ENERGETIC: Final Report

Project period: 01/08/2022 – 31/12/2022
Project award: DOI:10.5281/zenodo.6787467

Authors: Michael K. Bane, Oliver Brown, Teymoor Ali, Deepayan Bhowmik, Jamie Quinn, David Stansby.
Reviewers:
Release Date: 24 January 2023

Revision table			
Version	**Date**	**Name**	**Comments**
Release for review	24/01/2023	Michael Bane	Initial release
1.1	10/02/2023	Michael Bane	Updated initial recommendation

Table of Contents

1. Executive Summary

The ENERGETIC (ENergy-aware hEteRoGenEous compuTIng at sCale) project set out to look at whether the use of accelerators would give significant energy savings compared to CPU-only computations, and to determine which classes of codes should run on heterogeneous architectures. Furthermore, the project would examine opportunities and challenges for measuring energy in a standardised, portable manner irrespective of architectural platform.

As noted in the original proposal, data centres are key to our digital world but account for a massive 12% of UK electricity consumption every year or 38.54TWh (Booth, Carbon3IT, Mar 2020), costing£4.6B/yr. Regulations in the UK, EU and rest of the world mandate net zero in data centres in the next 20-30 years. The proposal aimed to test whether the use of heterogeneous architectures could significantly reduce the energy-to-solution of common HPC workloads and thus the energy consumed by UKRI data centres. In order to determine potential role of architectures in reducing energy-to-solution, we set out to identify typical algorithms representing a wide range of applications and then to apply micro-benchmarks on relevant target hardware (both in selected existing data centres and stand-alone, controlled compute environments). Furthermore, we set out to review existing energy monitoring frameworks in order to use the most appropriate for this work.

We used the benchmarks to quantify the energy consumption on different architectures, and determined configurations that reduce energy-to-solution across a range of computational problems. Our work can used to estimate the energy used by applications that use significant HPC resources and to provide evidence-based recommendations for the UKRI Net Zero DRI roadmap including future data centre procurements.

During the implementation of the proposal, we expected to identify challenges and barriers to energy monitoring at the application level on shared facilities and therefore planned a workshop involving key stakeholders with the intention of disseminating our experiences, understanding better the current landscape of energy monitoring in compute-focused Digital Research Infrastructure (DRI), and to begin working towards an agreed framework for comparable energy measurements across devices and systems.

The project members are:

- Dr. Michael K. Bane, Lecturer, Manchester Metropolitan University (MMU). PI
- Dr. Deepayan Bhowmik, Newcastle. Co-I
    - Mr. Teymoor Ali, Newcastle. Researcher
- Dr. Oliver Thomson Brown, Edinburgh Parallel Computing Centre (EPCC), University of Edinburgh. Co-I
- Dr. Jamie Quinn, University College London (UCL). Co-I
    - Dr. David Stansby, UCL. Researcher

This Final Report summarises our findings. In terms of energy savings we observed energy savings in each selected benchmark (when run for the largest of the test data sizes) when using an accelerator rather than solely CPU, although the accelerator with lowest energy-to-solution is sometimes GPU and sometimes FPGA.. In terms of measuring energy, we conclude that making such measurements are possible but challenges exist to exposing such data to users on HPC systems.

2. Introduction and Systems Considered by this Project

The project is interested in the energy consumed (where energy is the integration of instantaneous power over time) as the metric directly related to CO2 released in generating that energy. Note that the total Carbon footprint of computation would comprise energy consumed by compute nodes but also by infrastructure i) to cool these nodes, ii) for networking, iii) for storage. The project took a pragmatic approach by considering only the energy consumed by the compute nodes, presuming that savings here will also give proportional savings in cooling. Typically cooling response is proportional (but perhaps non-lineaerly) to power peaks & profile, rather than energy consumed, but the difference only becomes relevant when usage of the device swings rapidly between high and low. The energy consumed by a single compute node itself comprises several components: energy consumed by the central processing unit (CPU), its memory (RAM), by graphics processing units (GPU(s)), by field programmable gate arrays (FPGA(s)), by any attached storage and also by the remaining "dark silicon" (within the compute node, necessary to run the CPU etc). Below we discuss how (and when) it is possible to measure these components. Note that some PSUs (Power Supply Units e.g. transformers) also provide (board-side) data over "i2c" protocol and that some data centres use smart PDUs (Power Distribution Units that typically power several nodes) which could be queried to determine total energy consumption (including transformer losses and all the above components) of a given set of nodes.

