IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received December 22, 2020, accepted January 11, 2021, date of publication January 21, 2021, date of current version February 1, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3053409

Efficiency Near the Edge: Increasing the Energy Efficiency of FFTs on GPUs for Real-Time Edge Computing

KAREL ADÁMEK[1,2], JAN NOVOTNÝ[1,3], JEYARAJAN THIYAGALINGAM[1,4], (Senior Member, IEEE), AND WESLEY ARMOUR[1]

[1]Oxford e-Research Centre, Department of Engineering Science, University of Oxford, Oxford OX1 3QG, U.K.
[2]Faculty of Information Technology, Czech Technical University, 160 00 Prague, Czech Republic
[3]Research Centre for Theoretical Physics and Astrophysics, Institute of Physics, Silesian University in Opava, 746 01 Opava, Czech Republic
[4]Rutherford Appleton Laboratory, Science and Technology Facilities Council, Harwell Campus, Didcot OX11 0QX, U.K.

Corresponding author: Karel Adámek (karel.adamek@gmail.com)

This work was supported in part by the Science and Technology Facilities Council (STFC), U.K., under Grant ST/T000570/1, and in part by the OP VVV MEYS Funded Project "Research Center for Informatics" under Grant CZ.02.1.01/0.0/0.0/16_019/0000765.

ABSTRACT The Square Kilometer Array (SKA) is an international initiative for developing the world's largest radio telescope with a total collecting area of over a million square meters. The scale of the operation, combined with the remote location of the telescope, requires the use of energy-efficient computational algorithms. This, along with the extreme data rates that will be produced by the SKA and the requirement for a real-time observing capability, necessitates in-situ data processing in an edge style computing solution. More generally, energy efficiency in the modern computing landscape is becoming of paramount concern. Whether it be the power budget that can limit some of the world's largest supercomputers, or the limited power available to the smallest Internet-of-Things devices. In this article, we study the impact of hardware frequency scaling on the energy consumption and execution time of the Fast Fourier Transform (FFT) on NVIDIA GPUs using the cuFFT library. The FFT is used in many areas of science and it is one of the key algorithms used in radio astronomy data processing pipelines. Through the use of frequency scaling, we show that we can lower the power consumption of the NVIDIA A100 GPU when computing the FFT by up to 47% compared to the boost clock frequency, with less than a 10% increase in the execution time. Furthermore, using one common core clock frequency for all tested FFT lengths, we show on average a 43% reduction in power consumption compared to the boost core clock frequency with an increase in the execution time still below 10%. We demonstrate how these results can be used to lower the power consumption of existing data processing pipelines. These savings, when considered over years of operation, can yield significant financial savings, but can also lead to a significant reduction of greenhouse gas emissions.

INDEX TERMS Energy efficiency, high performance computing, real-time systems, parallel architectures, fast Fourier transforms.

I. INTRODUCTION

The Fast Fourier Transform (FFT) is one of the most fundamental and widely used numerical algorithms in scientific computing, with applications in a diverse range of areas such as astronomy, image processing, audio and radar signal processing, numerical solvers, such as partial differential solvers, and mechanical systems [1]. The FFT is also an integral part of many data processing pipelines, and is an important part in both image- [2]–[5] and time-domain [6]–[11] radio astronomy.

The upcoming, next-generation radio telescope, the Square Kilometer Array (SKA), will employ such complex data processing pipelines to deliver science products that will provide new and exciting insights into our Universe.

Previous studies, for example [12], estimate that the SKA will require an exascale size high performance computing (HPC) system to provide us with such scientific products. Where, the computational footprint of the FFT, depending on the data processing task, may occupy [13] up to 47% of the

The associate editor coordinating the review of this manuscript and approving it for publication was Byoung Wook Choi.

overall computational footprint measured in floating-point operations per second (or FLOPS). This makes the FFT a critical algorithm for the SKA.

Processing the data captured by the SKA possess many challenges. The SKA will produce extremely large volumes of data at unprecedented rates. Furthermore, the telescope itself must be located in a radio-quiet area due to its extreme sensitivity. This makes the persistent storage of all data not viable and transportation of these data to a well equipped (and suitably powered) data center impractical. Finally, some science cases such as the study of Fast Radio Bursts (FRBs), necessitate near real-time data processing. Meaning that data has to be processed close to the instrument itself. These constraints present significant challenges to software and system engineers, they demand high fractions of peak performance of the hardware, whilst maintaining the best possible energy efficiency of both software and hardware.

To address the need to minimize the power consumption of the locally installed hardware, close attention must be paid to the energy efficiency of the data processing algorithms, specifically the FFT. Given the emphasis on lower power consumption in HPC in general, the ability to compute the FFT more efficiently is of interest to many computational domains.

The near real-time processing constraint means that the execution time of the data processing algorithms must not be increased significantly. An increase in the execution time might lead to either failure to process data on time and hence a loss of scientifically important data or increased capital and operational costs as more hardware would be needed to meet the real-time requirement.

Motivated by this, we have studied the impact of dynamic frequency scaling (DFS) on the energy efficiency and execution time of the FFT on NVIDIA GPUs using the cuFFT library [14]. The GPU is the fastest and most energy efficient choice of hardware for image domain radio astronomy as shown by [15], with FPGAs a close second. There are other FFT libraries for GPU's, notably, the clFFT library which uses the OpenCL framework. clFFT is not a vendor supported library and was shown by [16] to be slower than cuFFT on NVIDIA GPUs thus we have not considered it for this work.

Our exhaustive study, conducted on a range of state-of-the-art GPUs shows that careful tuning of the core clock frequency can save, in the case of the A100 GPU, up to 47% (compared to the boost core clock frequency) of the energy consumption of the FFT (for V100 GPU it is 60%). This saving can have a significant impact on two fronts: financial savings in recurrent costs, and the associated reduced CO_2 emission. We also show that these carefully tuned frequencies can be replaced with a single frequency that is specific to each model of GPU and chosen floating-point precision, whilst still being able to save on average up to 43% of the FFT energy consumption for the A100 GPU or 50% for V100 GPU.

The main contributions of this work are:

- We have performed an in-depth investigation of cuFFT library's power consumption and execution time and how it changes with core clock frequency for a wide range of problem sizes and numerical precisions (FP16, FP32 and FP64) on five NVIDIA GPUs.
- We identify an optimal core clock frequency with the highest energy efficiency for all problem sizes and numerical precisions and have shown that a single mean optimal frequency per GPU model gives similar power savings regardless of problem size.
- We demonstrate how these results can be used to lower the power consumption of existing data processing pipelines.

Whilst this work has been motivated by the SKA radio telescope, the conclusions of the work are applicable to any computational task that employs cuFFT running on NVIDIA GPUs.

II. BACKGROUND

Power consumption in HPC is being solved on multiple levels. From construction at the level of the cluster to new energy efficient hardware. The power consumption of specific hardware depends on *execution time*, the time taken to finish a calculation, and also on the utilization of the hardware (memory, cache, computing cores). The software itself also plays an important role in power consumption. Energy can be saved through proper software design, making software stable [17] and through the use of appropriate algorithms.

However, concerns regarding energy efficiency in the modern computing landscape are not solely limited to HPC. Edge computing is becoming an increasingly important research area driven by the explosion of Internet-of-Things devices. The basic premise of edge computing is to capture and process data as close to their sources as is possible by utilizing light weight processors. Because edge computing aims to process data locally, it minimizes wider latency and bandwidth needs and allows for real-time feedback. It is estimated that by 2025 around 150 billion devices will be connected and creating data in real-time [18], with the FFT playing, not only an important role in the communication between devices, but also in processing collected data. Hence optimizing the energy efficiency of the FFT on edge devices is of importance from an environmental perspective. This has motivated us to include NVIDIA's Jetson Nano in our selection of hardware since it represents NVIDIA's low power edge computing solution.

The idea behind DFS, which is part of the dynamic voltage and frequency scaling (DVFS) method, is to make hardware more energy efficient under different loads by adjusting hardware performance which is achieved by changing clock frequencies to fit the application running on it. By decreasing the clock frequency of a component we decrease its performance while increasing its utilization and thus decreasing the power consumption of a given component. For example, Trefethen and Thiyagalingam [19] have investigated possible energy

savings when running software on CPUs with a different number of threads, compilers and CPU clock frequencies.

Applications can be broadly separated into two classes of performance, the first is where an application or algorithm is compute-bound. This is where the performance bottleneck of the application is the compute resource. This can be the number of floating-point operations which can be performed per second (FLOPS), but also the number of instructions which can be issued per second. The second broad category is memory bandwidth bound applications, where we have enough compute resources but we cannot supply the data through the memory bus to the computing cores quickly enough. In this case the performance is then limited by the memory bandwidth. This bandwidth limitation can occur at any level in the computers memory hierarchy, for example this might be at the level of access to the GPU main memory (called *device memory*), or at the level of one of the caches.

We have investigated the cuFFT library using the NVIDIA Visual Profiler (NVVP). This shows that for all investigated problem sizes GPU kernels used by the cuFFT library are device memory bandwidth bound.

A. FFT ALGORITHM

The one-dimensional discrete Fourier transformation (DFT) of a signal x is given by

$$X_l = \sum_{n=0}^{N-1} x_n \exp\left[-i2\pi \frac{nl}{N}\right], \tag{1}$$

where X_l is the l-th element of a transformed signal, x_n is the n-th element of an input signal, and N is the transformation length or the *FFT length*.

The cuFFT library [14] uses the Cooley-Tukey algorithm [20] for FFT sizes that can be decomposed as multiples of powers of primes from 2 to 127 and Bluestein's algorithm [21] otherwise. For longer FFT lengths the cuFFT library uses multiple GPU kernels to compute the entire FFT, which can be seen by studying the cuFFT library using the NVVP. In many cases, the Fourier transform is calculated more quickly if the FFT length is increased by padding to a more optimized length as was shown by [22].

The two-dimensional Fourier transformation is given by the formula

$$X_{l,k} = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} x_{n,m} \exp\left[-i2\pi \left(\frac{nl}{N} + \frac{mk}{M}\right)\right], \tag{2}$$

where $x_{n,m}$, $X_{l,k}$ is now an element of a matrix of size $N \times M$. The sums in this equation can be evaluated independently which allows us to decompose the two-dimensional Fourier transformation into two sets of one-dimensional Fourier transformations. This is routinely done and it is indeed what cuFFT does when calculating higher-dimensional (2D, 3D) Fourier transformations as shown by the NVVP. Thus by investigating the energy efficiency of the one-dimensional Fourier transformation we are also investigating the energy efficiency of the higher-dimensional Fourier transforms.

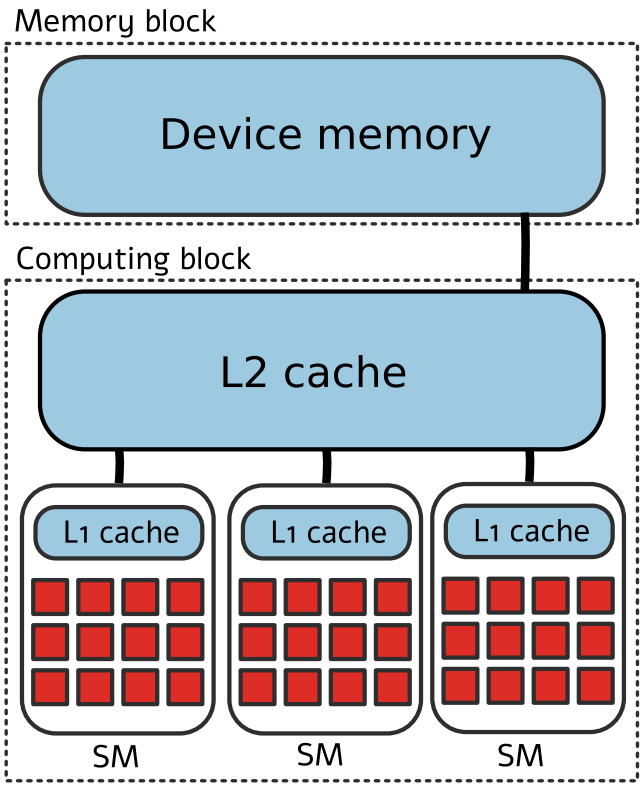

FIGURE 1. A schematic of the GPU architecture.

