GPU-BASED DATA PROCESSING FOR 2D MICROWAVE IMAGING ON MAST

J.C. Chorley,[1] R.J. Akers,[2] K.J. Brunner,[1,2] N.A. Dipper,[1] S.J. Freethy,[2]
R.M. Sharples,[1] V.F. Shevchenko,[2] D.A. Thomas,[2,3] and R.G.L. Vann[3]

[1] *Centre for Advanced Instrumentation, Department of Physics, Durham University, Durham DH1 3LE*

[2] *Culham Centre for Fusion Energy, Abingdon, Oxfordshire, OX14 3DB*

[3] *York Plasma Institute, Department of Physics, University of York, York, YO10 5DD*

Abstract

The Synthetic Aperture Microwave Imaging (SAMI) diagnostic is a Mega Amp Spherical Tokamak (MAST) diagnostic based at Culham Centre for Fusion Energy. The acceleration of the SAMI diagnostic data processing code by a graphics processing unit (GPU) is presented, demonstrating acceleration of up to 60x compared to the original Interactive Data Language (IDL) data processing code. SAMI will now be capable of inter-shot processing allowing pseudo-realtime control so that adjustments and optimisations can be made between shots. Additionally, for the first time the analysis of many shots will be possible.

This work was presented at IAEA TM FDPVA Nice Proceedings, June 2015.

Keywords: GPU, CUDA, microwave, imaging

I. INTRODUCTION

For current and certainly next generation fusion devices such as the International Thermonuclear Experimental Reactor (ITER), specialised hardware and techniques need to be employed in data processing tasks as the amount of data produced in experiments becomes large and difficult to handle with traditional approaches. This work aims to assess the suitability of one such technology, the graphics processing unit (GPU) as applied to the Synthetic Aperture Microwave Imaging (SAMI) diagnostic installed on the Mega Amp Spherical Tokamak (MAST). SAMI acquires 4GB raw data per shot and an existing Interactive Data Language (IDL) data processing code which calculates the cross-correlations between antenna pairs takes approximately 20-30 minutes per shot to run and as such is unsuitable for inter-shot processing. To improve on this situation, a GPU-based Compute Unified Device Architecture (CUDA) code has been developed demonstrating a significant acceleration of the data processing, making it possible for inter-shot processing in future campaigns on National Spherical Torus Experiment (NSTX-U) and MAST-U and greatly improving the capabilities of the SAMI diagnostic on these machines. Further, the GPU code will enable data-mining of many MAST shots from previous campaigns which has until now been impossible due to the runtime of the existing IDL code. The benefits of using GPUs for processing large data sets relevant for next generation fusion diagnostics is clearly demonstrated by the new accelerated GPU CUDA code for the high data rate SAMI diagnostic and a cost/benefit analysis is presented to emphasise the advantages of a GPU-based approach to data processing.

The rest of this paper is organised as follows. Section II gives a review of novel hardware used in big data experiments and section III discusses the high data rate SAMI diagnostic. Section IV introduces the GPU and programming paradigm and section V describes the SAMI data processing GPU solution. Section VI discusses the acceleration achieved by the CUDA code and the accuracy of cross-correlations obtained. Section VII is a discussion of the benefits and limitations of the GPU approach and section VIII concludes.

II. REVIEW OF NOVEL HARDWARE USED FOR LARGE DATA-PROCESSING TASKS

We are entering a mode of operation for tokamaks in which large amounts of data are produced. Alternative technologies such as the GPU have become very popular for some highly computationally demanding tasks due to the increased floating-point compute power, greater memory bandwidth and better energy efficiency they provide compared to modern CPUs.[1] For example, multiplying two large, dense matrices on both a multi-core architecture using only Open Multi-Processing (OpenMP) and on a GPU shows that for larger problem sizes the GPU far outperforms the multi-core OpenMP implementation.[2] Solutions involving only OpenMP or only Message Passing Interface (MPI) do not perform as well as GPU implementations[3] and the field of computational science is turning more to accelerators like the GPU and Intel Many-Integrated Core (MIC), or even field-programmable gate arrays (FPGAs) to achieve the computational power required as we approach the exascale. FPGA programming requires mapping the problem to the FPGA architecture and resources, such that most of the resources available on chip are utilized. FPGAs can provide effective acceleration when presented with a constant stream of data to be processed which keeps each logic element working every clock cycle. This is the case for signal processing and FPGAs are currently utilized for fast filtering in logic on the MAST Thomson scattering system[4] and many other fusion diagnostics. FPGAs will be important for the Square Kilometer Array (SKA) which needs to process signals from multiple antennas[5] and FPGAs have demonstrated their importance in the trigger and data acquisition system of the ATLAS particle detector experiment at the Large Hadron Collider (LHC) with the discovery of the Higgs boson.[6] With problems such as EFIT, FPGAs are less useful and GPUs have been demonstrated to be better suited to the problem with P-EFIT (Ref. 7). Recently, the Maxeler dataflow engine[8] has gained popularity in the field of HPC on FPGAs. Problems are reformatted as a continuous flow of data into the dataflow engine and inputs are buffered to maintain a constant but high bandwidth data stream. However the field of high performance computing on FPGAs is in its infancy due to a lack of developed compilers and a complex development cycle[9] and it is accelerators like the GPU and MIC that are most popular in computational science. Indeed, at the time of writing, four of the top ten supercomputers in the Top500 list are hybrid machines[10] utilizing either GPUs or MICs as an accelerator or

coprocessor; it is clear this is the direction in which computational science is going. Often on these machines, it is a heterogeneous programming model featuring GPU, MPI and OpenMP that achieves the best performance.[11]

Large data experiments such as the SKA are looking towards alternative technologies and specialised hardware such as the GPU, in order to ease the big data problem. Beam forming in radio astronomy has higher performance and is more energy efficient on a GPU system compared to a multi-core CPU system[12] and a recent analysis[13] concluded that for SKA, novel hardware and system architectures need to be developed to achieve the required power efficiency and compute capabilities. The European Organization for Nuclear Research (CERN) and LHC have investigated the potential of using GPUs in the high-level trigger algorithms used to select the interesting data to reduce the raw data obtained from this experiment at the nominal LHC collision rate of 40MHz or every 25 nanoseconds.[14] For tokamaks such as NSTX, in 2004 camera data amounted to 300 MB/pulse, or 600 MB/pulse if all the data had been archived whereas shortly afterwards in 2006, one camera alone can acquire 2GB/pulse.[15] For next generation tokamaks such as ITER, between 160GB/pulse, extrapolating from existing devices, to 100TB/pulse if physics reasoning and predicted acquisition rates are considered will be produced.[16] It is therefore worthwhile to investigate the suitability of hardware like the GPU for massive data processing tasks such as those in real-time fast control systems,[17] the GPU being more developed than FPGAs in high performance computing.