The project proposed highlighted two types of systems to be used:

a) **HPC**. We noted we would use the ExCalibur Hardware & Enabling Software FPGA testbed hosted by EPCC augmented by the purchase of a GPU card. This system hosts a variety of Xilinx and Intel FPGA cards. In this project we used the node with an Alveo U280 FPGA and a NVIDIA A100 GPU to run benchmarks. This node had an AMD EPYC 7502 CPU.
   The project also had access to one of UCL's HPC machines, Myriad [Myriad], which has a mixture of Intel Xeon Gold nodes (with some nodes having NVIDIA P100, V100, or A100 GPUs although these were not used within this project). For this project we used Myriad to run CPU benchmarks. All benchmarks were run on a node with an Intel Xeon Gold 6240 CPU, using all 36 available cores.
b) **Workstation**. Newcastle was to purchase workstation comprising CPU host with FPGA and GPU PCIe cards. Various delivery delays meant this did not arrive until past half-way of the project. To mitigate, Newcastle provided access to a similar Linux (Ubuntu) workstation that comprises of an Intel i9-11900KF CPU, NVIDIA A2000 GPU, and a Alveo U50 FPGA.

3. Overview of Measuring Energy Consumption on Heterogeneous Systems

We undertook a short literature survey and practical "hands-on" experimentation to discover and compare approaches to measuring energy consumed on CPU, GPU and FPGA architectures. We give an overview of these approaches in this section.

CPU Cores Energy Measurements

Measuring energy consumed by CPU cores is relatively straightforward.

Intel provides RAPL ("Running Average Power Limit") [see Chapter 14, Volume 3, Intel] which provides accumulated energy consumption (since an arbitrary point in time) for various system domains. This data is updated at approximately one millisecond intervals and is exposed via model specific registers (MSRs), also frequently referred to as hardware counters (HWCs). MSRs may require elevated privileges for read access. The specific domains which RAPL provides data for varies between client processors (Package, Power Plane 0 and Power Plane 1) and server processors (Package, Power Plane 0, and DRAM).
Intel chips are used in various UKRI DRI including Cirrus (EPCC system) and CSD3 (Cambridge system).

AMD also provides RAPL data via MSR, see for example [Schone, 2021]. AMD chips are used in various UKRI DRI including the Tier 1 national supercomputing service ARCHER2, and Tier 2 facilities such as the Sulis high through-put facility [Sulis].

On the EXCALiBUR H&ES FPGA Testbed, Likwid [LIKWID] was used to take energy measurements of the AMD CPU on the compute node which also hosted an NVIDIA A100 and a Xilinx U280 FPGA. Likwid is a performance monitoring framework designed to work across a variety of architectures, including Intel, IBM, and AMD CPUs, and NVIDIA GPUs. One of the tools included in the framework gives access (and an interface) to hardware performance counters, and hence also power measurements, which can be integrated over time by the user to calculate energy.

There were no ARM chips within the test systems of this project. Currently ARM chips are only used in UKRI DRI at specific test facilities, such as Fulhame at EPCC, or GW4 Isambard. Note that although ARM is well known for its low power CPUs, ARM HPC processors use approximately the same power and energy as any other comparable CPUs [MB3D6.9].

GPU Energy Measurements

All GPUs used within this project were NVIDIA GPUs, all of which support nvidia-smi and NVML (NVIDIA Management Library). The former provides a text interface listing various properties of one or more given GPU cards, including the instantaneous power. Polling nvidia-smi frequently and then integrating the power measurements over time will give an estimation of the energy consumed by the GPU. NVML is a C-based API which provides total energy consumption since the last time the driver was reloaded. To find the total energy consumption of a particular job it can be run at the start and end, the difference between the return values taken. This project took the polling nvidia-smi approach since NVML would require either manually instrumenting code or developing a portable wrapper.

