliography

Girish Kumar Beeharry, Stuart David MacPherson, and Gary Peter Janse Van Vuuren. Multifrequency interferometry telescope for radio astronomy (mitra): Science and technology. In *AFRICON, 2013*, pages 1–3. IEEE, 2013.

MA Clark, PC La Plante, and Lincoln J Greenhill. Accelerating radio astronomy cross-correlation with graphics processing units. *International Journal of High Performance Computing Applications*, page 1094342012444794, 2012. URL http://arxiv.org/pdf/1107.4264.

C CUDA. Programming guide: Cuda toolkit documentation, 2015.

Adam Deller. Difx developer pages, 2014. URL http://cira.ivec.org/dokuwiki/doku.php/difx/start. [Online; accessed 19-September-2014].

Adam T Deller, SJ Tingay, M Bailes, and C West. Difx: a software correlator for very long baseline interferometry using multiprocessor computing environments. *Publications of the Astronomical Society of the Pacific*, 119(853):318–336, 2007.

James Di Francesco. NAASC: A crash course in radio astronomy and interferometry: Aperture Synthesis. https://science.nrao.edu/science/meetings/ppt-presentation/jdf.webinar.2.pdf/at_download/file, 2014. Accessed: 29 October 2014.

Sabera Bibi Ginourie. A prototype front-end and backend receiver system for radioastronomy. BSc Dissertation, University of Mauritius, March 2010.

K Golap, N Udaya Shankar, S Sachdev, R Dodson, and Ch V Sastry. A low frequency radio telescope at mauritius for a southern sky survey. *Journal of Astrophysics and Astronomy*, 19(1-2):35–53, 1998.

Dominique Ingala and Stuart D MacPherson. An overview of the mitra radio telescope signal chain. In *AFRICON, 2013*, pages 1–5. IEEE, 2013.

Pritvi Jheengut. Software Correlation. BSc Dissertation, University of Mauritius, April 2008.

NRAO. Types of radio waves, 2014. URL https://public.nrao.edu/types-of-radio-waves. [Online; accessed 19 October 2014].

Chris Phillips. Lba disk recorder header format, 2005. URL ftp://ftp.atnf.csiro.au/pub/people/vlbi/evlbi/memos/lbadr-head.pdf. [Online; accessed 7 January 2015].

Nitisha Pirthee. DIGITAL BACK END FOR MITRA PROTOTYPE. BSc Dissertation, University of Mauritius, April 2013.

Nicolas Platel. Implémentation d'un corrélateur sur une carte gpu. Internship report, Mauritius Radio Telescope, 2010.

Simon Ratcliffe. Embedded Supercomputing: Radio Astronomy at the Limit. In *2015 GPU Technology Conference (GTC 2015)*, 2015. URL http://on-demand.gputechconf.com/gtc/2015/presentation/S5222-Simon-Ratcliffe.pdf.

A Richard Thompson, James M Moran, and George W Swenson Jr. *Interferometry and synthesis in radio astronomy*. John Wiley & Sons, 2008.

uomnews.wordpress.com. Multifrequency interferometry telescope for radio astronomy in mauritius, 2011. [Online; accessed 26 October 2014].

Wikipedia. Software-defined radio — wikipedia, the free encyclopedia, 2014. URL http://en.wikipedia.org/w/index.php?title=Software-defined_radio. [Online; accessed 19-October-2014].

David J. Wilner. NRAO: Imaging and deconvolution, 2014. URL https://science.nrao.edu/science/meetings/2014/14th-synthesis-imaging-workshop/lectures-files/wilner_siw2014.pdf. Accessed: 29 October 2014.

Andrew Woods, Michael Inggs, and Alan Langman. Accelerating a Software Radio Astronomy Correlator using FPGA co-processors. In *Symposium on Application Accelerators in High Performance Computing*, 2009. URL http://www.researchgate.net/publication/228407144_Accelerating_a_Software_Radio_Astronomy_Correlator_using_FPGA_co-processors/file/79e4150ad456b10386.pdf.

Andrew Woods, Michael Inggs, and Alan Langman. Accelerating Software Radio Astronomy FX Correlation with GPU and FPGA Co-processors. Master's thesis, University of Cape Town, October 2010. URL http://www.rrsg.uct.ac.za/members/drew/andrew_woods_msc.pdf.

www.skatelescope.org. What is radio astronomy, 2014. URL www.skatelescope.org/radio-astronomy. [Online; accessed 31 January 2014].