B. GPU ARCHITECTURE

The GPU design methodology is different to that of a CPU. A CPU architecture is aimed at low latency computations, but also has lower throughput. In other words, the CPU can execute a wider range of complicated algorithms quickly, for example a complicated branching code, but the number of concurrently running tasks is small. A GPU architecture has high latency but also high throughput, on a GPU one can execute thousands of simple tasks concurrently but each task takes longer to complete.

The GPU architecture, which is, in simplified form, shown in Fig. 1, is divided into the memory block and the compute block. The compute block is further divided into caches and streaming multiprocessors (SM) which are responsible for executing the computations. The SMs are further divided into specialized units such as floating-point cores or special function units (which are responsible for complex operations such as computing transcendental functions). The memory hierarchy on the GPU is distributed between these two blocks. The device memory which runs at the memory clock frequency has the lowest bandwidth on the GPU card and it is the memory that the CPU (host) can read/write into via the PCIe bus. The L2 cache is shared between the SMs, the L1 cache is private to each SM and the shared memory is shared amongst a group of threads called a threadblock. The L2, L1 and shared memory bandwidth is proportional to the core clock frequency, thus by using a lower core clock frequency we also decrease the bandwidth of these caches. The core clock frequency, as well as the memory clock frequency, can only be set to predefined values.

Different GPUs may use different memory modules. Amongst the tested GPUs were GPUs with GDDR memory modules (P4, Jetson Nano) which allow us to change the memory clock frequency, but also GPUs with HBM2 modules (Titan V, V100, A100) which do not allow us to change the memory clock frequency.

When measuring the power consumption and performance of the GPU it is important to keep the GPU utilized. For example, the NVIDIA V100 GPU has 80 streaming multiprocessors (SM) where each SM is able to run up to 2048 threads. This gives more than 150 thousands threads which can execute concurrently. Thus in our measurements, we have used a fixed amount of data containing a different number of individual Fourier transforms to keep the GPU utilized for all tested FFT lengths.

C. REAL-TIME PROCESSING

Data processing can be composed of a single step but more often is a series of processing steps which together form a data processing pipeline.

The ability of the application to process data in a real-time processing scenario can be described by the real-time ratio. The real-time ratio is calculated as $S = t_a/t_p$, where t_a is the time needed to acquire a given amount of data by the telescope, sensor, etc. and t_p is the time taken to process that data. The time required to process the data must also include the time spent transferring data to the GPU and back which could represent a significant part of the processing time. When $S \geq 1$ the pipeline is processing data in real-time or quicker and when $S < 1$ the pipeline is not managing to process data in real-time. If we assume that our toy pipeline has a real-time ratio of $S = 1$ that pipeline is processing the data in time but has no performance buffer to call on if needed. In such a case any increase in the execution time leads to $S < 1$ and in order to process data in real-time again we must add more hardware to share the processing load. This situation is however unrealistic and a real-world pipeline would have a performance buffer to call on in the case of an unexpected event. We must also keep in mind that an increase in hardware does not necessarily equate to the same increase in a pipelines performance. The parallelization of a given task might be non-trivial, for example, communication between GPUs could be a limiting factor. In our case this approximation is appropriate as Fourier transformations which can fit into the memory of the GPU can be easily distributed amongst the GPUs.

In this work we consider two situations. The first is where the real-time processing pipeline exists and where the spare performance can be used to increase the energy efficiency of the pipeline. In the second case, we are interested in how much additional hardware is needed to process data in real-time at the best energy efficiency.

III. RELATED WORK

As of November 2019, the first two positions in the top 500 list of supercomputers are held by systems that use GPUs. Within the top ten, five systems contained GPUs. In the Green 500 list, GPUs are used in eight out of the top ten supercomputers. A clear demonstration that it is important to understand the power consumption, energy efficiency and potential energy savings for GPUs using DVFS.

The different approaches of how to measure the power consumption, power and performance modeling and also the results of DVFS for selected applications were reviewed by Mei el al. [23]. The authors note that the effect of DVFS depends not only on the architecture but also on the characteristics of the GPU application. They have found the optimal frequency for 42 GPU applications and found that 12 of them benefited from an increased core frequency compared to the default whereas for 30 applications the optimal frequency was lower than the default core frequency, and values of these optimal frequencies were different for most GPU applications. The authors called for a deeper investigation into their differences. A useful review of the DVFS technique is provided by Mittal and Vetter [24]. The review by Bridges *et al.* [25] looked into the modeling of the power consumption by GPUs.

A number of published studies have investigated the reliability of power measurements using internal sensors. Burtscher *et al.* [26] published their experience of using built-in sensors when measuring the power consumption of NVIDIA K20 GPUs. They described several issues that they encountered when using these sensors and suggested methods to correct for these. The accuracy of the built-in sensors was investigated by Fahad *et al.* [27] who found that the average mean error using an abstract model of a GPU is about 10% compared to measurements using external power meters. This error value was confirmed by Arafa *et al.* [28] who measured the energy consumption of almost all PTX instructions for four generations of NVIDIA GPUs. They have found that the Maxwell and the Turing generations of GPUs have high energy consumption when compared to the Pascal and the Volta generations of NVIDIA GPUs which are found to be more energy efficient.

There are a number of papers where authors have used DVFS in the context of GPUs [23], [29]–[43]. Guerreiro *et al.* [36] classified GPU applications into four different categories which describe their behavior when DVFS is applied. These categories are an extension of the compute-bound, memory-bound distinction. Zamani *et al.* [39] have investigated energy savings by lowering voltage below save limits while detecting and correcting errors which arises from undervolting. The undervolting on GPUs was also explored by Mendes *et al.* [43], where authors have achieved lower energy consumption without performance degradation and in some cases with better performance. Wang and Chu [40] introduce a fine-grained analytical model for estimation of the execution time of different GPU kernels. They have also investigated the memory latency and its dependency on core and memory frequency.

The early work on GPU power consumption and DVFS was performed by Jiao *et al.* [37]. They investigated the

behavior of several GPU applications which included the FFT algorithm, however, the cuFFT library was not studied because there were better performing FFT implementations at the time. The FFT algorithm was part of the benchmark used by Guerreiro *et al.* [36]. The FFT was also indirectly included in Mei *et al.* [23] as part of the convolution, and in Tang *et al.* [38] where the author investigated the effect of DVFS on deep learning applications.

In relation to radio astronomy and the SKA, there are several works. Price *et al.* [44] made a detailed investigation into power consumption, voltage and frequency scaling of the GPU implementation of the correlator for the SKA. The power consumed by the GPU in the domain of radio astronomy was investigated by Romein [45]. The performance of the cuFFT library was investigated by Jodra *et al.* [46] along with its power consumption. However, increases in energy efficiency due to frequency scaling were not investigated.

IV. EXPERIMENTAL SETUP AND EVALUATION

The code that we have used[1] for measurements of the energy efficiency of the FFT algorithm consists of a basic implementation of the NVIDIA cuFFT library [14].

The code first generates input data as pseudo random numbers on the host and then we transfer the data from the host to the device via the PCIe bus. The code runs the FFT algorithm on the GPU multiple times whilst the power used by the GPU is measured as described below. The measurements gained from multiple runs are used to calculate a relative standard deviation which we use to represent the measurement error in the results presented. We provided the GPU with enough data to ensure that it is fully utilized. The Fourier transform used was an out-of-place one-dimensional transform as provided by cuFFT. When the FFT algorithm ends the measurement of the power is stopped. Thus only the power consumption of the FFT algorithm on the GPU is measured. The calculated result is transferred back to the host. The result is then compared to the result from the same transformation again performed by the GPU, but this time using the GPU's default settings. This is done to ensure correctness.

To technically achieve the above scenario we log the timestamp, power consumption, current core clock frequency and current memory clock frequency. For that we use the NVIDIA System Management Interface (`nvidia-smi`) for all GPU cards except the Jetson Nano, where we have used the `tegrastats` utility. For both we have specified the measurement interval to be 10 ms as our tests have showed that a setting of time sampling below 10 ms does not lead to an improvement in the time resolution of our data. The actual time between samples varied and the actual achieved sampling rate from the driver is on average 14.2 ms for all tested FFT lengths and cards. This sampling rate fulfills the criterion of at least 15 ms (66.7 Hz) recommended by Burtscher *et al.* [26] to accurately measure the energy consumption of real-world kernels.

[1] can be found at https://github.com/KAdamek/cuFFT_benchmark

TABLE 1. List of the allowed core clock frequencies from maximal f_{max} up to minimal f_{min} frequency for all cards and their corresponding frequency step size (f_{step}). The size of the frequency step alternates between values shown in the column f_{step} with the exception of the Jetson Nano.

Card name	$f_{\max}$ [MHz]	$f_{\min}$ [MHz]	f_{step} [MHz]
Tesla A100	1410	210	15
Tesla V100	1530	135	7, 8
Titan V	1912	135	7, 8
Tesla P4	1531	455	12, 13
Jetson Nano	921.6	76.8	76.8

For the localization of the FFT algorithm and establishing the execution time we have used the `nvprof` utility where we have included the timestamp. Finally we log the beginning and end of each GPU kernel execution to a file. This way we produce two files containing all of the needed metrics for all possible combinations of core clock frequencies for a specific FFT length, bit precision and GPU card. The final combination (via the timestamp comparison) of these files is done by using a simple R script. Here we compute all other metrics including energy efficiency, optimal clock frequency, mean optimal core clock frequency and computational performance. In the script we also verify that the current core clock frequency is the same as the requested one, and compare the measured execution time from `nvprof` with the log timestamps of the `nvidia-smi` query. Using this method we have found that, for the Titan V, the core clock frequency is capped to 1335 MHz by the driver[2] during the computation, but during the copy of the results is set to a higher core clock frequency (1837 MHz). For frequencies lower than 1335 MHz, no capping is observed. An example of the GPU kernel power consumption and active core clock frequency, which was localized using log file timestamps, is shown for the V100 GPU in Fig. 2 (top). An example of the frequency capping on the Titan V GPU is shown in Fig. 2 (bottom).

The choice of clock frequencies for both the memory bus and the computational cores are limited to a set of supported frequencies defined by the hardware itself. The supported core clock frequency can easily be changed by the driver API. The allowed clock frequencies of the device memory bus are limited or not changeable depending on the memory type. Since the cuFFT library is completely limited by device memory bandwidth this suggests that lowering the memory frequency would not lead to substantial increases in the energy efficiency. Thus, we have not changed the memory clock frequency in this work. Moreover the High Bandwidth Memory (HBM) which is present on the newest GPU cards (Titan V, Tesla V100, Tesla A100) operates on a fixed memory clock frequency. The ranges and step sizes of the core clock frequencies that we have used are summarized in Table 1.