NVIDIA GPUs are used in various UKRI DRI including Cirrus [Cirrus], Baskerville [Baskerville], and JADE2 [Jade2].

FPGA Energy Measurements

Xilinx FPGAs (from AMD) can be queried via Xilinx Run Time (XRT), specifically by use of the “xbutil” command which provides data on instantaneous power and current & voltage data for various power rails. This data is also exposed via PCI directories within /sys/devices on the host, providing current draw and voltage in separate files at ~0.1s cadence. These were sampled and logged, multiplied together to get an estimate of power consumption, and integrated over time to get an estimate of total energy consumption.

The project considered primarily Xilinx FPGA platforms due to their prominence over Intel/Alterra FPGA platforms and due to lack of ready access to HPC-level Intel FPGA platforms.
FPGAs are not currently a key element of UKRI DRI, and as with ARM chips exist only in specific test facilities such as the ExCALIBUR FPGA testbed.

Note that we only consider energy to solution and neglect energy to compile the FPGA bitstream (and similarly neglect energy to compile for CPU and GPU). Whilst a typical ~hours’ FPGA synthesis (on a CPU) may consume significant energy, this would become insignificant when consider a production code running thousands of times per synthesis.

Workstation, Node and Cluster Energy Measurements

There are also various options for determining the energy measurement of a full node (whether HPC node or standalone workstation) and of a set of nodes.

Some motherboard PSUs support i2c, IPMI or RedFish protocols to provide instantaneous power consumption of the transformer. These gives the total (CPU, RAM, disk, fans and “dark silicon”) power consumption, either on the output (board side) of the transformer or on the input side (in which case measurements also include any transformer losses).

In a typical data centre, nodes will be powered from Power Distribution Units (PDUs) which are also instrumented, although it is not typical for users to be able to directly obtain such data. For resilience, each HPC node is likely to be able to draw power from two PDUs and each PDU would power different sets of HPC nodes. This makes determination of power provided to a given node non-trivial, although it is possible where a protocol such as SNMP or IPMI is supported. Some nodes also support a Baseboard Management Controller (BMC) which allows (typically) system administrations access to various metrics including power data.

For standalone workstations, a comparative approach is to use a wall socket measuring device such as “UK Plug Power Meter” [UKplug] or the UK equivalent of Kill-A-Watt [KillAWatt].

The above approaches are beyond the scope of the project. However, in order to ground truth the RAPL/nvidia-smi/xbutil approach of ENERGETIC we also obtained intrusive measurements as described in Section 4.1 “Hardware/Software Measurement Approach”.

Software Interfaces to Obtain Energy Measurements

In addition to above there are several other approaches to measuring energy consumption via software.

The limited scope of this project meant that these could not be practically explored but for completeness we give a short summary and references for each. We would recommend that future work explores these in more depth.

On a Linux system, the **sensors** suite of utilities can be employed to examine available sensor data. Depending upon hardware and Linux kernel configuration this may include instantaneous power consumption of the CPU and other system elements, and superuser access may be required. Typically, “lm_sensors” or some X11 client interface would be used to expose such data. Similarly, applications such as Open Hardware Monitor [Open Hardware Monitor] and hwmonitor [CPUID] can be downloaded and run on MS Windows, exposing available data. Many such applications exist for Windows, see https://www.techspot.com/article/2009-monitoring-analysis-benchmarking-software-apps/, with varying access for CPU and GPU data. In all the above cases, for Linux and Windows, these utilities do not currently provide power nor energy data for FPGA hardware.

PAPI, the Performance Application Programming Interface, is a performance-monitoring library that includes an API and some standalone commands, for measuring power on CPUs and GPUs. [PAPI]

GEOPM, the Global Extensible Open Power Manager, is “a framework for exploring power and energy optimizations targeting heterogeneous platforms. The GEOPM package provides many built-in features. A simple use case is reading hardware counters and setting hardware controls with platform independent syntax using a command line tool on a particular compute node. An advanced use case is dynamically coordinating hardware settings across all compute nodes used by a distributed application is response to the application's behavior and requests from the resource manager.” [GEOPM]

PowerAPI aims to bring relevant actors together to get a communally accepted API that can help drive energy efficiency at various levels within an HPC system stack, see [Laros et al] and [PowerAPI].