III. SAMI DIAGNOSTIC

III.A. Data processing requirements

The SAMI diagnostic has the highest data rate on MAST. For a typical shot on MAST, the amount of other data produced amounts to approximately 120-140 MB and the images produced account for approximately 350 MB. SAMI scans over 16 frequency channels in the range 10 GHz-35.5 GHz and the signals are digitised by 14-bit ADCs sampling at 250 MSPS, giving a data rate of 8GB/s. So for a single 500ms shot on MAST 4GB raw data is acquired.[18] MAST aims for 30 shots a day meaning SAMI acquires 120 GB/day making SAMI a good test case for future high data rate diagnostics and the associated data handling.

III.B. Description of the diagnostic

The SAMI system installed on MAST is a phased array of eight linearly polarized Vivaldi antennas with a configuration that has been optimized to achieve maximum synthetic aperture efficiency satisfying both space and bandwidth requirements.[19] SAMI measures the intensity of microwave emission from the plasma. For typical MAST parameters, thermally born electrostatic electron Bernstein waves (EBWs) convert to electromagnetic waves via a two stage process in the plasma edge. This allows them to be detected outside of the plasma by SAMI. This radiation is emitted anisotropically and is coplanar with the density gradient and the magnetic field at the mode conversion surface. Since the density gradient is known, SAMI can be used to deduce the magnetic field line pitch. SAMI operates in two modes simultaneously: the passive imaging mode detecting spontaneous thermal emission, and the active probing mode measuring the back-scattered signal from an active probing source.[20] The data is acquired on two ADC boards (each digitizing eight channels) by two FPGA boards running embedded Linux for control. The data is streamed as a series of UDP packets over ethernet from the FPGAs via fibre optic cable to a file on the data storage computer. A schematic of the SAMI data acquisition system is presented in FIG. 1 showing the plasma on the left and both the active probing signal (indicated by the solid arrow) and spontaneous emission from the plasma (indicated by the dashed arrows). SAMI has produced the first ever 2D thermal electron Bernstein emission (EBE) maps of a plasma, identifying the location of B-X-O mode conversion windows in over-dense plasmas[21] and SAMI is the first diagnostic to measure magnetic pitch angle through simultaneous 2D Doppler backscattering.[22] FIG. 2 is the image reconstruction for shot 27022 at 260ms showing the brightness distribution or microwave intensity as a function of angular position. The location of the mode conversion windows is clearly identified. The images are reconstructed from the cross-correlations between each pair of antenna signals which are calculated by the data processing code post shot. According to van Cittert-Zernike theorem,[23,24] these cross-correlations are samples of the Fourier transform of the brightness distribution in front of the antenna array. An image of the plasma can be reconstructed by performing an inverse Fourier transform to give an approximation to the real source brightness distribution.

III.C. Data processing

The raw data SAMI collects needs extensive processing to: *i.*) correct for phase drift between I and Q components, *ii.*) correct for phase differences between antennas due to RF electrical lengths and *iii.*) perform sideband separation. Better sideband suppression is obtained if the phase dispersion between I and Q signals introduced by the IF signal cables and filters is corrected first. Once the sidebands are successfully separated, the phase difference between antenna channels can be corrected. This phase is affected by differences in path lengths giving rise to unknown phases between the RF signals. Once the raw data has been corrected for these effects, the cross-correlations between each antenna pair are calculated from which the images can be formed. This process is shown schematically in FIG. 3 with the most significant system bottlenecks enclosed in circles. The Fourier filter operations and the cross-correlation steps dominate the data processing.

The numerical operations carried out in the data processing chain are presented in TABLE I for a typical MAST shot and the data is processed as a series of many independent vectors. At the start of the processing chain we have 16 real signals and for a typical MAST shot 16 frequency channels and 3119 time sweeps containing 2500 time points. After extracting the middle of each time sweep to remove noise introduced by switching frequency channel, each time sweep contains 2000 points. With sideband separation 16 real signals get converted to 8 complex signals and there are now two data arrays for upper and lower sideband. There are $\texttt{nAnt} \times (\texttt{nAnt} - 1)/2 = 28$ unique cross-correlations to calculate where $\texttt{nAnt}$ = number of antennas, and therefore there are 28 associated baselines. The image is then formed by aperture synthesis as the sum of the products of antenna cross correlations and associated basis functions.

IV. GRAPHICS PROCESSING UNITS

Since 2006, with the introduction of the GeForce 8800, GPUs have been more readily adopted by the scientific community for high performance computing. This is mostly due to the effort made by companies like Nvidia in the development of languages and programming paradigms such as CUDA which make programming GPUs more accessible to the scientific community.[25,26] Prior to this, it was difficult to write programs to carry out scientific

computation on a GPU because non-graphics computations needed to be expressed with a graphics API such as OpenGL.[27]

A GPU is attached to a CPU via PCIe (see FIG. 4) and expensive parts of a computation are offloaded to the GPU. GPUs work by making massive use of long vector units. In software, each CUDA thread is mapped to one element of a hardware vector unit. A full vector unit is 32 threads, or a warp, and a warp is processed in parallel by the hardware. GPUs are essentially designed to do massive parallel computations and provide the largest acceleration for data parallel problems with all threads executing the same instructions on different data. Many problems in computational science such as n-body problems, collision detection, probabilistic Potts model simulations and Cellular Automata simulations, are data parallel and ideal for acceleration with a GPU.[29] The larger problem can be broken down into many independent smaller problems which can be solved simultaneously with a single instruction operating on multiple data (SIMD). Iterative solvers where the next iteration depends on results from the previous iteration do not follow the SIMD model and as such are poor candidates for acceleration with a GPU.

V. GPU DATA PROCESSING CODE

V.A. Motivation for GPU code

For the M8 (2011 and early 2012) and M9 (2013) campaigns on MAST, raw data for approximately 3000 shots was obtained, requiring a 12TB RAID data storage system. On this system, with an AMD Phenom(TM) II X2 560 processor, it takes approximately 30 minutes to calculate the cross-correlations for one shot with the existing IDL code. MAST operates on a cycle of 8 hours on, 16 hours off and aims to carry out 30 shots per day. Therefore with the IDL code, SAMI data could be processed in 15 hours. In reality, the raw data for SAMI was not processed overnight as it was acquired but was stored for processing at some later date. Despite the length of data acquisition on MAST being relatively short (a MAST pulse is 500 milliseconds) and the time between consecutive pulses on MAST being relatively long (between 15 and 20 minutes in most cases), the IDL data processing code has a significantly long runtime. For devices like MAST-U and future devices with much longer pulse lengths, this processing time will increase further so it is essential to dramatically speed

up the processing time. The accelerated GPU code has enabled overnight data processing. More impressively, data can be processed between shots on MAST moving into a new regime of inter-shot data processing.