The energy for a specific core clock frequency is defined as

$$E_f = \sum_i P_i \cdot t_i, \tag{3}$$

[2] driver version 450.36.06

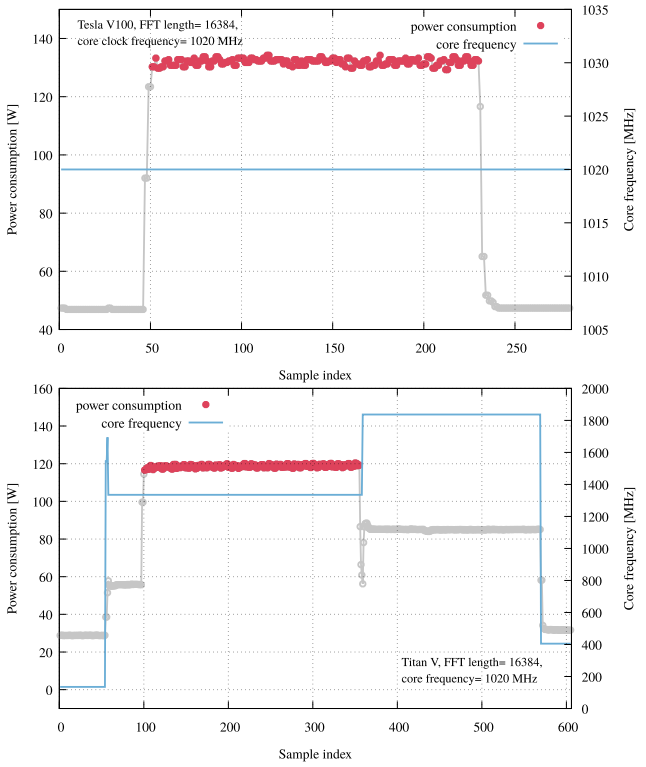

FIGURE 2. Parts of the log file with the GPU kernel highlighted (red dots) by the R script between the two non-computing parts of the GPU run (grey line dots) showing the reported power consumption. The blue line corresponds to the measured core clock frequency. Specifically, the data displayed are from measurements on the Tesla V100 (top) and Titan V (bottom) for an FFT length of 2^{14}, single precision and the core clock frequency set to 1020 MHz (Tesla V100) and 1912 MHz (Titan V).

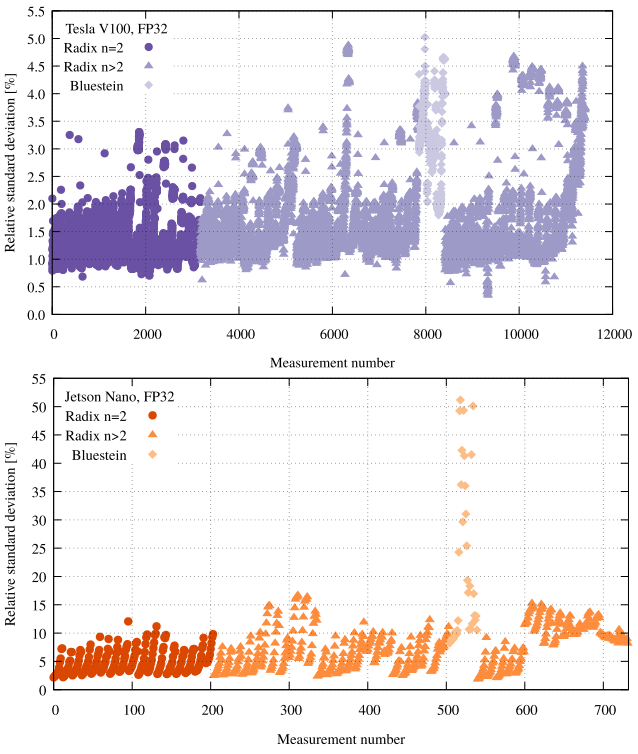

FIGURE 3. Measurement error (V100 GPU at the top, Jetson Nano at the bottom) for all tested FFT lengths at all tested core clock frequencies.

where P_i corresponds to the reported power for a sample index i and t_i is the time between the current sample and the previous one. Then the energy efficiency for a specific core clock frequency is given as

$$E_{\text{ef}} = C_{\text{p}} \cdot t / E_f, \tag{4}$$

where t corresponds to the time of the whole run of the computation, E_f is the energy and C_{p} is the computational performance in FLOPS given by

$$C_{\text{p}} = \left[5N \log_2(N) \cdot N_{\text{b}} \cdot N_{\text{FFT}}\right] / t, \tag{5}$$

where N_{b} is the number of FFT runs of length N and N_{FFT} is the number of FFTs computed per run. The number of Fourier transforms performed (N_{FFT}) depends on the FFT size as follows

$$N_{\text{FFT}} = M_{\text{GB}} / (N \cdot \text{B}), \tag{6}$$

where M_{GB} is the desired amount of memory used for FFTs in GB and B is the byte size of the input data type. The optimal core clock frequency for a specific FFT length is then found as the one with the minimal consumed energy.

We define the increase in energy efficiency as

$$I_{\text{ef}} = E_{\text{ef,o}} / E_{\text{ef,d}}, \tag{7}$$

where $E_{\text{ef,o}}$ and $E_{\text{ef,d}}$ are the energy efficiencies for the optimal frequency and the boost frequency respectively (given by (4)).

The measurement error, that is the relative standard deviation, for the V100 GPU and the Jetson Nano is shown in Fig. 3. We have observed that the measurement error, in general, is around 5% for all cards except the Jetson Nano. The GPU cards use instrumentation amplifiers for the current/voltage/power monitors, hence the potential error in the measurement is expected to be around 3–5% [47]. The results of our power measurement correspond to the expected characteristics of the on-board chips.

For Fourier transformations of higher radices (7+) or for Fourier transformations which use the Bluestein algorithm we observe a measurement error of up to 5%. The measurement error increases with decreasing core clock frequency and increasing number of GPU kernels used for the FFT calculation.

The measurement error for the Jetson Nano is usually below 15% for all FFT lengths, and is below 10% for power-of-two FFT lengths. The highest measurement error that we have observed is for Bluestein FFT lengths. For these lengths, cuFFT uses multiple kernels (for $N = 139^2$ eleven GPU kernels are used) thus the high measurement error is due to the different loads these GPU kernels exert on the GPU and also the differing power consumption between them. The Bluestein FFT lengths represent a marginal case. Due to large measurement errors for Bluestein FFT lengths on the Jetson Nano we have not included these measurements into our

calculations of mean optimal frequency. However, we present these results for the sake of completeness.

For the measurement of the execution time we have used the NVIDIA Visual Profiler. Using this we have found that the measurement error for the execution time was below 0.3%.

Using propagation of uncertainty the error of the energy (3) is dominated by the measurement error of the power consumption. Based on that, the error in the increase in energy efficiency (7) is given by

$$\sigma_{\mathrm{R}}(I_{\mathrm{ef}}) = \sqrt{2}\sigma_R(E_{\mathrm{ef}}), \tag{8}$$

where σ_{R} is the relative error and we have assumed that the relative error in $E_{\mathrm{ef,o}}$ and $E_{\mathrm{ef,d}}$ are equal. This gives an error for the increase in the energy efficiency of 7% for all GPUs except the Jetson Nano where the error is 21%. These values represent the worst case scenario since most of measurement errors are well below these values.

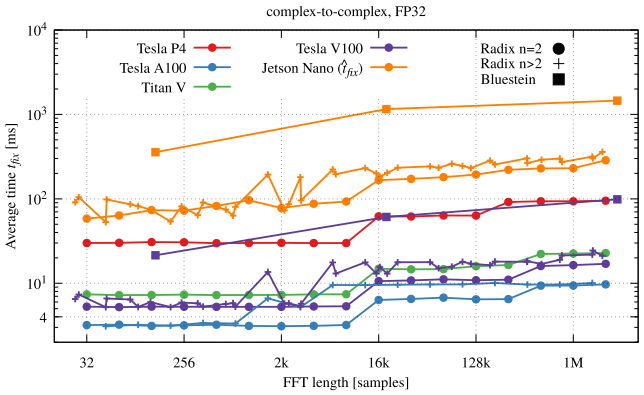

FIGURE 4. The execution time t_{fix} (for FP32) required to process a fixed amount of data for different FFT lengths. The discontinuities in the execution time indicate a change of optimized GPU kernel that is used to calculate the FFT. Results for the Jetson Nano are for one quarter of the memory size.

V. RESULTS

For our investigation, we have used five different NVIDIA GPUs from four recent architecture generations, namely Tesla A100 (Ampere), Tesla V100 (Volta), Tesla P4 (Pascal), Jetson Nano (Maxwell) and Titan V (Volta). The relevant hardware specifications can be found in Table 2. The Tesla A100, Tesla V100, and Tesla P4 are aimed at scientific applications, the P4 GPU also offers improved energy efficiency for its generation. The Jetson Nano is a low-powered all-in-one solution for autonomous systems. The Titan V and Jetson Nano are consumer grade GPUs.

GPUs have two different frequency settings: a base and a boost core clock frequency. If not stated otherwise, we have used the boost core clock frequencies. This is because the GPU's default behavior is to perform calculations at the boost core clock frequency. This is indeed what is observed when the GPU is set to default mode and we run our cuFFT code.

We have measured the complex-to-complex (C2C) one-dimensional transform for three different floating-point precisions; double (FP64), float (FP32) and half (FP16). The Tesla P4 and the Jetson Nano GPUs have limited support for the double precision format. Furthermore, the Tesla P4 do not support the half (FP16) floating-point precision. In addition, when using half precision (FP16), the cuFFT library supports only power-of-two FFT lengths.

We have investigated various FFT lengths but focused on lengths that are powers-of-two because FFT algorithms are not only best suited to processing such lengths, but also offer superior execution time performance with powers-of-two lengths. When calculating non-power-of-two FFT lengths it is often faster [22] to pad the data which needs to be Fourier transformed to the nearest higher power-of-two FFT length and then Fourier transform.

First, we present execution times for processing a fixed amount of data t_{fix} which offers an insight into the level of optimization provided by the cuFFT library. The memory requirements to store the data needed for the Fourier transform grows linearly with the FFT length N and the number of floating point operations performed grows as $\mathcal{O}(N \log N)$. Since the cuFFT library is limited by the device memory bandwidth, the execution time is dominated by the time required to transfer the data to computing cores and to store the result back to the device memory. In the ideal case the algorithm would read and store the data only once. However, due to the limited size of the cache, at some point the implementation needs to store or read additional data from the device memory. These additional accesses will increase the execution time. If the algorithm processes a fixed amount of input data the contribution of the initial read and store of the results remains constant. Therefore an increase in the execution time will indicate a change of the GPU kernel.

If we fix the amount of data that is processed then the number of FFTs performed N_{FFT} depends on the FFT length as given by (6). The execution time of a single FFT within a batch is given as $t_t = t_{\mathrm{fix}}/N_{\mathrm{FFT}}$. The execution time t_{fix} for processing a fixed amount of data for various FFT lengths is shown in Fig. 4 for FP32 and in Fig. 5 for FP16 and FP64 precision. The execution time for the Jetson Nano is for 1/4 of the amount of data so the comparable value of t_{fix} is $t_{\mathrm{fix}} = 4\hat{t}_{\mathrm{fix}}$. This is due to the low amount of available memory on the card.

The execution time t_{fix} increases in proportion to the length of the Fourier transform. However, we see regions of the same execution time with sudden increases after specific FFT lengths. These abrupt changes represent a transition from one optimized GPU kernel to another as is shown by the NVIDIA profiler. We must take these changes into account in our analysis since these GPU kernels might behave differently. When the execution time t_{fix} does not increase for a given range of problem sizes (for example from FFT length $N = 32$ to $N = 8192$) it means that the higher number of floating-point operations which come with a larger problem size utilizes GPU resources other than the device memory bandwidth. Given that the Tesla P4 and Jetson GPUs do not fully support all tested floating-point precisions the execution time of Fourier transformations on these GPUs exhibit different behaviors.