PowerStack aims to work with **“**experts from academia, research laboratories and industry in order to design a holistic and extensible power management framework, which we refer to as the PowerStack. The

PowerStack explores hierarchical interfaces for power management at three specific levels: batch job schedulers, job-level runtime systems, and node-level managers. Each level will provide options for adaptive management depending on requirements of the supercomputing site under consideration. Site-specific requirements such as cluster-level power bounds, user fairness, or job priorities will be translated as inputs to the job scheduler. The job scheduler will choose power-aware scheduling plugins to ensure compliance, with the primary responsibility being management of allocations across multiple users and diverse workloads. Such allocations (physical nodes and job-level power bounds) will serve as inputs to a fine-grained, job-level runtime system to manage specific application ranks, in-turn relying on vendor-agnostic node-level measurement and control mechanisms." [PowerStack]

Variorum is "an extensible, vendor-neutral library for exposing power and performance capabilities of low-level hardware knobs across diverse architectures in a user-friendly manner." [Variorium]

It is important to note that there is much active research into measurement and reduction of energy consumption of HPC facilities, see [InsideHPC] and the working groups of the Energy Efficient HPC Working Group [EEHPCWG] of which Bane is an active member.

4. ENERGETIC Workshop at CIUK

On Friday 2nd December 2022, the ENERGETIC project held a discussion workshop collocated with Computing Insight UK (CIUK) in Manchester on 1-2 December 2022 [CIUK]. There were around 25 attendees to the ENERGETIC workshop, with UKRI Digital Research Infrastructure (DRI) users, system administrators, and hardware vendors represented. After a short introduction, the workshop divided in to three groups, each focused on one discussion topic.

The three discussion prompts presented to participants (to discuss one per group) were:

- Energy advantage.
    - Some applications may consume less energy overall when run on one or more accelerators (i.e. GPUs, FPGAs). Which types of applications? Which accelerators?
- Fair comparison across architectures.
    - Energy benchmarking frameworks should be standardised to compare fairly across architectures.
- User-level energy benchmarking.
    - Energy usage is highly dependent on specific job, so users should be able to benchmark their own jobs' energy usage across UKRI DRI.

As is natural with such discussions, the scope of responses was wider than suggested by the prompts. Members of the ENERGETIC project were on hand to moderate and write notes on the discussion, but the decision had been taken beforehand not to enforce the topics too strictly. Discussion relevant to the wider ENERGETIC project was allowed and recorded. Below is a summary of the discussions in each group. This may not represent the views of the ENERGETIC project.

Energy advantage

- The applicability of accelerator devices can be limited by the scale of the problem being tackled. Weather forecasting for example, which requires whole-earth models, needs large distributed computers. Accelerated systems at that scale are a very significant investment, and no such machine is currently available in the UK [TOP500]. On the other hand, such systems (with GPUs) do exist, and currently dominate the top 10 largest (by compute capacity) supercomputers worldwide.
    - Users may be forced to use less energy efficient architectures by the scale of their problems.
    - Equivalently, the most energy efficient architecture will be dependent on the scale of the problem. Small problem sizes may run on devices with limited available memory, such as FPGAs, but sufficiently large FPGA systems may not exist for larger problem sizes.