Given the long runtime of the IDL code, processing the raw data for many shots and performing many-shot analysis has previously been impossible. The SAMI multi-shot cross-correlation data could help derive scaling laws; for example early analysis indicates a dependence between electron Bernstein wave power and D-alpha emission so it is essential to investigate many shots to find correlations between plasma parameters. Many-shot analysis has previously proved to be vital in the derivation of scaling laws such as ITER98Y2 (Ref. 30) which is now the commonly used model for ITER design. This is based entirely on empirical scaling from regression analysis of data in the ITPA (Ref. 31) database which includes a large number of shots from a number of different tokamaks.

V.B. Suitability of SAMI for GPU acceleration

The suitability of the SAMI data processing code for parallelization by a GPU and CUDA is evident as shown in FIG. 5. The data is organised as a series of vectors of length `nInt` for each of the 8 antennas and each of the 16 frequency channels, `nf`, which are switched between `nSweeps` times. Essentially, the time series is split into `nSweeps` blocks of `nInt` points and each block of `nInt` points is operated on identically. As seen previously in TABLE I, for a typical MAST shot where `nInt` = 2000, `nf` = 16 and `nSweeps` = 3119, the numerical computations consist of almost 800,000 vector operations of length 2000 elements being operated on identically, and the cross-correlation calculation consists of almost 2,800,000 vector operations. This is a SIMD scenario which is perfect for parallelization on a GPU by CUDA. Threads are grouped into thread blocks on the software level, typically each thread block represents a new vector and threads in the thread block represent individual data points in the vector. CUDA kernels are then launched with multiple thread blocks and multiple threads per block to process a lot of the data simultaneously. A typical kernel launch would include 32 thread blocks each with 256 threads.

V.C. Description of the GPU code

The raw data is stored in binary files where the frequency channels are multiplexed in time. The `read_bin_raw_gpu.cu` function reads in the data. The different frequency components are demultiplexed by looping over each frequency channel, `nf`, and sweep, `nSweeps`, reading `nInt*nAnt` points from the two data files (one for channels 0-7, one for channels 8-15) and placing it in the correct location in the data array for efficient processing. Since the switching period and frequency order may vary from shot to shot a `bootconfig.rfctrl.ini` file is consulted which informs where to start reading the binary file from on each iteration. The data is then copied to the GPU and the necessary signal processing tasks and data conditioning are performed. The noise of the local oscillator switch is removed, a smoothing box-car average is performed on each set of `nInt` points, the middle of each set of `nInt` points is taken to avoid any residual switching noise and unsmoothed subtraction and the data corresponding to channel 0, channel 1, channel 8 and channel 9 is shifted to correct for ADC timing errors. The data is then filtered with a bandpass and notch filter to remove some unwanted signals from each block of `nInt` and the IF dispersion between I and Q components caused by cable lengths is corrected. Performing sideband separation converts 16 real signals into 8 complex signals and another filter is performed for upper and lower sideband with calibration data correcting for phase offsets and balancing amplitudes between the I and Q components. Calibration of the RF phase is also performed to correct the phase drifts in the RF channels after sideband suppression. Finally, once these corrections have been made, the cross-correlations between the signals for each antenna pair, frequency sweep and upper and lower sideband are calculated. This process is illustrated in FIG. 6. In total, 14 CUDA kernels were constructed to process the data and the CUFFT library was used to perform Fourier transforms.

As indicated in FIG. 3, the most computationally expensive parts of the program are performing the Fourier filtering, and the cross-correlation calculation itself. However, a lot of the initial data conditioning, as operations on a set of vectors, is ideal to target with a GPU. The data is therefore transferred to the GPU as soon as possible, all the processing performed on the GPU and the result copied back to the CPU once all the processing has completed, with no intermediate data traffic. Data movement is often the bottleneck to hybrid CPU-GPU computation.

However, there is a problem with this model for the SAMI computation and for any large data problem such as those expected in the next generation of fusion diagnostics. GPUs have limited memory available and very few models have enough to process an entire 4GB shot at once. Each SAMI shot is 4GB of 14 bit (or two byte) integers. Immediately in software, the amount of data is doubled as each two byte integer is converted to a four byte float to perform scientific computation, so each SAMI shot is actually an 8GB data processing problem. It is necessary for big data problems like SAMI to carve the data up into chunks and process each, exploiting CUDA streams and concurrency which means to overlap a memory copy to the GPU with kernel execution on the GPU. The GPU can be effectively kept busy by copying the next chunk of data to the GPU whilst computing on the previous chunk. The high-end GPUs such as the Tesla K40C do even better as these GPUs have two copy engines to facilitate bi-directional memory copies. The resulting chunk_{i-1} can be copied from the GPU, whilst computation is being performed on chunk_i and the copying of chunk_{i+1} to the GPU is occurring simultaneously. This advantage is illustrated in FIG. 7 for a idealized scenario where the copy time and kernel execution time is assumed to be equal. Instead of having a single CUDA stream where successive data chunks are processed serially (whilst the data in each chunk is processed in parallel), if there are multiple CUDA streams, each with their own instruction queue, memory copies and kernel execution can be overlapped between streams and the time taken to process the data can be significantly reduced. There is a balance between the number of streams used and the size of each data chunk, as each stream needs its own memory to hold different data chunks simultaneously but ideally as much data as possible should be processed at once so there are fewer data chunks. For example, with SAMI, if the whole 8GB (which doesn't fit in the GPU memory) was carved up into four data chunks each having size 2GB and there were four CUDA streams processing the data, then the memory requirements would still be 8GB and this scenario would not work as the GPU does not have enough memory for all of the streams. In practice, SAMI used three CUDA streams and the size of each data chunk depends on the parameters `nInt`, `nf` and `nSweeps` which vary from shot to shot due to the switching period and frequency order varying from shot to shot. Typically the data would be carved up into many smaller chunks and the number of data chunks would be on the order of 100 chunks.