TABLE 2. GPU card specifications. The shared memory bandwidth is calculated as BW(bytes/s) = (bank bandwidth (bytes)) × (clock frequency (Hz)) × (32 banks) × (# multiprocessors).

	Tesla A100	Tesla P4	Titan V	Tesla V100	Jetson Nano
CUDA Cores	6912	2560	5120	5120	128
SMs	108	20	80	80	2
Base/Boost Core Clock	1095/1410 MHz	810/1063 MHz	1220/1455 MHz	1290/1530 MHz	921 MHz
Memory Clock	1215 MHz	3003 MHz	850 MHz	877 MHz	1600 MHz
Dv. m. bandwidth	1555 GB/s	192 GB/s	652 GB/s	900 GB/s	25.6 GB/s
Memory modules	HBM2	GDDR5	HBM2	HBM2	LPDDR4
Shared m. bandwidth	19035 GiB/s	2657.5 GiB/s	14550 GiB/s	15300 GiB/s	230.25 GiB/s
Memory size	40 GB	8 GB	12 GB	32 GB	4 GB
TDP	400 W	75 W	250 W	300 W	5/10 W
CUDA version	11.1.74	10.0.130	10.0.130	10.0.130	JetPack 4.2 SDK

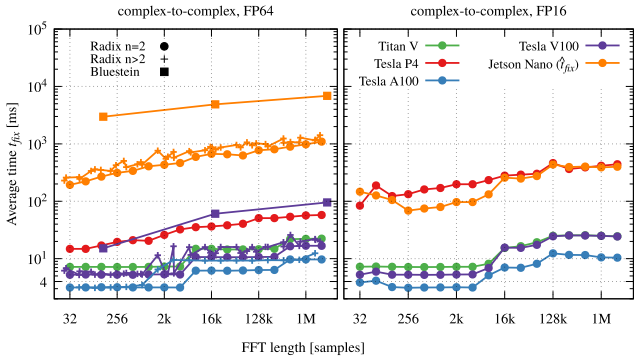

FIGURE 5. The execution time t_{fix} (for FP16 and FP64) required to process a fixed amount of data for different FFT lengths. The discontinuities in execution time indicate a change of optimized GPU kernel that is used to calculate the FFT. Results for the Jetson Nano are for one quarter of the memory size.

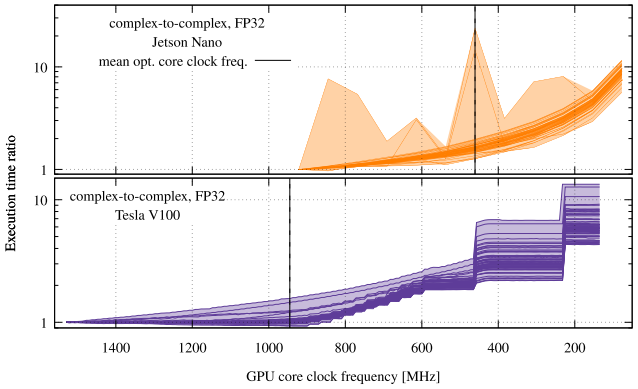

FIGURE 6. Ratio of the execution time t_f over the default execution time t_d measured for the V100 GPU and the Jetson Nano. Every investigated FFT length is shown and represented by a single line.

In this work, results are presented per *FFT batch*, which is the number of FFT's of a given length which fit into the fixed amount of memory that we have chosen to work with. However, most of our results, such as energy efficiency, are independent of the number of FFTs calculated provided that the GPU is fully utilized. The execution time for different core clock frequencies is denoted by t_f. The execution time with boost frequency is denoted as t_d and is taken as the execution time for the default settings. Furthermore, we have focused our discussion on the V100 GPU as it is currently widely used and the most available scientific GPU and the Jetson Nano as it represents NVIDIA's low power edge computing solution. We point out any deviations from these behaviors in the other tested GPUs when they occur.

A. FREQUENCY SCALING

First, we present the behavior of the execution time with changing core clock frequency. This is shown as a ratio of execution time t_f over default execution time t_d in Fig. 6, which shows all tested configurations for FP32 precision.

There are three distinct behaviors, the execution time is:

a) decreasing at first;
b) slightly increasing;
c) increasing notably with each frequency decrease.

In the case of the V100 GPU, the first two behaviors a) and b) are in the majority. For a few specific FFT lengths (notably for $N = 8192$) we have observed behavior c). We have observed this behavior throughout multiple measurements and always for the same FFT lengths. Other tested GPUs behaved similarly to the V100 GPU.

The Jetson Nano exhibits a different behavior, where most of the configurations belong to case c) with notable peaks which are present for Bluestein FFT lengths.

The energy consumed per FFT batch calculated by equation (3) with fixed length $N = 16384$ for different GPUs is shown in Fig. 7. For the measurement, we have used a batch of 16384 FFTs (in the case of FP32 this represents 2 GB of input data) in order to fully saturate the GPU. Notably, the energy per FFT batch on the Titan V GPU does not change above 1335 MHz. This is because the card does not run at the user selected frequency but is capped by the driver to 1335 MHz.

As the core clock frequency decreases the power consumption of the GPU changes non-linearly. This is shown in Fig. 8 for the V100 GPU and the Jetson Nano.

The frequency at which the energy per FFT batch reaches a minimum was selected as the *optimal frequency*. The optimal frequency is different for each tested FFT length for a given GPU and precision. The optimal frequency expressed as a percentage of the default core clock frequency for all precisions is shown in Fig. 9. The optimal frequency for the A100 varies more than other GPUs due to the shallow minimum where the greater range of core clock frequencies has a similar energy efficiency, this can be seen in Fig. 7.

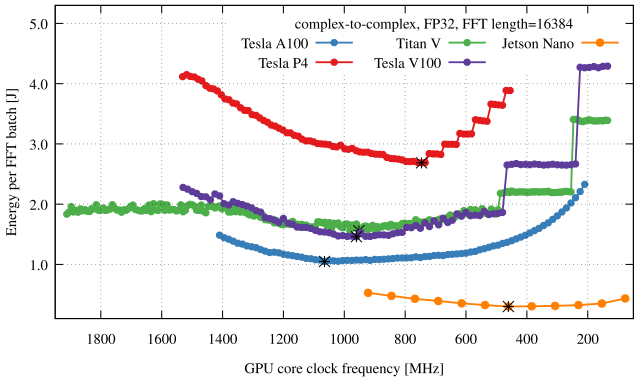

FIGURE 7. The energy consumed per FFT batch changes with core clock frequency. The minimum, emphasized by a black star for each tested GPU, represents the most efficient configuration and the value of the optimal frequency. Note that the Jetson Nano processes one quarter of the data compared to other GPUs.

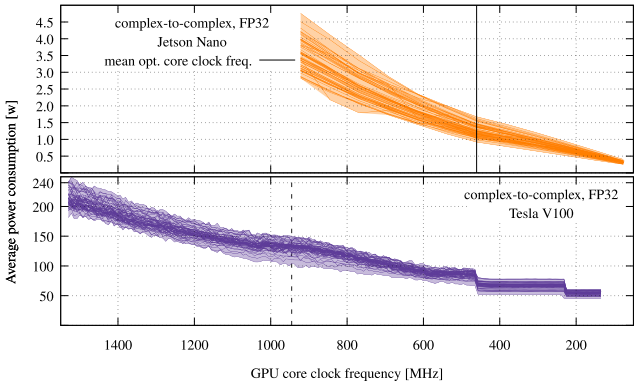

FIGURE 8. Averaged power consumption as a function of core clock frequency for all tested FFT lengths. The Jetson Nano is shown independently as its behavior is different from the rest of the tested GPUs which are represented by the V100 GPU.

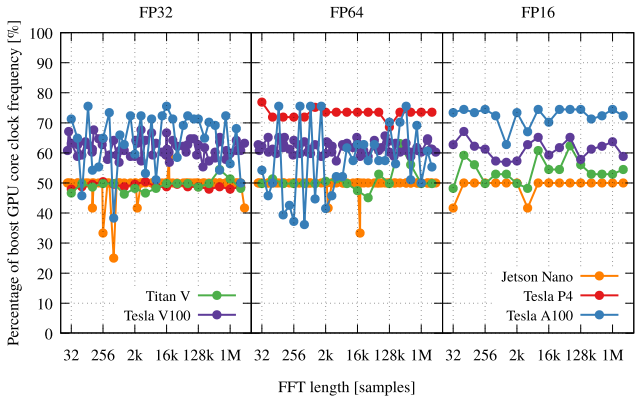

FIGURE 9. Value of the optimal frequency expressed as a percentage of the boost clock frequency. The value of the optimal frequency is consistent through different precisions with the exception of the Tesla P4 GPU.

B. ENERGY SAVINGS

To acquire the following results we have selected the optimal frequency for each FFT length and measured the consumed power to calculate the energy efficiency using

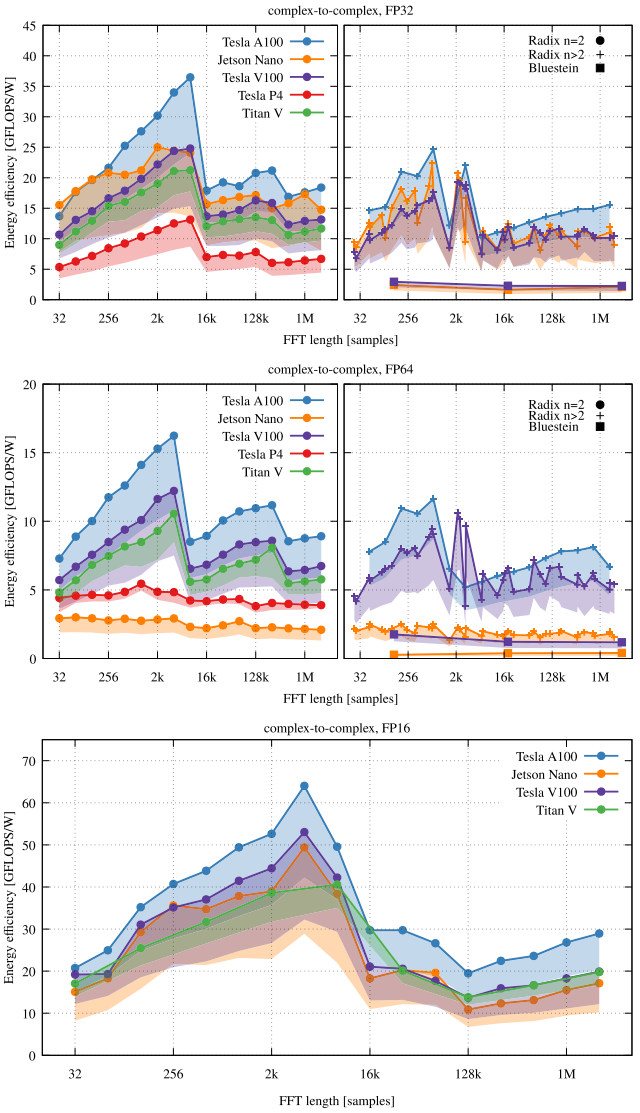

FIGURE 10. Floating-point operations per second per Watt (GFLOPS/W) for optimal frequency. The colored region shows the improvement from the default frequency.

equation (4). The energy efficiency expressed as the number of GFLOPS/W is shown in Fig. 10.