- When overheads with a fixed energy cost such as cooling and memory dominate, the problem of minimising energy efficiency is equivalent to the problem of maximising parallel efficiency.
    - Problems should be packed into as few nodes/accelerators as possible, to minimise overhead energy cost.
    - Gaining energy advantage on a given architecture requires maximising the usage of the nodes/accelerators throughout the runtime of the job.
- Programming models and programmability can be a major barrier to energy efficient use of accelerators. FPGA programming is particularly hard, but efficient GPU code is also challenging to write.
    - Inefficient use of an accelerator will never be energy efficient, but well optimised implementations require advanced programming skills/knowledge.
    - Avoiding memory or communication bound implementations may be challenging or even impossible for some algorithms.
    - Non-trivial to exploit the host CPU while the accelerator is being used (thus avoiding the host idling in terms of computation (but consuming some energy) while waiting for data from the accelerator).
    - Better tools and skilled RSEs required to design and implement energy efficient codes (see Green Software Engineers discussed in the "fair comparison across architectures" group)
- As well as considering individual codes, we should consider multi-program workflows/pipelines. Smart workload management tools, could identify the most energy efficient partition of a heterogeneous system to run individual components on.
    - The challenge here is developing the model on which the workload manager relies.
    - Within codes this same model could be adopted by tasking frameworks which use Petri nets to map task dependencies and available resources, such as Fraunhofer's GPI-Space [EHSD4.2].
- General agreement that data on energy efficient applications and hardware should be shared.
    - Perhaps something similar to the Kaggle project, which shares datasets and code for machine learning [Kaggle].

Fair comparison across architectures

- Agreement that it is important to capture the energy usage of the whole node.
    - Multiple nodes is more of a challenge as unclear how to handle shared resources such as the network and the filesystem.
- Methodology behind energy & power measuring tools not always clear.
    - Typically relies on hardware counters, but not always clear (to users) what *exactly* that means.
    - Not clear to what extent the measurements encompass the whole node.
    - Energy is key, but hardware counters (typically) provide power measurements which must be integrated.
    - In distributed systems, a parallel reduction would be required to produce a single average or total number (or all nodes would need to be individually reported).
    - Sample rate may be important for uneven workloads. No standard for this, may be up to the user to determine what is necessary and implement their own sampling solution.
- Variation in what's possible across platforms, as power measurement is implemented by hardware vendors, with no formal standard interface (or methodology).
    - Standardisation of interface and methodology would be very useful.
    - As an example, the ENERGETIC project was unable to make use of the Cirrus facility at EPCC for benchmarking, as access to the energy encounters was not available in the OS. The OS is

not standard, and is instead provided by the hardware integrator, HPE. The systems team have raised a case with HPE in the hopes that this can be fixed.

- Ideally, the way in which energy will be measured will be specified at the procurement stage of UKRI DRI, in order to ensure that the functionality is consistently available.
- Properly comparing energy usage across architectures may not be entirely automatable, and may require training of users.
    - The group suggested that Green Software Engineers (GSEs) be trained to do this kind of work, as well as developing and deploying energy efficient codes, and training others.

User-level energy benchmarking

- Slurm can be configured to give per-node power information, but not more refined data than this currently.
    - Requires hardware counters to be supported (for example through IPMI on Intel architectures), and operating system support, as noted above.
    - More sophisticated energy monitoring plugins could be written for slurm and incorporated in the future.
- Power data from the PDU is time consuming for system administrators to set up.
- Accuracy of information is important, as universities are focused on running costs.
- Many tools exist, but they are not refined or user-friendly enough.
    - There is no standard interface, and they vary in quality.
    - There is a lack of education for users/students about how to benchmark their own energy usage (see above comment re GSEs providing training)
    - Tools do not always expose what their methodology is or which assumptions are embedded in their measurements.
- Examples of tools:
    - Intel RAPL, a standard tool on Intel architectures.
    - Likwid, a more generic tool which supports many architectures.
    - VTune, an Intel profiling tool.
- An important point, and sometimes one of contention, is that there are security considerations related to energy benchmarking, as a consequence of the access to hardware counters.
    - Important that whole node is reserved for the one user doing benchmarking (node exclusive) -- this is also important for ensuring accurate energy benchmarking.
- HPE Cray are generally good at providing lots of information on energy usage.
- As also discussed in other groups, it is unclear how to deal with shared resources such as the network and storage.
- Memory access patterns likely to impact energy efficiency.
    - Certainly they impact performance, and the two are linked.
- Hard, and maybe undesirable, to try and quantify “science” to try and create “science-per-Joule” metric, but should be possible to compare one code across many architectures.
- Characteristics of an ideal tool:
    - Would provide time series data per job, including CPU, memory, and IO usage and energy usage.
    - Would export data to users without root privileges.
    - Would allow live time series that would enable on the fly configuration to take place.
    - Would support a range of accelerators, and even be able to suggest the best choice.
    - Would enable optimal frequency selection per job (based on CPU/memory/communication boundedness).