VI. ACCELERATION RESULTS AND ACCURACY

Code development was carried out on a machine with an Intel (R) Xeon(R) CPU E5-2670 @ 2.60 GHz with a Tesla K40C GPU with 12GB GDDR5 and PCIe 3.0 x16 lanes, which is independent to the SAMI data storage machine. The data for a few shots was made available on this machine to verify correctness and obtain some preliminary acceleration results, shown in TABLE II. The original IDL code averaged a time of 17 minutes and 18 seconds to calculate the cross-correlation data for a full shot. The serial C version as an intermediate step to the CUDA version was able to complete in 7 minutes and 44 seconds, giving an acceleration of 2.2x over the IDL. The raw data for both the IDL and C implementations was loaded in `/dev/shm` shared memory in the form of a RAM disk. Accelerating further with CUDA on the Tesla K40C gave a time of just 17 seconds, an acceleration of 26x over the serial C and 59x over the original IDL. The total run time of 17 seconds for the Tesla K40C is for raw data loaded in `/dev/shm`. If the data is retrieved from the hard drive, the run time of the code is significantly increased to 76 seconds. After demonstrating this acceleration on the Tesla K40C, a dedicated GPU card was obtained for SAMI, a GeForce GTX770 with 4GB GDDR5. Running the code and retrieving the raw data from hard disk gives a runtime of 70 seconds for one shot and if we mount the data in a RAM disk, again the run time is significantly reduced to 25 seconds which is comparable to the Tesla K40C system. It is evident that as the GPU processes the data so quickly, retrieving the data from hard disk creates a serious bottleneck.

The CUDA times given in TABLE II are the total time taken for the code to run which can be further split up into CPU time and GPU time as shown in TABLE III for the GeForce GTX770. The time taken to read the raw data dominates the run time. The GPU time, including memory copies to and from the GPU, is consistent between the two cases but if the data is mounted in a RAM disk, the time taken to read all of the raw data from file is significantly reduced by 3.7x. The unaccounted for time contributing to the total run time is due to the setup of the correction data and filter functions used to calibrate and condition the data which is shot dependent. The GeForce GTX770 on the SAMI system then cycled through SAMI data for 1837 shots in 30 hours, averaging a total run time of 58 seconds per shot.

Looking at TABLE IV and TABLE V the benefit of using CUDA streams and overlapping

kernel execution and memory copies is evident. TABLE IV shows the execution times for a single stream implementation and multi-stream implementations on the Tesla K40C. For a single stream, the total GPU time, including memory copies is 12 seconds where as if multiple CUDA streams are used the GPU time is nearly halved to between six and seven seconds. In this case, increasing the number of streams further provides no reduction to the GPU time. TABLE V shows the division of GPU time for the single stream case. The GPU time is dominated by the kernel execution and copying the result from the GPU is fast as we are reducing the amount of data by calculating the cross-correlations by two orders of magnitude. By referring to the multi-stream times in TABLE IV it is clear that successful overlap of the memory copies has been achieved as the latency of the memory copies is hidden by the kernel execution time. The total GPU time including memory copies for multi-stream implementations is approximately equal to the kernel execution time for a single CUDA stream.

Having achieved a nearly 60x acceleration over the IDL, it is important to ensure the CUDA calculation is accurate. Using shot 27022 as an example, the cross-correlations were calculated by the IDL code and the CUDA code respectively. FIG. 8 shows the images of microwave emission on MAST at 13GHz and 320ms reconstructed from the cross-correlation data. The images produced for the IDL calculation and the CUDA calculation match well. In fact, the absolute error between the IDL cross-correlations and the CUDA cross-correlations is less than 10^{-8} and the relative error is less than 10^{-4}. The relative error is greater than the absolute error as some of the cross-correlations are smaller than machine precision.

VII. DISCUSSION

VII.A. Cost-performance analysis

It is important to emphasise that in TABLE II the C version is a serial implementation, utilizing a single core on the multi-core CPU. Perhaps a fairer comparison would be to compare a parallel implementation utilizing all eight cores of the E5-2670 CPU using OpenMP. Assuming perfect scaling, using eight cores would give a time of 58 seconds and using 16 threads, two per core with Intel's hyper-threading would give a time of 29 seconds which is competitive with both GPU implementations. However, it is unlikely this kind of

perfect scaling would occur and the multi-core times would be worse than this in reality so the GPU implementation maintains the best performance. Looking at the cost of each device, at launch in October 2013 the Tesla K40C with 12GB GDDR5 was $7699 and has a double precision peak performance of 1.4 TFlops where as the E5-2670 had a launch price in March 2012 of $1552 and peak performance of 166.4 GFlops based on a clock speed of 2.6 GHz, eight cores and eight instructions per cycle. In addition, the cost of 12GB DDR3 RAM must be included to the E5-2670 processor to make this a fair comparison; four 3GB cards can be purchased for approximately $80. Therefore, the Tesla K40C has a slightly better performance per dollar of 0.181 GFlops/$ compared to the E5-2670 system which has a performance per dollar of 0.102 GFlops/$. However, since the launch of each device the price of the Tesla K40C has fallen considerably and if it was purchased today, it would cost approximately $4600 where the cost of the E5-2670 has remained roughly constant and only fallen by $100 or so. Therefore, at current prices, the Tesla K40C GPU has a much better performance per dollar of 0.298 GFlops/$. For SAMI, as the amount of resulting data being copied from the device has been reduced, the benefit of moving to the high-end Tesla K40C with bi-directional memory copies is small as can be seen by comparing the GPU run times of the Tesla K40C and GeForce GTX770 in TABLE III and TABLE IV. The performance gain by moving to the Tesla K40C is only one second. However, the GeForce GTX770 with 4GB GDDR5 costs approximately $460, an order of magnitude less than the Tesla K40C, so the performance per dollar for the GeForce GTX770 is an order of magnitude greater in the SAMI case.

VII.B. Further improvements to the code

To increase performance further, work needs to be done on reducing the CPU time and how the code accesses the raw data. It would be better to read all the data at once rather than looping over each `nf` and `nSweeps` and demultiplexing the data. A simple program was created to read all of the data from each raw data file in a single call to `fread` and the time taken for this single read is 1.66 seconds, a significant reduction on the 12.95 seconds it takes to do the looped read. This significantly reduces the total run time of the code on the Tesla K40C system to approximately eight seconds and similar reductions can be expected for the GeForce GTX770 SAMI system. Of course, the data would then need to be re-arranged into

an efficient order for processing on the GPU.