The change in the execution time for the optimal frequency with respect to the default execution time as a percentage is shown in Fig. 11. The change in GFLOPS is shown in Fig. 12. The peaks visible in Fig. 11 correspond to FFT lengths which displayed case c) type behavior of the execution time (Fig. 6).

The increase in the energy efficiency (7) with respect to the boost core clock frequency is shown for different precisions in Fig. 13.

We see that the optimal frequency of different FFT lengths as shown in Fig. 9 is roughly the same for a given GPU and precision across all tested FFT lengths. Furthermore, the optimal frequency is roughly the same across all numerical precisions for a given GPU with the exception of Tesla P4 GPU and the A100 GPU. Based on this we have calculated a *mean optimal frequency* for a given GPU and precision

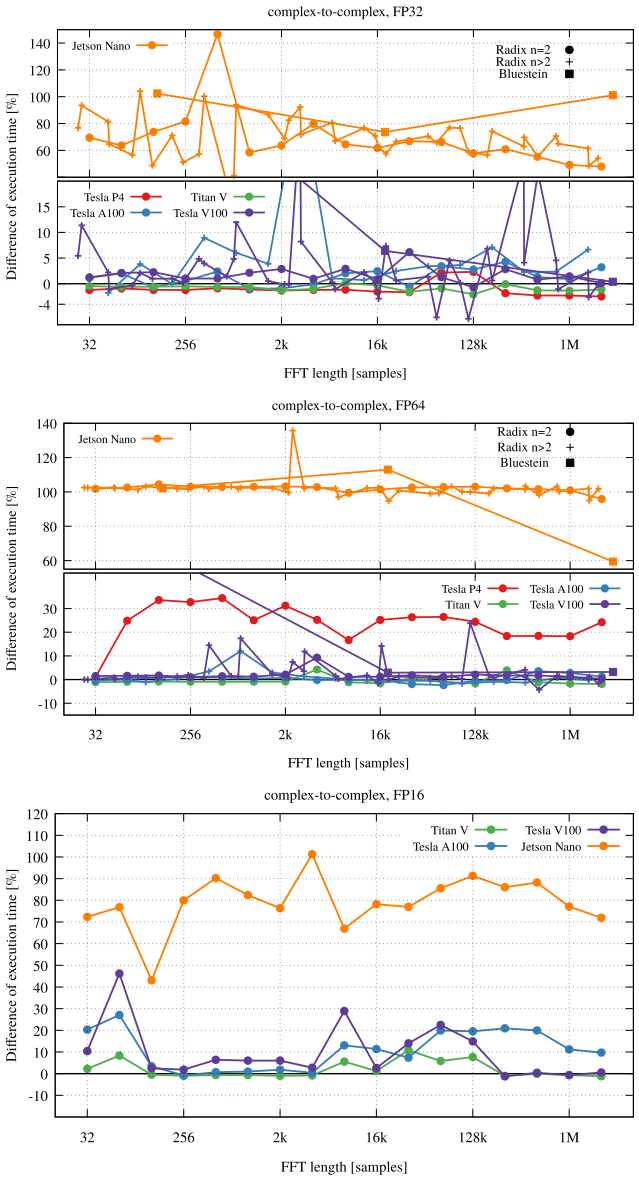

FIGURE 11. Increase in the execution time for optimal frequencies as a percentage of the default execution time t_d.

TABLE 3. Mean optimal core clock frequencies.

Card name	FP32 [MHz]	FP64 [MHz]	FP16 [MHz]
Tesla A100	915	810	1020
Tesla V100	945	945	937
Tesla P4	746	1126	NA
Titan V	952	967	1042
Jetson Nano	460.8	460.8	460.8

by averaging optimal frequencies which achieves a similar increases in energy efficiency for all measured FFT lengths. The increase in energy efficiency using the mean optimal frequency is shown in Fig. 14. The values of mean optimal frequencies are listed in Table 3.

When considering existing pipelines, it is also interesting to study the relationship between the increase in energy efficiency and the increase in the execution time. This relationship indicates the cost (in units of execution time) of any

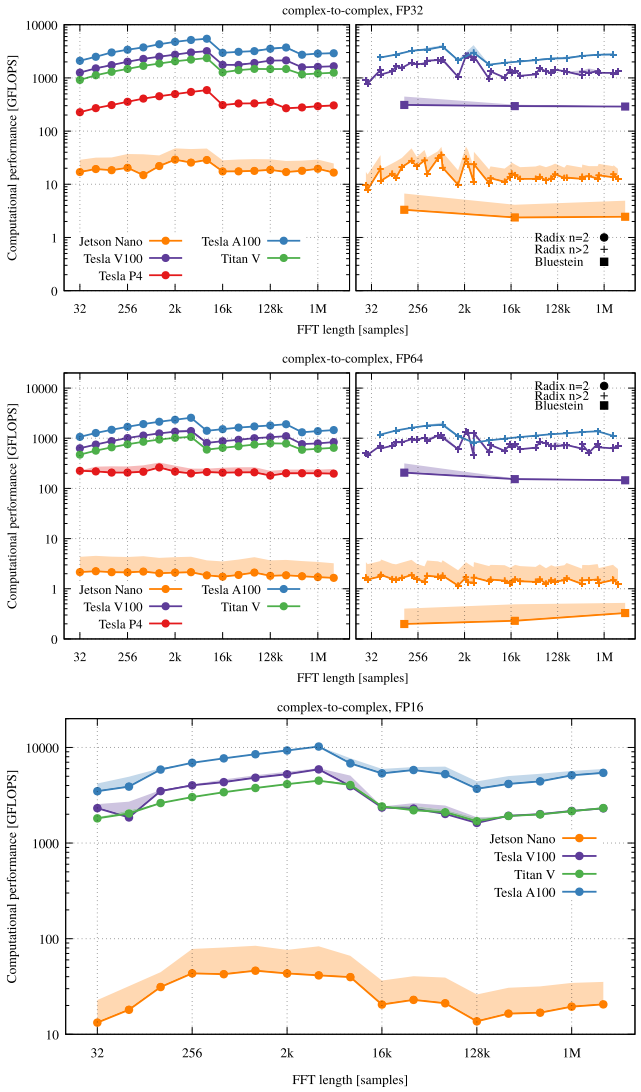

FIGURE 12. Floating-point operations per second (GFLOPS) for optimal frequencies. The colored region shows the change from the default frequency.

increase in energy efficiency. This is shown for the A100 GPU in Fig. 15, for the V100 GPU in Fig. 16 and for the Jetson Nano in Fig. 17. In these figures, we have selected seven equally spaced frequencies (closest possible match) starting with the boost core clock frequency and ending with the mean optimal frequency and presented the achieved increase in energy efficiency and the increase in the execution time. The values of both the increase in energy efficiency and the execution time for a given core clock frequency differ for each FFT length.

As discussed in section II-A the cuFFT library uses decomposition of higher dimensional Fourier transforms into a set of one-dimensional transforms. We have also measured the increase in energy efficiency of the higher-dimensional FFTs using the same procedure as for one-dimensional FFTs. The measured increase in energy efficiency of power-of-two FFTs on the V100 GPU is shown in Fig. 18. The results for other GPUs are similar.

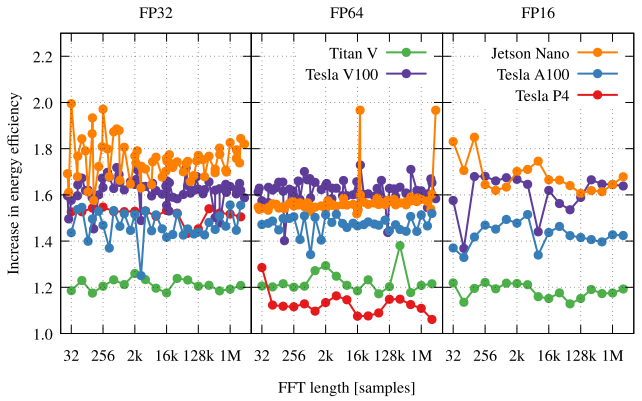

FIGURE 13. The increase in the energy efficiency for optimal frequencies with respect to the boost core clock frequency for all tested FFT lengths. The two peaks observed in the Jetson Nano data are due to the use of the Bluestein algorithm.

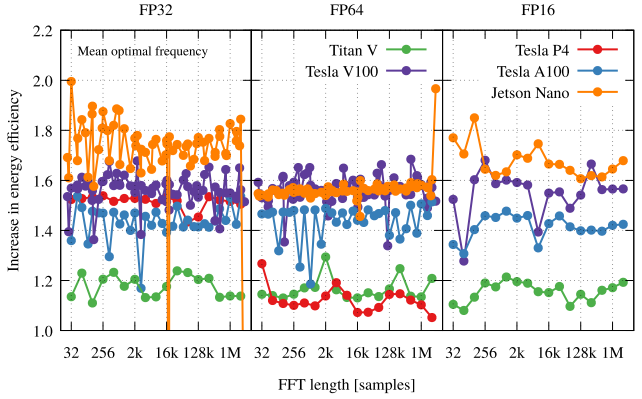

FIGURE 14. The increase in the energy efficiency for the mean optimal frequency with respect to the boost core clock frequency for all tested FFT lengths. The two peaks observed in the Jetson Nano data are due to the use of the Bluestein algorithm.

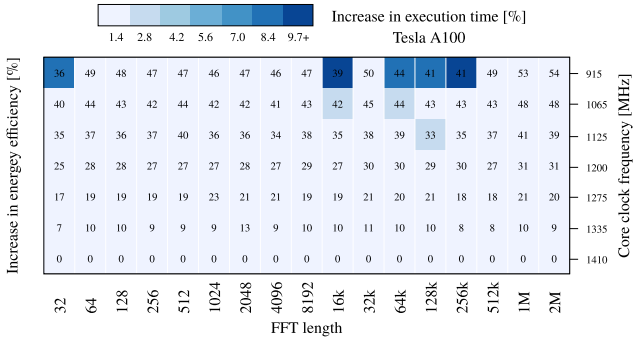

FIGURE 15. Trade-off between an increase in energy efficiency in percent (represented by a number in each cell) and an increase in execution time (represented by a color) for the A100 GPU.

C. INTEGRATION INTO EXISTING PIPELINES

To demonstrate the applicability of the mean optimal frequency in existing pipelines we have employed part of the data processing pipeline[3] used for the detection of pulsars in

[3]Source code for this pipeline is on GitHub https://github.com/KAdamek/cuFFT_energy_efficiency_example

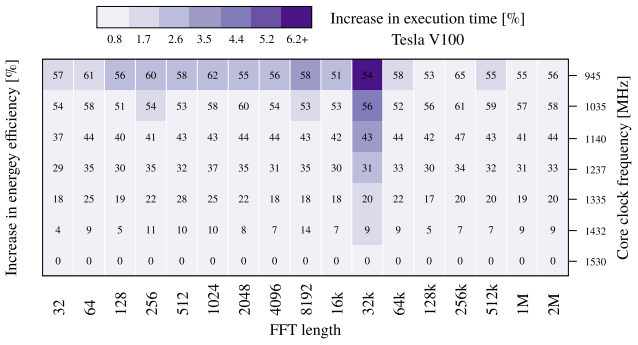

FIGURE 16. Trade-off between an increase in energy efficiency in percent (represented by a number in each cell) and an increase in execution time (represented by a color) for the V100 GPU.