- Would be great to see standardisation of energy data between hardware vendors to better enable comparison.
- Tools would need to be standardised if (when) journals or grants began requiring energy usage data.

Conclusions

In spite of the disparate discussion topics, common themes appeared across all groups.

1. Standards for energy measurement across platforms are highly desirable.
2. Better tools are needed to support those standards, as well as support less experienced users.
3. There is a need for better education for UKRI DRI users on energy efficient computing.

There was broad agreement that standards for energy measurement across platforms are highly desirable. Since there are various valid choices concerning the specific implementation of energy measurement, it is very important that the chosen methodology is clearly documented. It is often not clear exactly what is being measured by energy measurement tools, or how. An ideal situation would be that all UKRI DRI systems had the same (or equivalent) tools available to consistently and compare energy usage across systems. This would need to be mandated in procurement process and would require support from hardware vendors and integrators, as energy measurement needs to be supported in both the hardware and operating system as well as by application-level tools. Even system administrators may not have access to the information required to monitor energy usage at the level of a particular job, or even a single node.

It was broadly agreed across discussion groups that better tools are needed. There are high-quality tools which support specific architectures, or even multiple architectures, but only one at a time. The user is responsible for implementing whole-node energy measurements (or as close to it as the available tools allow). Integration with the job scheduling tool Slurm (as on ARCHER2 [A2DOCS]) creates a very user-friendly experience, as it requires no additional effort from the user and makes the data available asynchronously within 24 hours of job completion, but does lack transparency about how the data is collected. There was no known or suggested standard way to include the use of shared resources such as the network and storage in the energy usage.

The final theme which occurred across discussion groups was the need for education on all aspects of energy efficient computing. Users who wish to make their own scientific computing 'greener' may be unsure how to make appropriate measurements of their energy usage (in part due to the complexity of tools), how to interpret the data they may gather, or what to do about it once they have the data. One suggestion was the development of 'Green Software Engineers (GSEs)'. Given the finding of the wider ENERGETIC project that even on heterogeneous systems parallel efficiency is still a good proxy for energy efficiency, the skills of a GSE would not be dissimilar from those of Research Software Engineers (RSEs) or HPC developers, but there is room for specific training on energy measurement and reduction, and the impact of hardware choices.

The ENERGETIC project thanks the organisers of CIUK 2022 for hosting the ENERGETIC workshop, and to all attendees for contributing to the valuable discussion.

5. Energy Measurement Tools and Techniques

While a plethora of energy measurement tools are available, they are predominantly compute-architecture specific. A general overview of such options for measuring energy on different architectures was given in Section 2. However, a generic methodology for measuring energy in an orderly fashion for HPC systems that covers heterogeneous hardware is currently missing but of greater interest to the UKRI Net Zero DRI scoping project. This is somewhat challenging given that the wider variety of hardware available in various HPC clusters.

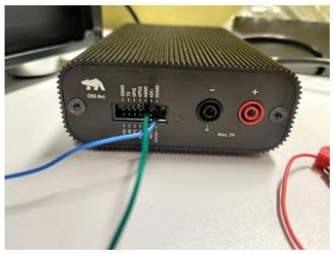
Within the scope of ENERGETIC we address this by curating various methodologies that are used for the experimental purposes showing the effectiveness of those choices. In addition, we've experimented with physical measurements using power meters and data loggers. This section provides details of the approaches taken within the ENERGETIC project, for each of the HPC and workstation scenarios.

Given that, we have access to only Intel CPUs, NVIDIA GPUs and Xilinx FPGAs, we make use of corresponding tool sets, i.e., RAPL, nvidia-smi and xbutil, respectively. For the standalone system, measurements were supplemented by physical power measurements. Within the scope of this work, we performed experiments on HPC systems hosted at EPCC and UCL, and one stand-alone workstation at Newcastle, with combinations of CPU, GPU and FPGAs. The specifications of these machines are given in Section 2 and summarised in Table 1, below, along with the measurement approaches employed.