The GPU processing time could also be improved by examining the benefit of a multi-GPU system, where different GPUs process different chunks of the data simultaneously. As the data is carved up into multiple chunks currently and the chunks are processed essentially serially (except for the added parallelism provided by using multiple CUDA streams and concurrency) by a single GPU, the next logical step would be to demonstrate multi-GPU parallelism. For example, if there was one GPU, six data chunks and three CUDA streams, the data would be processed as illustrated in FIG. 7 (b) but if there were two GPUs, each with three CUDA streams, the first three chunks could be processed by the first GPU and the last three chunks could be processed simultaneously by the second GPU, significantly reducing the GPU compute time of the application. This could easily be achieved in software by simply looping over the number of GPUs, selecting the GPU to switch between contexts and executing the same code on each GPU with different data chunks. As seen from TABLES III, IV and V, the kernel execution time is between six and seven seconds and the total run time of the code is dominated by data movement which will still be a limiting factor in a multi-GPU implementation.

Features provided on GPUs with the highest compute capability such as dynamic parallelism (compute capability 3.5) may further accelerate the SAMI data processing code. The SAMI GeForce GTX770 has a compute capability of 3.0 and can not implement dynamic parallelism. The SAMI GPU code could be profiled with Nvidia Visual Profiler to identify hotspots and areas to focus further acceleration efforts on. The acceleration provided by the SAMI GPU code is sufficient for the current SAMI data processing requirements but these features could be exploited in the future to achieve further acceleration.

VII.C. Limitations of GPU approach

Dividing large datasets up and using CUDA streams is a solution to the limited GeForce GTX770 4GB GDDR5 memory and similarly, using multiple GPUs alleviates this issue of limited RAM. Indeed, even the higher end Tesla card with 12GB GDDR5 was limited for SAMI and many other big data problems such as those discussed in the introduction and next generation big data fusion diagnostics will benefit from utilizing CUDA streams or multiple GPUs. The next generation of Nvidia GPUs, Pascal GPUs, are expected to have

more GDDR5, up to 32GB which will significantly improve on this situation.

For scientific applications, it is essential to have confidence in results of calculations and GPUs have the option to enable ECC memory to ensure accuracy. With ECC on, some of the available memory is used for the ECC bits, 6.25% in the Tesla K40C, further reducing the limited available memory but protection against memory corruption errors is gained.

Additionally, the benefit provided by using a RAM disk has been demonstrated which significantly reduces the CPU time of the application compared to using the hard disk where the data is stored. The advantage gained by using a GPU to process the data is not as great if data is retrieved from hard disk as the time taken to do this dominates the total run time.

VII.D. Alternative approaches

Reducing the run time of the code to reach the real-time scale for SAMI data processing would be beneficial to real-time identification of the location of B-X-O mode conversion windows in over-dense plasmas such as those in MAST and is essential for potential future multi-megawatt electron Bernstein wave (EBW) current drive and heating systems.[32,33] Further acceleration of the GPU based SAMI data processing code could potentially make this possible. If the data could be streamed directly from the acquiring FPGAs to the GPU significant speed gains could be achieved. The direct streaming from a GPU to an FPGA has been demonstrated.[34] Alternatively, the field of HPC on FPGAs, whilst still in its infancy, is gaining in popularity and this provides potential for reaching the real-time data processing scale for SAMI. Similar cross-correlation computation has been done on an FPGA for a passive millimetre wave aperture synthesis imager[35] so there is potential the SAMI data processing could be done on FPGA technology. The SAMI data processing chain is well suited to being recast onto the Maxeler hardware discussed in section II and this is an area to potentially investigate in the future once the field has become well established in the scientific community.

VIII. CONCLUSION

The benefit of an accelerated GPU data processing code for the SAMI diagnostic has been successfully demonstrated which will enable the analysis of cross-correlation data for

many shots. In the future, inter-shot processing will be able to be performed on NSTX-U and MAST-U. The GPU code has fulfilled the current SAMI data processing requirements.

The suitability of a GPU data processing code has been assessed for SIMD data problems using the SAMI diagnostic as a test case and the assessment of the cost-performance benefit of a GPU solution has been provided. In this case the runtime of the SAMI data processing code has been accelerated almost 60x and the ability to perform multi-shot analysis of the SAMI data, previously impossible due to the computational power required to process the data and essential to find correlations between plasma parameters has been provided. It is likely large data processing tasks for other diagnostics will experience these benefits with a GPU processing code. The usefulness and desirability of data processing with a GPU for the next generation of tokamak and diagnostic operation where large amounts of data will be produced has been illustrated.

ACKNOWLEDGEMENTS

This work is funded by EPSRC grant EP/K504178 for the Fusion Doctoral Training Network, the SAMI grant EP/H016732, the UK Magnetic Fusion Research Programme EP/I501045 and the EUROfusion pedestal grant ER-WP15_CCFE-03. This work has been carried out within the framework of the EUROfusion Consortium and has received funding from the Euratom research and training programme 2014-2018 under grant agreement No 633053. The views and opinions expressed herein do not necessarily reflect those of the European Commission.

REFERENCES

[1] L. YANG et al., "High Performance Data Clustering: A Comparative Analysis of Performance for GPU, RASC, MPI and OpenMP Implementations", *The Journal of Supercomputing*, **70**, 284 (2014)

[2] K. THOUTI and S.R. SATHE, "Comparison of OpenMp and OpenCL Parallel Processing Technologies", *International Journal of Advanced Computer Science and Applications*, **3**, 56 (2012)

[3] E. MONTEIRO et al., "Parallelization of Full Search Motion Estimation Algorithm for Parallel and Distributed Platforms", *International Journal of Parallel Programming*, **42**, 239 (2014)

[4] G.A. NAYLOR, "An FPGA based control unit for synchronization of laser Thomson scattering measurements to plasma events on MAST", *Fusion Engineering and Design*, **85**, 280 (2010)

[5] P. THIAGARAJ et al., "FPGA processing for High Performance Computing", *Emerging Technology Conference*, Manchester, 30th June - 1st July (2015)

[6] N. GARELLI, "The Evolution of the Trigger and Data Acquisition System in the ATLAS Experiment", *Journal of Physics: Conference Series*, **513**, 012007 (2014)

[7] X.YUE et al., "Fast equilibrium reconstruction for tokamak discharge control based on GPU", *Plasma Physics and Controlled Fusion*, **55**, 085016 (2013)

[8] O. PELL and V. AVERBUKH, "Maximum Performance Computing with Dataflow Engines", *Computing in Science and Engineering*, **14**, 98 (2012)

[9] J.D. CAPPELLO and D. STRENSKI, "A Practical Measure of FPGA Floating Point Acceleration for High Performance Computing", *IEEE 24th International Conference on Application-Specific Systems, Architectures and Processors*, **24**, 160 (2013)

[10] "Top500 Lists June 2015", www.top500.org/lists/2015/06 (June 2015)