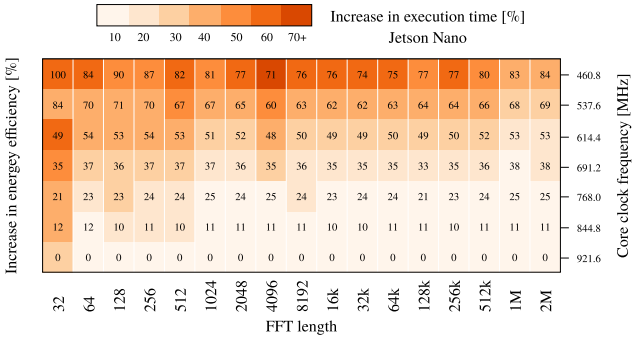

FIGURE 17. Trade-off between an increase in energy efficiency in percent (represented by a number in each cell) and an increase in execution time (represented by a color) for the Jetson Nano.

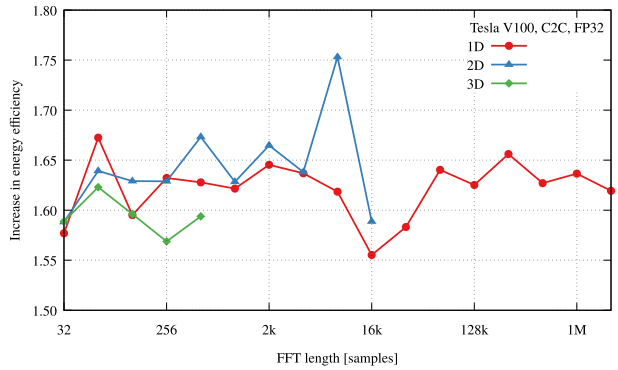

FIGURE 18. Increase in energy efficiency of power-of-two FFT on the V100 GPU.

time-domain radio astronomy data. The pipeline uses several computational steps: FFT, power spectrum calculation; mean and standard deviation calculation; and the harmonic sum. All pipeline stages are performed on the GPU and the core clock frequency was changed only during the cuFFT library execution.

The harmonic sum adds the value of higher harmonics of the pulsar in the power spectrum to the pulsar's expected fundamental frequency thus increasing the signal-to-noise ratio of the pulsar in the power spectrum. The harmonic sum can add up to 32 higher order harmonics which decreases the FFT execution footprint in the pipeline's total execution time.

TABLE 4. Increase in energy efficiency for different configurations of our toy data processing pipeline for V100 GPU.

num. harmonic summed	cuFFT % of total exec. time	Increase in Energy efficiency
2	60.85	1.291
4	58.56	1.290
8	55.92	1.267
16	53.73	1.260
32	51.34	1.240

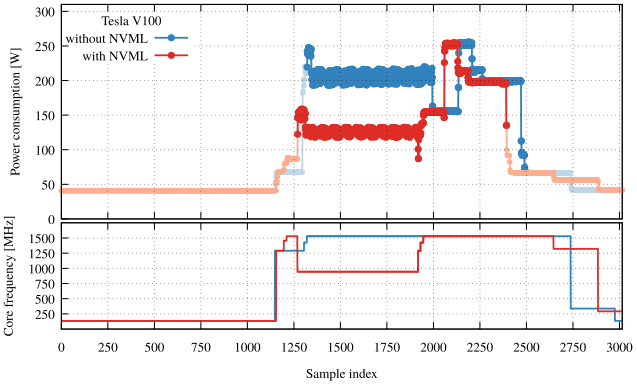

FIGURE 19. Measured power consumption (top) and core clock frequency (bottom) for part of a radio astronomy data processing pipeline.

To change the frequency for just the cuFFT library during the pipeline execution we have used the NVIDIA Management Library (NVML) [48]. This approach, however, has limitations because the library is fully supported only on scientific (Tesla) NVIDIA GPUs. The measured power consumption and the core clock frequency for the V100 GPU are shown in Fig. 19 and the increase in energy efficiency for different configurations of the pipeline is listed in Table 4.

The usage of the NVML library is simple. Before the GPU kernel execution the core clock frequency is (for a given GPU) set using `nvmlDeviceSetGpuLockedClocks` providing maximum and minimum core clock frequency. When the calculation is finished the GPU core clock frequency is returned to default by calling `nvmlDeviceResetGpuLockedClocks`.

The FFT length used for the computation was $N = 5 \cdot 10^5$ which was not used in our measurements or in our calculation of the mean optimal frequency.

D. PROFILING

For profiling, we have used the NVIDIA visual profiler (NVVP) and the V100 GPU. Based on the different behavior of the execution time t_{fix} shown in Fig. 4 we have selected three representative power-of-two FFT lengths ($N = 8192$, $N = 16\text{k}$, $N = 2\text{M}$) which are calculated by different kernels. The profiling results for these kernels are shown in Fig. 20. For our study of compute utilization we have used two indicators. The first is the compute utilization as reported by the NVVP, the second metric is the issue slot utilization, which tells us how many instruction slots are used. The next quantity displayed in Fig. 20 is the device memory bandwidth utilization (device MBU). Fig. 20 also shows the normalized execution time from fastest to slowest to provide context for the other displayed quantities.

VI. DISCUSSION

The dependency of the execution time on the core clock frequency is shown in Fig. 6. This displays the three previously discussed behaviors a), b) and c). However, the Jetson Nano only exhibits the third type of behavior c). All other GPU's, represented by the V100 GPU, exhibit a composition of all three behaviors with cases a) and b) being dominant.

The behavior in case a), might be due to reduced cache contention which slightly increases the hit rate of the unified cache as shown by NVVP. However, it might also be a systematic error caused by measurement using the NVIDIA driver, which is based on the GPU core clock frequency. Similar behavior was observed by Mendes *et al.* [43] on the AMD Vega 10 Frontier Edition GPU, where for default core clock frequencies and voltage the power cap of the GPU is surpassed, which activates the GPU protection mechanisms, this halts the execution until the power is reduced. For lower core clock frequencies or lower voltages, the GPU protection mechanism is not activated, resulting in higher performance. This behavior might be architecture dependent and may not apply to NVIDIA GPUs; however, our power measurements' time resolution did not allow us to confirm this.

In the case b), the memory latencies are not hidden by computations thus the execution time is dominated by the time required to load data from memory, i.e. it is governed by the memory clock frequency which is not changing thus the execution time remains mostly unchanged. The slow increase in the execution time is due to lower rate of issuing memory requests. When the core clock frequency is low enough for the application to be compute bound the execution time becomes dominated by computations and there is a direct dependency of the execution time on the core clock frequency. How the device memory latency is hidden and how it changes with the ratio of core and memory clock frequency is described in [40].

This is also supported by the analysis of the performance counters from NVVP which shows that an increase in the execution time at a particular critical frequency is due to the saturation of the number of issued instructions (see Fig. 20). This leads to a reduction in memory requests to the device memory which, in turn, leads to poor latency hiding of the device memory accesses. Therefore most of the threads are waiting for data but there are not enough threads with data to utilize the floating-point operation units. Thus the floating-point operation utilization remains mostly unchanged.

The sharp increase in the execution time t_{fix} for low frequencies, which are present in all cases, are due to the change of the P-state to a state corresponding to the idle status of the GPU with reduced voltage which reduces the available GPU resources severely.

Lastly, case c) occurs due to the high utilization of one of the caches. Since the cache bandwidth decreases with the core clock frequency each decrease in frequency lowers

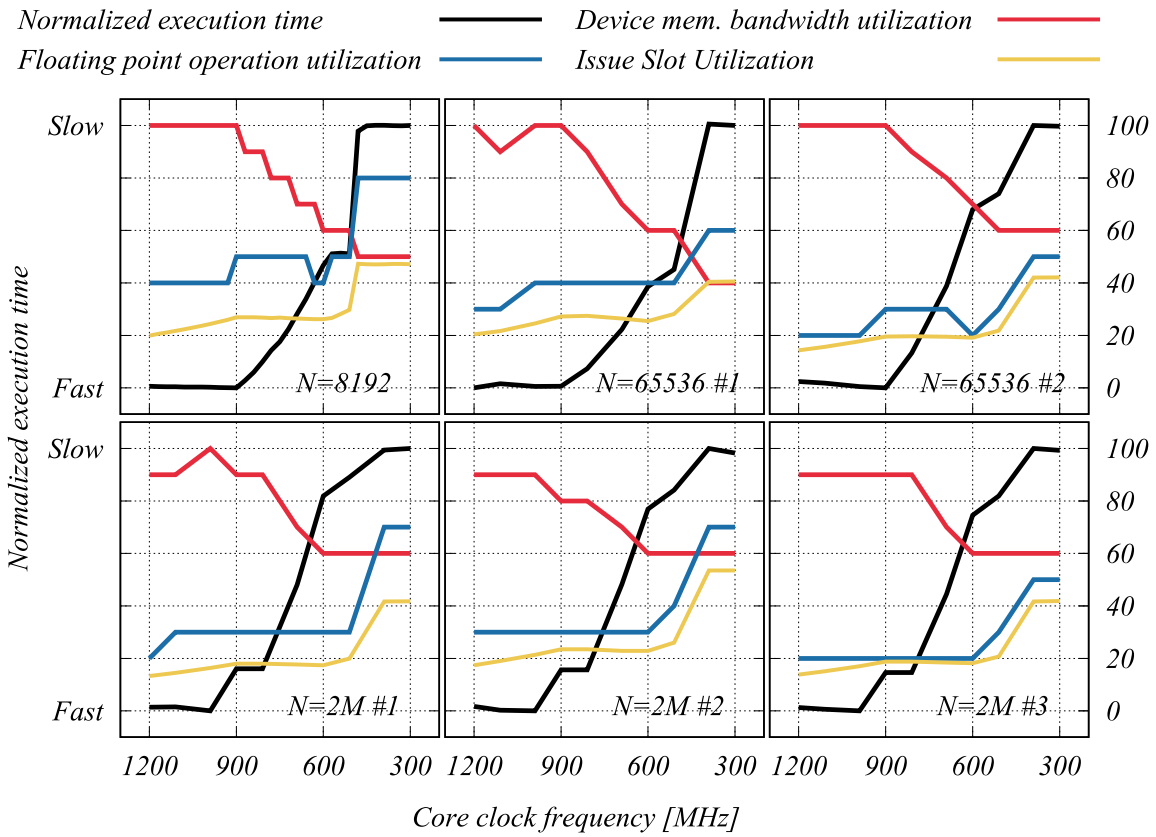

FIGURE 20. Profiling results for the V100 GPU using the NVIDIA visual profiler. Longer FFT lengths use more than one GPU kernel to calculate the Fourier transform which are numbered.

the bandwidth which is already fully utilized leading to a decrease in performance.

The average power consumption shown in Fig. 8, tells us why, even with longer execution times, we can improve energy efficiency. The rate of the decrease in power consumption is higher than the rate at which the execution time increases. This is especially visible around $f = 1000$ Hz for the V100 GPU and about $f = 450$ Hz for the Jetson Nano. These frequencies roughly coincide with the mean optimal frequency for the given GPUs.

A. REAL-TIME PROCESSING

The energy efficiency is shown in Fig. 10, the change in the execution time is shown in Fig. 11 and the change in GFLOPS is shown in Fig. 12.

In the language of costs, Fig. 11 is equivalent to the increase in capital costs as an increase in execution time directly translates into more hardware needed in order to meet the constrains of real-time data processing. On the other hand, the increase in energy efficiency (Fig. 10) is related to operational costs, where better energy efficiency translates into lower operational costs. However we must bear in mind that operational costs include cooling, facility management, etc. which could be increased by the requirement for more hardware due to longer execution times.