Machine	Classification	Configuration	CPU energy	GPU energy	FPGA energy
Excalibur H&ES FPGA testbed (NextGenIO) at EPCC	HPC	AMD CPU, NVIDIA GPU, Xilinx FPGA	Likwid	Polling nvidia-smi and integrating over time	xbutil
Myriad at UCL	HPC	Intel CPU	RAPL	N/A	N/A
Newcastle machine	Workstation (Linux boot)	Intel CPU NVIDIA GPU Xilinx FPGA	RAPL	Polling nvidia-smi and integrating over time	Polling xbutil and integrating over time
	Data logger		Clamp over CPU => instant power or can add from output	Clamp over GPU => instant power or can add from output	Clamp over FPGA => instant power or can add from output

Table 1: Computing architectures and their energy measurement methods used in ENERGETIC

While the software-based measurements are vendor provided and standard, the methodology used in physical measurement require some description which was developed in this project. It is to be noted that physical measurements capture the real-life data while the software-based approaches are the best estimates, albeit fairly accurate estimates. One of the key differences are the physical measurements also include the energy consumed by other system electronics, which is reflective of the overall energy usage. However, the physical approach is intrusive and requires access to power inputs which is not always feasible and probably not required for the holistic estimation. Our experiments with both type of measurements ensure the reliability (and the differences) of both measurement techniques so that informed recommendations can be made.



Hardware:
- Current Clamp
- Otii arc 3 datalogger
- Multimeter

Power Measurement Software:
- CPU (RAPL)
- GPU (NVML)
- FPGA (XBUTIL)

Figure 1: Physical energy measurement using clamp meter and data logger

Physical Measurement Approach

To complement the software-based energy measurement the project employer a power analyser with data loggers along with current clamp meters (and other peripherals) for the physical energy measurement of the standalone machine at Newcastle. The overall set up and the list of required tools are shown in Figure 1 and Table 2, respectively.

Hardware/Software Tool	Description
Qoitech AB Otii DC Power Analyzer	Otii Arc is a small, portable power supply, a current and voltage measurement unit and a data-acquisition module.
RS PRO RS CP-10 Current Clamp	Current Clamp is a transducer which will allow the datalogger to measure low electrical or/and electronic current up to 80 amperes AC/DC. Clamp measure conversion (10mV/1A)
AstroAI Digital Multimeter 6000	The multimeter is designed to measure current, voltage, and resistance.
Wall Power Monitor	Plug-in type power meter socket measures and displays power consumption of a connected appliance

Table 2: Standalone Machine Data Capture Tools

Once the algorithm was implemented for the selected architecture, the current clamp was connected to the datalogger using 4mm banana plugs into the 'SENSE+' and 'AGND' pins. The datalogger software was launched within the Linux operating system to allow the user to start/stop logging. An example of the software interface is shown in Figure 2. The current clamp was attached to the non-insulated part of the cable from the power supply to the chosen architecture (see Figure 1).