[11] C. XU et al., "Collaborating CPU and GPU for large-scale high-order CFD simulations with complex grids on the Tianhe-1A supercomputer", *Journal of Computational Physics*, **278**, 275, (2014)

[12] A. SCLOCCO et al., "Radio Astronomy Beam Forming on Many-Core Architectures", *IEEE 26th International Parallel and Distributed Processing Symposium*, **26**, 1105 (2012)

[13] E. VERMIJ et al., "Challenges in exascale radio astronomy: can the SKA ride the technology wave?", *The International Journal of High Performance Computing Applications*, **29**, 37 (2015)

[14] P. LUJAN et al., "GPU Enhancement of the Trigger to Extend Physics Reach at the LHC", *Journal of Physics: Conference Series*, **513**, 012019 (2014)

[15] W.M. DAVIS et al., "Storage and analysis techniques for fast 2-D camera data on NSTX", *Fusion Engineering and Design*, **81**, 1975 (2006)

[16] J.B. LISTER et al., "The ITER project and its Data Handling Requirements", *Proceedings of the International Conference on Accelerator and Large Experimental Physics Control Systems*, **9**, 589 (2003)

[17] R. CASTRO et al., "Soft real-time EPICS extensions for fast control: A case study applied to a TCV equilibrium algorithm", *Fusion Engineering and Design*, **89**, 638 (2014)

[18] B.K. HUANG et al., "FPGA-based embedded Linux technology in fusion: The MAST microwave imaging system", *Fusion Engineering and Design*, **87**, 2106 (2012)

[19] S.J. FREETHY et al., "Optimization of wide field interferometric arrays via simulated annealing of a beam efficiency function", *IEEE transactions on antennas and propagation*, **60**, 5442 (2012)

[20] S.J. FREETHY et al., "Lensless passive and active microwave imaging on MAST", *Plasma Physics and Controlled Fusion*, **55**, 124010 (2013)

[21] V.F. SHEVCHENKO et al., "Synthetic aperture microwave imaging with active probing for fusion plasma diagnostics", *Journal of Instrumentation*, **7**, P10016 (2012)

[22] D.A. THOMAS et al., "2D Doppler backscattering using synthetic aperture microwave imaging of MAST edge plasmas", *Nuclear Fusion*, **56** 026013 (2016)

[23] P.H. VAN CITTERT, "Die Wahrscheinliche Schwingungsverteilung in Einer von Einer Lichtquelle Direkt Oder Mittels Einer Linse Beleuchteten Ebene", *Physica*, **1**, 201 (1934)

[24] F. ZERNIKE, "The concept of degree of coherence and its application to optical problems", *Physica*, **5**, 785 (1938)

[25] J.D. OWENS et al., "GPU Computing", *Proceedings of the IEEE*, **96**, 879 (2008)

[26] E. WYNTERS, "Parallel processing on Nvidia graphics processing units using CUDA", *Journal of Computing Sciences in Colleges*, **26**, 58 (2011)

[27] J. NICKOLLS and W.J. DALLY, "The GPU Computing Era", *IEEE Computer Society*, **30**, 56 (2010)

[28] NAG, *Accelerating Applications with CUDA*, **CCFE**, Nov (2013)

[29] C.A. NAVARRO et al., "A Survey on Parallel Computing and its Applications in Data-Parallel Problems Using GPU Architectures", *Communications in Computational Physics*, **15**, 285 (2014)

[30] "ITER Physics Basis", *Nuclear Fusion*, **39**, 2175 (1999)

[31] N.W. EIDIETIS et al., "The ITPA disruption database", *Nuclear Fusion*, **55**, 063030 (2015)

[32] J. URBAN et al., "A survey of electron Bernstein wave heating and current drive potential for spherical tokamaks", *Nuclear Fusion*, **51**, 083050 (2011)

[33] V.F. SHEVCHENKO et al.,"Two dimensional studies of electron Bernstein wave emission in MAST", *Fusion Science and Technology* **59**, 663 (2011)

[34] R. BITTNER et al., "Direct GPU/FPGA communication Via PCI express", *Cluster Computing-The Journal of Networks Software Tools and Applications*, **17**, 339 (2014)

[35] N.A. SALMON et al., "Minimising the costs of next generation aperture synthesis passive millimetre wave imagers", *SPIE Proceedings on Millimetre Wave and Terahertz Sensors and Technology IV*, **8188**, (2011)

LIST OF FIGURES

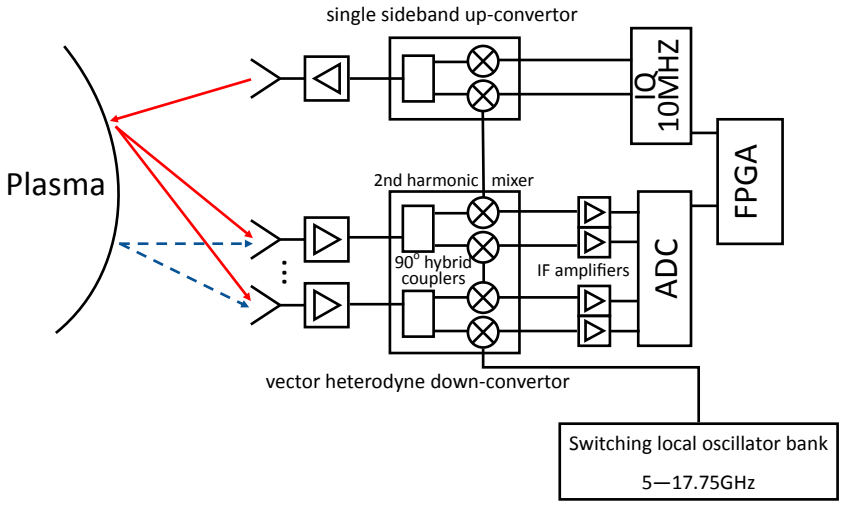


FIG. 1. A schematic of the SAMI data acquisition system showing both active probing (solid arrows) and passive imaging (dashed arrows) modes.