For FP32 precision we see that the new A100 GPU (Ampere architecture) is clearly more energy efficient than all other GPUs from older generations. It is almost 30%-50% more efficient than the V100 GPU for power-of-two FFT lengths below 16k. The Jetson Nano is also more energy efficient than the V100 GPU for almost all FFT lengths. Its energy efficiency is, for very short ($N \leq 256$) FFT lengths, equal to the energy efficiency of the A100 GPU.

When we look at the change in the execution time we see that the Jetson Nano requires approximately 60% more time to finish compared to the execution time at the boost core clock frequency. With one extreme case where the execution time is 140% longer. This means on average 60% more hardware to achieve real-time data processing with the best energy efficiency.

This behavior is not reproduced by the V100 GPU or the A100 GPU where the increase in energy efficiency is not, for the most part, at the expense of the execution time. The change in the execution time for both GPUs is below 5%. There are more significant increases in execution time for the non-power of two FFT lengths which can cause increases of up to 20% in execution time. Small changes in the execution time on both the V100 GPU and the A100 GPU offers a possibility to improve existing real-time processing pipelines without substantial change in hardware.

We see similar behavior for the V100 GPU and the A100 GPU at FP64 precision. The slow-down in execution time suffered by both GPUs due to the lower core clock frequencies is within 5%. The execution time for most of the non-power

of two FFT lengths does not increase above 20%. The Tesla P4 GPU and the Jetson Nano do not fully support FP64 precision. This manifests in less significant improvements in GFLOPS/W, much higher execution times and a decrease in GFLOPS. In the case of the Jetson Nano we would have to double the number of cards in order to process data in real-time.

At FP16 precision we have four GPUs which support this precision: A100 GPU, V100 GPU, Titan V GPU and the Jetson Nano. Regarding energy efficiency, the A100 GPU is the most energy efficient GPU. The V100 GPU and the Jetson Nano are comparable but the V100 GPU is the overall more energy efficient GPU of the two. When we look at the change in execution time we see that both the A100 GPU and the V100 GPU typically have a 10% increase or less, but at some FFT lengths the increase could be much higher, e.g. for V100 GPU 50% increase for N = 64 or for the A100 GPU 20% increase for higher FFT lengths (N = 64k+). This behavior means that we have to be more careful about potential energy savings since at some FFT lengths the increase in execution time might be too high for real-time data processing. The change in execution time of the Jetson Nano is again large and we would need to have almost twice the number of GPUs to process data in real time at the best possible energy efficiency.

B. INCREASE IN ENERGY EFFICIENCY

The increase in the energy efficiency for the optimal frequency is shown in Fig. 13. The corresponding figure for the mean optimal frequency is Fig. 14. The difference in the increase in energy efficiency for the boost core clock frequency between the optimal frequency and the mean optimal frequency is, for the A100 GPU, 4 percentage points. That is, an average increase in energy efficiency for the optimal frequency which is tuned for each FFT length is 47% whereas the average increase in energy efficiency for the mean optimal frequency is 43%. For the V100 GPU this difference is 5 percentage points (61% and 56%). For both the V100 GPU and the A100 GPU this holds for all FFT lengths and precisions with a very limited number of exceptions for FP16 precision. This allows us to use one core clock frequency and achieve similar energy savings without determining the optimal frequency for each FFT length. A similar result is observed for the Jetson Nano with the exception of Bluestein FFT lengths which are responsible for the peaks in the results.

The dependency between the increase in energy efficiency and the change in the execution time for FP32, shown in Fig. 16 for the V100 GPU but more notably in Fig. 17 for the Jetson Nano, is non-linear. We see that we can achieve an interesting increase in energy efficiency even for increases in execution time which are below 10%.

Lastly, our practical test with our example data processing pipeline shows that integration of the DVFS techniques into existing pipelines is possible with minimal changes to the codebase. The NVML library allows us to target only the duration of the cuFFT library call within the pipeline and thus reduce power consumption. The increase in energy efficiency (for the boost core clock frequency) are summarized in Table 4 corresponds to the expected values based on the FFT execution time footprint within the pipeline. For the first configuration with 2 harmonics, the FFT execution time corresponds to 60% of the total execution time. The average increase in energy efficiency for V100 GPU with boost core clock frequency (based on Fig.14) is about 50%. Considering the FFT execution time footprint we should get 30% increase in energy efficiency which is indeed what we have measured. This behavior is consistent with other configurations of the pipeline.

VII. CONCLUSION

We have measured the power consumption when calculating the Fourier transformation at different numerical precisions (FP32, FP64, FP16) on NVIDIA GPUs using the NVIDIA cuFFT library and quantified the possible energy savings when DVFS techniques are used. For each tested GPU, precision, and a wide range of FFT lengths, we have found the optimal core clock frequency to minimize power consumption. We have also measured the change in execution time of the Fourier transform when DVFS is applied, which is an important consideration for real-time data processing because this can increase when the core clock frequencies of the GPU are modified.

We have presented the achieved energy efficiency in GFLOPS/W. Along with this we have presented the increase in energy efficiency when using our optimal core clock frequency compared to the boost and base core clock frequency for each GPU. We have also presented the increase in the execution time of the Fourier transform when DVFS is applied.

The decrease in power consumption and change in the execution time depends on the GPU used. The average increase in the energy efficiency for FP32, FP64, and FP16 precisions compared to the boost core clock frequency is 40% for the A100 GPU and 60% for the V100 GPU. The increase in the execution time for both GPUs is below 5% (with few exceptions as outlined). The Jetson Nano offers higher increases in energy efficiency to that of the V100 GPU. On average 70% for FP32, 55% for FP64 and 70% for FP16 but at the expense of execution time which increases by more than 60%. For the P4 GPU and the Titan V GPU we have not achieved a significant increase in energy efficiency.

Our results have shown that the Ampere architecture (A100 GPU) is significantly more energy efficient for all numerical precisions than the V100 GPU which represents the comparable scientific GPU from the previous Volta generation. When the V100 GPU is compared to the Jetson Nano the V100 GPU is less energy efficient at FP32 precision. For short and long FFTs at FP32 precision the Jetson Nano is almost 50% more energy efficient than the V100 GPU. For FP16 precision the V100 GPU has similar energy efficiency as the Jetson Nano. The Jetson Nano does not fully support double precision thus the V100 GPU is significantly more energy efficient at this precision.

We have shown that values of optimal core clock frequencies for all tested FFT lengths for a given GPU and numerical precision are similar, with few exceptions. This allowed us to define a mean optimal core clock frequency unique to each tested GPU and precision, but is the same for all FFT lengths. Using the mean optimal core clock frequency, we have achieved a similar energy efficiency when compared to the energy efficiency achieved with the optimal core clock frequency for each tested FFT length. For the A100 GPU the difference is only 4 percentage points and for the V100 GPU the difference is 5 percentage points. For the other GPUs the loss is similar.

We have also presented the practical implementation of these results in our example data processing pipeline which is available as an open source code. We have demonstrated how to change the core clock frequency of the GPU to the mean optimal core clock frequency using the NVIDIA Management Library and demonstrated a decrease in power consumption which is in agreement with the results presented in this work.

Finally we have highlighted how, from an environmental perspective, increasing the energy efficiency of the FFT algorithm will be an important consideration for edge computing and IoT.

ACKNOWLEDGMENT

The authors would like to acknowledge the use of the University of Oxford Advanced Research Computing (ARC) facility in carrying out this work (`http://dx.doi.org/10.5281/zenodo.22558`). They would like to express their gratitude to the Research Centre for Theoretical Physics and Astrophysics, Institute of Physics, Silesian University in Opava for institutional support.

REFERENCES

[1] E. O. Brigham, *The Fast Fourier Transform and Its Applications* (Signal Processing Series). Upper Saddle River, NJ, USA: Prentice-Hall, 1988.

[2] S. van der Tol, B. Veenboer, and A. R. Offringa, "Image domain gridding: A fast method for convolutional resampling of visibilities," *Astron. Astrophys.*, vol. 616, p. A27, Aug. 2018.

[3] A. R. Offringa, B. McKinley, N. Hurley-Walker, F. H. Briggs, R. B. Wayth, D. L. Kaplan, M. E. Bell, L. Feng, A. R. Neben, J. D. Hughes, and J. Rhee, "WSCLEAN: An implementation of a fast, generic wide-field imager for radio astronomy," *MNRAS*, vol. 444, no. 1, pp. 606–619, Oct. 2014.

[4] B. Veenboer, M. Petschow, and J. W. Romein, "Image-domain gridding on graphics processors," in *Proc. IEEE Int. Parallel Distrib. Process. Symp. (IPDPS)*, May 2017, pp. 545–554.

[5] J. S. Farnes, B. Mort, F. Dulwich, K. Adamek, A. Brown, J. Novotny, S. Salvini, and W. Armour, "Building the world's largest radio telescope: The square kilometre array science data processor," in *Proc. IEEE 14th Int. Conf. E-Sci. (E-Sci.)*, Oct. 2018, pp. 366–367.

[6] S. Dimoudi, K. Adamek, P. Thiagaraj, S. M. Ransom, A. Karastergiou, and W. Armour, "A GPU implementation of the correlation technique for real-time Fourier domain pulsar acceleration searches," *Astrophysical J. Suppl. Ser.*, vol. 239, no. 2, p. 28, Dec. 2018.

[7] K. Adámek, S. Dimoudi, M. Giles, and W. Armour, "Improved acceleration of the GPU Fourier domain acceleration search algorithm," in *Astronomical Data Analysis Software and Systems XXVI*, vol. 522, P. Ballester, J. Ibsen, M. Solar, and K. Shortridge, Eds. San Francisco, CA, USA: ASP, Apr. 2020, p. 477.

[8] K. Adámek and W. Armour, "A GPU implementation of the harmonic sum algorithm," in *Astronomical Data Analysis Software and Systems XXVII*, vol. 523, P. J. Teuben, M. W. Pound, B. A. Thomas, and E. M. Warner, Eds. San Francisco, CA, USA: ASP, Oct. 2019, p. 489.

[9] L. Levin, W. Armour, C. Baffa, E. Barr, S. Cooper, R. Eatough, A. Ensor, E. Giani, A. Karastergiou, R. Karuppusamy, and, "Pulsar searches with the SKA," in *Proc. Int. Astronomical Union*, 2017, vol. 13, no. S337, pp. 171–174.

[10] K. Rajwade, B. Stappers, C. Williams, E. Barr, M. C. Bezuidenhout, M. Caleb, L. Driessen, F. Jankowski, M. Malenta, V. Morello, S. Sanidas, and M. Surnis, "MeerTRAP in the era of multi-messenger astrophysics," *Proc. SPIE*, vol. 1147, pp. 78–85, Dec. 2020, doi: 10.1117/12.2559937.

[11] S. Sanidas, M. Caleb, L. Driessen, V. Morello, K. Rajwade, and B. W. Stappers, "MeerTRAP: A pulsar and fast transients survey with MeerKAT," in *Pulsar Astrophysics the Next Fifty Years*, vol. 337, P. Weltevrede, B. B. P. Perera, L. L. Preston, and S. Sanidas, Eds. Cambridge, U.K.: Cambridge Univ. Press, Aug. 2018, pp. 406–407.

[12] T. Cornwell and B. Humphreys. (2010). *SKA Exascale Software Challenges*. [Online]. Available: http://www.skatelescope.org/uploaded/27848_128_Memo_Cornwell.pdf

[13] R. Jongerius, S. Wijnholds, R. Nijboer, and H. Corporaal, "An end-to-end computing model for the square kilometre array," *Computer*, vol. 47, no. 9, pp. 48–54, Sep. 2014.