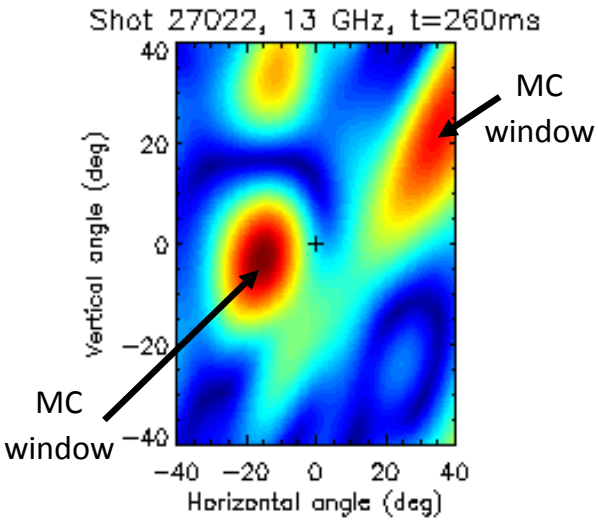


FIG. 2. The typical image reconstruction showing the brightness distribution for shot 27022 at 13 GHz and 260ms. The location and size of B-X-O mode conversion (MC) windows are clearly identified by peaks in the brightness distribution.

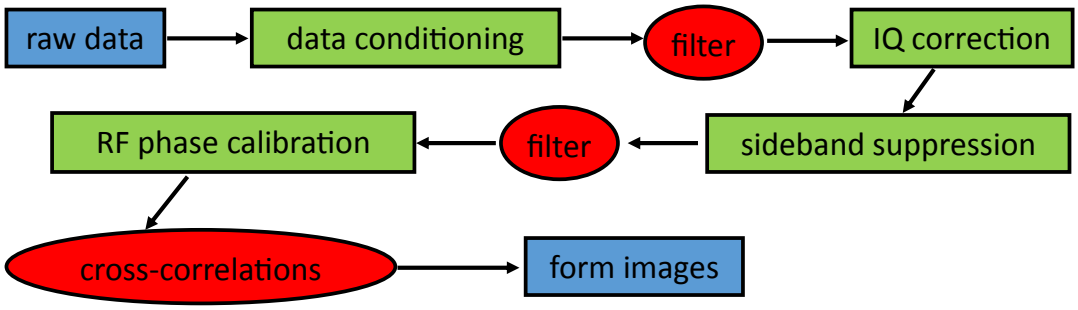


FIG. 3. The data processing chain. Computationally expensive calculations are enclosed in circles and refer to the Fourier filter and cross-correlation steps.

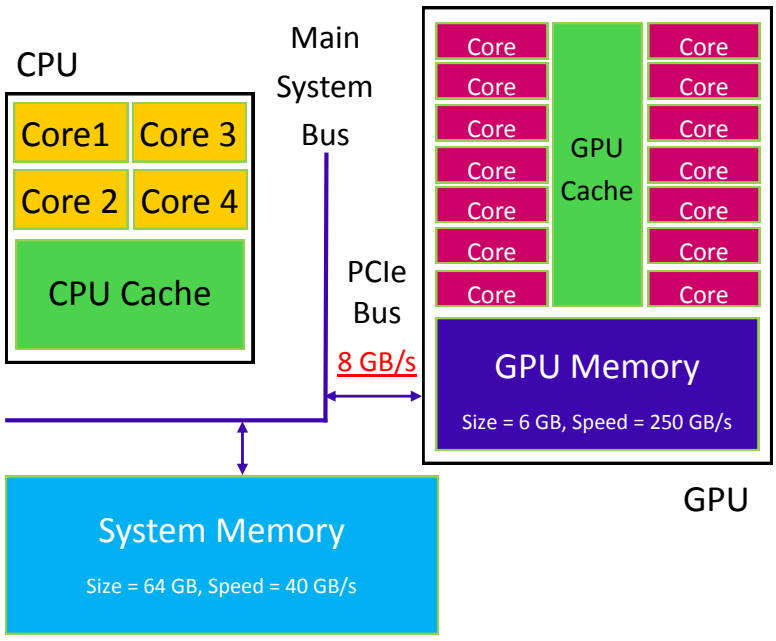


FIG. 4. A schematic of a hybrid CPU-GPU system adapted from Ref. 28.

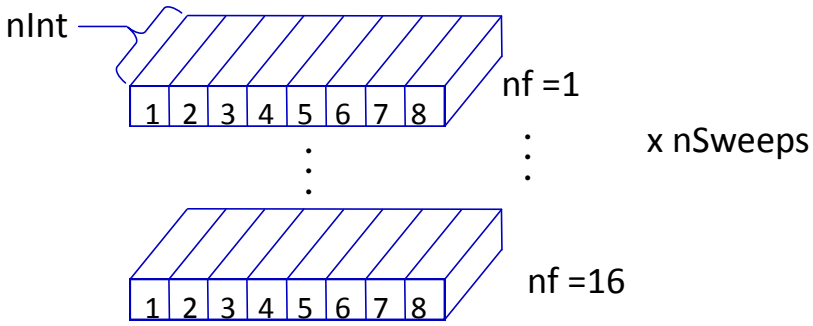


FIG. 5. A schematic of the SAMI data structure highlighting the SIMD nature. The data is processed as a series of vector operations on vectors of size `nInt`.