[14] NVIDIA. (2020). *Cufft Library User's Guide, v11.0.3*. [Online]. Available: https://docs.nvidia.com/cuda/cufft

[15] B. Veenboer and J. W. Romein, "Radio-astronomical imaging: FPGAs vs GPUs," in *Euro-Par 2019: Parallel Processing*, R. Yahyapour, Ed. Cham, Switzerland: Springer, 2019, pp. 509–521.

[16] P. Steinbach and M. Werner, "Gearshifft–the FFT benchmark suite for heterogeneous platforms," 2017, *arXiv:1702.00629*. [Online]. Available: http://arxiv.org/abs/1702.00629

[17] E. Meneses, O. Sarood, and L. V. Kale, "Assessing energy efficiency of fault tolerance protocols for HPC systems," in *Proc. IEEE 24th Int. Symp. Comput. Archit. High Perform. Comput.*, Oct. 2012, pp. 35–42.

[18] D. Reinsel, J. Gantz, and J. Rydning, "The digitization of the world from edge to core," Int. Data Corp., Framingham, MA, USA, 2018.

[19] A. E. Trefethen and J. Thiyagalingam, "Energy-aware software: Challenges, opportunities and strategies," *J. Comput. Sci.*, vol. 4, no. 6, pp. 444–449, Nov. 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877750313000173

[20] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series," *Math. Comput.*, vol. 19, no. 90, pp. 297–301, 1965. [Online]. Available: http://www.jstor.org/stable/2003354

[21] L. Bluestein, "A linear filtering approach to the computation of discrete Fourier transform," *IEEE Trans. Audio Electroacoust.*, vol. AU-18, no. 4, pp. 451–455, Dec. 1970.

[22] D. Strelák and J. Filipovic, "Performance analysis and autotuning setup of the cuFFT library," in *Proc. 2nd Workshop Autotuning Adaptivity Approaches Energy efficient HPC Syst.* New York, NY, USA: Association Computing Machinery, 2018, pp. 1–6, doi: 10.1145/3295816.3295817.

[23] X. Mei, Q. Wang, and X. Chu, "A survey and measurement study of GPU DVFS on energy conservation," *Digit. Commun. Netw.*, vol. 3, no. 2, pp. 89–100, May 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2352864816300736

[24] S. Mittal and J. S. Vetter, "A survey of methods for analyzing and improving GPU energy efficiency," *ACM Comput. Surv.*, vol. 47, no. 2, pp. 1–23, Jan. 2015, doi: 10.1145/2636342.

[25] R. A. Bridges, N. Imam, and T. M. Mintz, "Understanding GPU power: A survey of profiling, modeling, and simulation methods," *ACM Comput. Surv.*, vol. 49, no. 3, pp. 1–27, Dec. 2016, doi: 10.1145/2962131.

[26] M. Burtscher, I. Zecena, and Z. Zong, "Measuring GPU power with the K20 built-in sensor," in *Proc. Workshop Gen. Purpose Process. GPUs*. New York, NY, USA: Association Computing Machinery, 2014, pp. 28–36, doi: 10.1145/2588768.2576783.

[27] M. Fahad, A. Shahid, R. R. Manumachu, and A. Lastovetsky, "A comparative study of methods for measurement of energy of computing," *Energies*, vol. 12, no. 11, p. 2204, Jun. 2019. [Online]. Available: https://www.mdpi.com/1996-1073/12/11/2204

[28] Y. Arafa, A. ElWazir, A. ElKanishy, Y. Aly, A. Elsayed, A.-H. Badawy, G. Chennupati, S. Eidenbenz, and N. Santhi, "Verified instruction-level energy consumption measurement for NVIDIA GPUs," in *Proc. 17th ACM Int. Conf. Comput. Frontiers (CF)*. New York, NY, USA: Association for Computing Machinery, 2020, pp. 60–70, doi: 10.1145/3387902.3392613.

[29] Y. Abe, H. Sasaki, S. Kato, K. Inoue, M. Edahiro, and M. Peres, "Power and performance characterization and modeling of GPU-accelerated systems," in *Proc. IEEE 28th Int. Parallel Distrib. Process. Symp.* Washington, DC, USA: IEEE Computer Society, May 2014, pp. 113–122, doi: 10.1109/IPDPS.2014.23.

[30] A. Sethia and S. Mahlke, "Equalizer: Dynamic tuning of GPU resources for efficient execution," in *Proc. 47th Annu. IEEE/ACM Int. Symp. Microarchitecture*, Dec. 2014, pp. 647–658.
[31] Q. Wang, C. Liu, and X. Chu, "GPGPU performance estimation for frequency scaling using cross-benchmarking," in *Proc. 13th Annu. Workshop Gen. Purpose Process. Graph. Process. Unit*. New York, NY, USA: Association Computing Machinery, Feb. 2020, pp. 31–40, doi: 10.1145/3366428.3380767.
[32] D. Loghin and Y. M. Teo, "The energy efficiency of modern multi-core systems," in *Proc. 47th Int. Conf. Parallel Process. Companion*. New York, NY, USA: Association Computing Machinery, Aug. 2018, pp. 1–10, doi: 10.1145/3229710.3229714.
[33] R. Ge, R. Vogt, J. Majumder, A. Alam, M. Burtscher, and Z. Zong, "Effects of dynamic voltage and frequency scaling on a K20 GPU," in *Proc. 42nd Int. Conf. Parallel Process.*, Oct. 2013, pp. 826–833.
[34] V. Chau, X. Chu, H. Liu, and Y.-W. Leung, "Energy efficient job scheduling with DVFS for CPU-GPU heterogeneous systems," in *Proc. 8th Int. Conf. Future Energy Syst.* New York, NY, USA: Association Computing Machinery, May 2017, pp. 1–11, doi: 10.1145/3077839.3077855.
[35] J. Lee, V. Sathisha, M. Schulte, K. Compton, and N. S. Kim, "Improving throughput of power-constrained GPUs using dynamic voltage/frequency and core scaling," in *Proc. Int. Conf. Parallel Archit. Compilation Techn.*, Oct. 2011, pp. 111–120.
[36] J. Guerreiro, A. Ilic, N. Roma, and P. Tomás, "DVFS-aware application classification to improve GPGPUs energy efficiency," *Parallel Comput.*, vol. 83, pp. 93–117, Apr. 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167819118300243
[37] Y. Jiao, H. Lin, P. Balaji, and W. Feng, "Power and performance characterization of computational kernels on the GPU," in *Proc. IEEE/ACM Int. Conf. Green Comput. Commun. Int. Conf. Cyber, Phys. Social Comput.* Washington, DC, USA: IEEE Computer Society, Dec. 2010, pp. 221–228.
[38] Z. Tang, Y. Wang, Q. Wang, and X. Chu, "The impact of GPU DVFS on the energy and performance of deep learning: An empirical study," in *Proc. 10th ACM Int. Conf. Future Energy Syst.* New York, NY, USA: Association Computing Machinery, Jun. 2019, pp. 315–325, doi: 10.1145/3307772.3328315.
[39] H. Zamani, Y. Liu, D. Tripathy, L. Bhuyan, and Z. Chen, "GreenMM: Energy efficient GPU matrix multiplication through undervolting," in *Proc. ACM Int. Conf. Supercomputing*. New York, NY, USA: Association Computing Machinery, Jun. 2019, pp. 308–318, doi: 10.1145/3330345.3330373.
[40] Q. Wang and X. Chu, "GPGPU performance estimation with core and memory frequency scaling," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 12, pp. 2865–2881, Dec. 2020.
[41] K. Fan, B. Cosenza, and B. Juurlink, "Predictable GPUs frequency scaling for energy and performance," in *Proc. 48th Int. Conf. Parallel Process.* New York, NY, USA: Association Computing Machinery, Aug. 2019, pp. 1–10. [Online]. Available: https://depositonce.tu-berlin.de/handle/11303/9528, doi: 10.1145/3337821.3337833.
[42] J. Guerreiro, A. Ilic, N. Roma, and P. Tomas, "GPU static modeling using PTX and deep structured learning," *IEEE Access*, vol. 7, pp. 159150–159161, 2019.
[43] F. Mendes, P. Tomas, and N. Roma, "Exploiting non-conventional DVFS on GPUs: Application to deep learning," in *Proc. IEEE 32nd Int. Symp. Comput. Archit. High Perform. Comput. (SBAC-PAD)*, Sep. 2020, pp. 1–9.
[44] D. C. Price, M. A. Clark, B. R. Barsdell, R. Babich, and L. J. Greenhill, "Optimizing performance-per-watt on GPUs in high performance computing," *Comput. Sci.-Res. Develop.*, vol. 31, no. 4, pp. 185–193, Nov. 2016.
[45] J. W. Romein, "A comparison of accelerator architectures for radio-astronomical signal-processing algorithms," in *Proc. 45th Int. Conf. Parallel Process. (ICPP)*, Aug. 2016, pp. 484–489.
[46] J. L. Jodra, I. Gurrutxaga, and J. Muguerza, "A study of memory consumption and execution performance of the cuFFT library," in *Proc. 10th Int. Conf. P2P, Parallel, Grid, Cloud Internet Comput. (3PGCIC)*, Nov. 2015, pp. 323–327.
[47] (Mar. 2018). *Power Consumption Measurement With NVIDIA-SMI*. [Online]. Available: https://forums.developer.nvidia.com/t/power-consumption-measurement-with-nvidia-smi/59322/3
[48] NVIDIA. (2020). *Nvidia Management Library*. [Online]. Available: https://docs.nvidia.com/deploy/nvml-api

KAREL ADÁMEK received the Ph.D. degree in theoretical astrophysics from the Silesian University in Opava, Czech Republic, in 2016. During his Ph.D. study, he worked on accretion processes around compact objects. From 2015 to 2019, he was a Postdoctoral Research Associate with the Oxford e-Research Centre, Department of Engineering Science, where he was working on accelerating algorithms on GPUs for the square kilometer array radio telescope. He is currently a Postdoctoral Research Assistant with the Faculty of Information Technology, Czech Technical University.

JAN NOVOTNÝ received the Ph.D. degree in theoretical physics and astrophysics from the Silesian University in Opava, Czech Republic, in 2017. He has worked as a Postdoctoral Research Associate with the Oxford e-Research Centre, Department of Engineering Science, University of Oxford. There he worked in a team developing accelerated algorithms on GPUs for the SKA. He is currently an Associate Professor with the Institute of Physics, Silesian University in Opava. His research interests include computational algorithms, high-performance computing, and spherically symmetric relativistic stars with a polytropic equation of state.

JEYARAJAN THIYAGALINGAM (Senior Member, IEEE) heads the Rutherford Appleton Laboratory (RAL), Scientific Machine Learning Research Group, Science and Technology Facilities Council (STFC), Harwell. The SciML Group focuses on the development and application of machine learning and signal processing techniques for addressing fundamental scientific problems. Prior to joining STFC-RAL, he was an Assistant Professor with the School of Electrical Engineering and Electronics and Computer Sciences, University of Liverpool, and prior to that at the University of Oxford both as a Postdoctoral Researcher and later as a James Martin Fellow. He has also worked in industry, including MathWorks, U.K. His research interests include machine learning models, data processing algorithms, and signal processing. He is also a Fellow of the British Computer Society. He also serves as an Associate Editor for the *Patterns* journal.

WESLEY ARMOUR received the M.Phys. degree in fundamental particle physics and cosmology and the Ph.D. degree in lattice gauge theory. He is currently the Director of the Oxford e-Research Centre and an Associate Professor with the Department of Engineering Science, University of Oxford. He began his career in theoretical particle physics and then spent several years advancing algorithms and methods in protein crystallography. He then moved to the University of Oxford, where he also leads the Scientific Computing Group at the Centre.