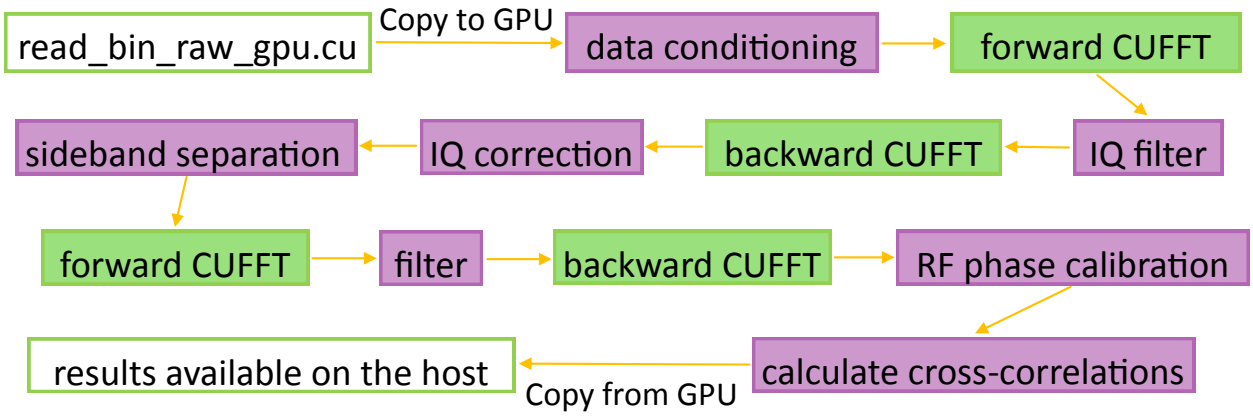
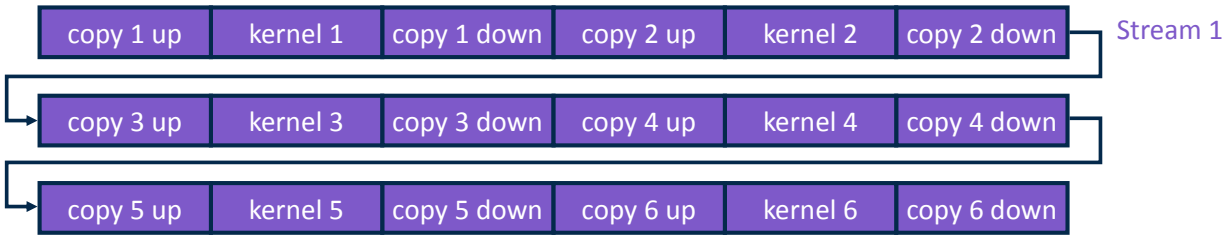
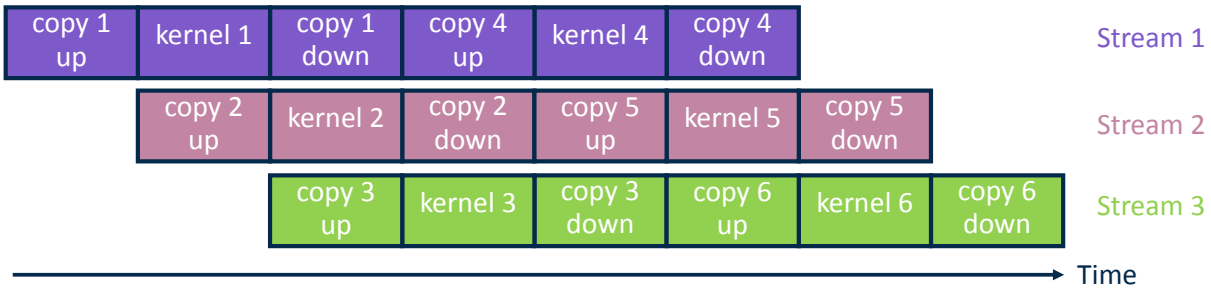


FIG. 6. A schematic of the SAMI CUDA data processing code showing an initial data copy to the GPU, all data processing performed on the GPU and finally a resulting data copy back to the CPU.

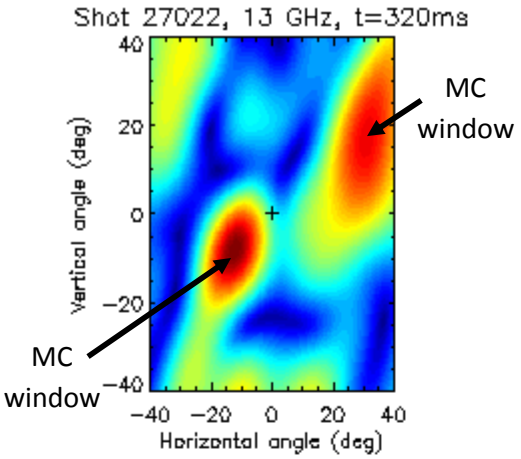


(a) The order of execution with a single CUDA thread.

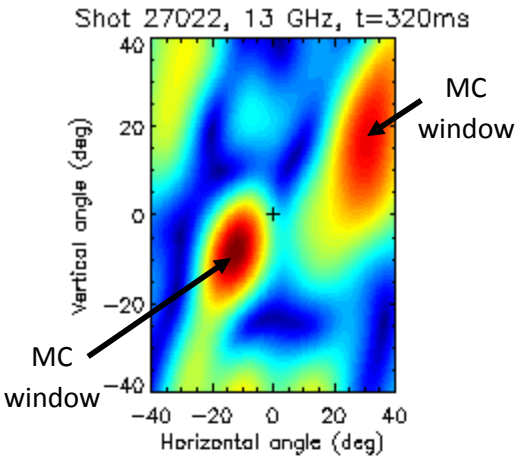


(b) The benefit of exploiting three CUDA streams and concurrency is illustrated.

FIG. 7. A schematic showing (a) the order of execution using a single CUDA thread where the total time to process the data is 18 units of time, and (b) using three CUDA streams to process the data reduces the units of time required to process the data to eight.



(a) Image reconstructed from the cross-correlations calculated by the IDL code.



(b) Image reconstructed from the cross-correlations calculated by the CUDA code.

FIG. 8. The images reconstructed for shot 27022 at 13GHz and 320ms for (a) the IDL code and (b) the CUDA code . The images are in good agreement with relative error less than 10^{-4}.

LIST OF TABLES

TABLE I. Numerical operations and typical size

PHYSICS OPERATION	MATHEMATICAL OPERATION	TYPICAL VECTOR LENGTH	TYPICAL NUMBER OF OPERATIONS
data conditioning	box-car average smooth	2500	798464
	extract middle	2500 ->2000	798464
	shift (not applied to all channels)	2000	199616
filter	FFT	2000	798464
	bandpass filter	2000	798464
	IQ filter	2000	798464
	inverse FFT	2000	798464
IQ correction	vector-scalar multiplication	2000	798464
sideband suppression	vector addition	2000	399232
	copy (for upper and lower sideband)	2000	399232
filter	FFT	2000	798464
	bandpass filter	2000	798464
	inverse FFT	2000	798464
RF phase calibration	complex vector scalar multiplication	2000	798464
cross-correlations	vector mean	2000	798464
	subtract vector mean	2000	798464
	vector dot product	2000	2794624

TABLE II. Acceleration Results

			CUDA			
			Tesla K40C		**GeForce GTX770**	
	IDL	**C**	/dev/shm	hard drive	/mnt/ramdisk	hard disk
Total time (s)	1038.38	464.55	17.42	76.02	25.44	70.34
Acceleration		**2**	**59**	**13**	**40**	**14**

TABLE III. Read times on SAMI system

	/mnt/ramdisk	**hard disk**
CPU read (s)	16.38	60.81
GPU incl. memcopies (s)	7.32	7.40
Total runtime (s)	25.44	70.34

TABLE IV. Runtimes for different numbers of streams on Tesla K40C

	Number of streams			
	1	**2**	**3**	**4**
CPU read (s)	12.77	12.86	12.94	12.94
GPU incl. memcopies (s)	11.96	6.32	6.82	6.85
Total runtime (s)	24.89	19.39	20.01	20.06

TABLE V. Split of GPU time on the Tesla K40C for one stream implementation

	Time (s)
Copying to device	4.85
Kernel execution	6.94
Copying from the device	0.17