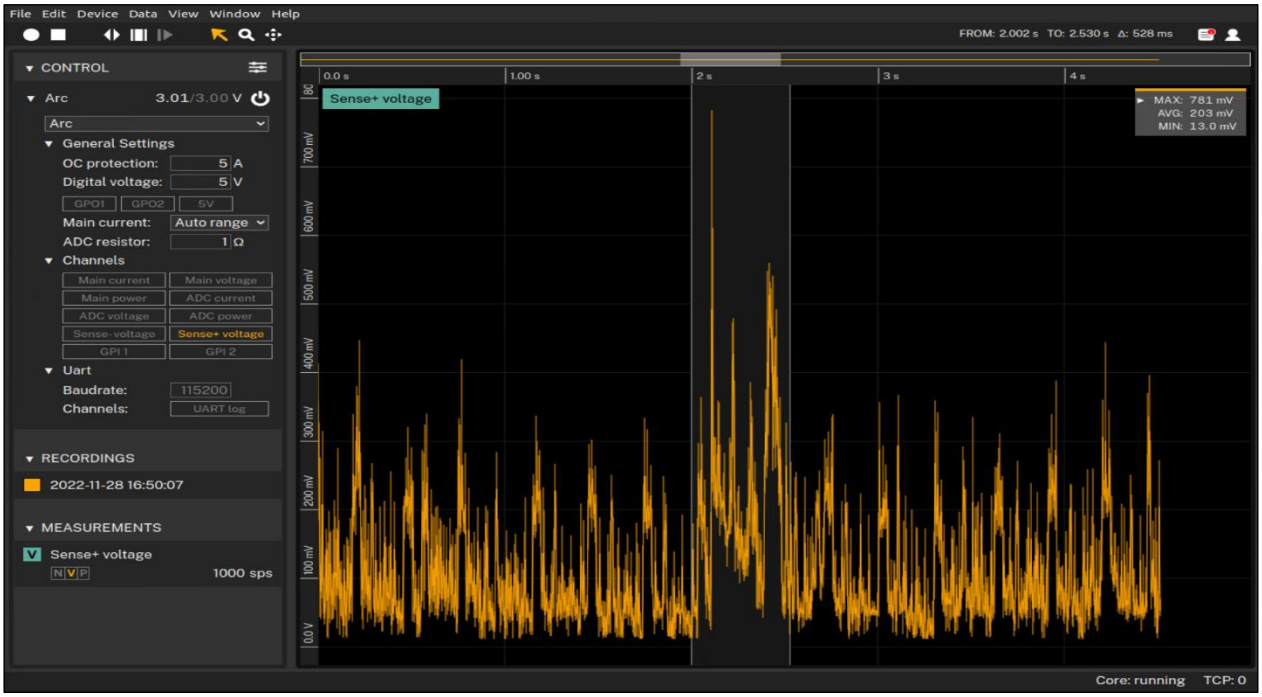


Figure 2: Qoitech AB Otii DC Power Analyzer data logging software interface

In terms of operating procedure, the algorithm and datalogging software were run at the same time, the clamp meter measures the current and outputs a voltage signal. As specified in the data sheet, for every 1A measured the clamp outputs 10mV which is recorded by the datalogger. The datalogging software displays a line graph showing the current consumption over time. The mean current value is automatically computed from the software by using the start and stop timeline markers for the time the algorithm is executed. The current value is multiplied by the constant voltage (12V) supplied by the power rail to give the power. The total energy for the benchmark algorithm is computed by multiplying the mean power by the time it took to execute the algorithm.

Software-based Measurement Approach

The software approach uses energy measurement APIs provided by hardware manufacturers to read instantaneous power (Watts). A shell script wascreated to sample (every 0.1s) the power values from the API into a CSV file while the algorithm is being executed. The measurement approach varies depending on the device, RAPL gives energy consumed by the CPU directly which is written to a binary file found in the '`/sys/class/powercap/intel-rapl/intel-rapl:0`' directory. The Newcastle implemented bash script to read the RAPL values takes the initial value of the file and final value while the algorithm is being executed, and subtracts them to determine the energy consumed while the algorithm was running.

The nvidia-smi command '`nvidia-smi --query-gpu=power.draw --format=csv --loop-ms --filename=gpu_utillization.csv`' creates a CSV file of the power values being sampled every millisecond, and a script started and stopped this command while analgorithm was executed. The FPGA uses xbutil command '`watch -n 0.1 xbutil examine --device 0000:03:00.1 --report electrical`', with a similar process to the GPU measurement using a script to start and stop the reporting.

To make fair comparison, the software/hardware measurement approaches on the workstation were completed at the same time for all algorithms executed on each architecture. This allows us to create ground-truth data from the physical measurement to compare with estimated energy from the software power models (RAPL, nvidia-smi & xbutil). In addition to this we also used the wall power meter which measured the total energy consumed by the whole machine.

Note that we did not have physical access to either HPC facility, and did not have time to organise access to PDU data with facility managers.

in Figures 3 and 4, we compare mean measurements for each approach and observe a common pattern in relative values, if not in absolute values. (We additionally provide comparative box whiskers plots in Appendix). The disparity between both software and physical measurements are due to various factors.

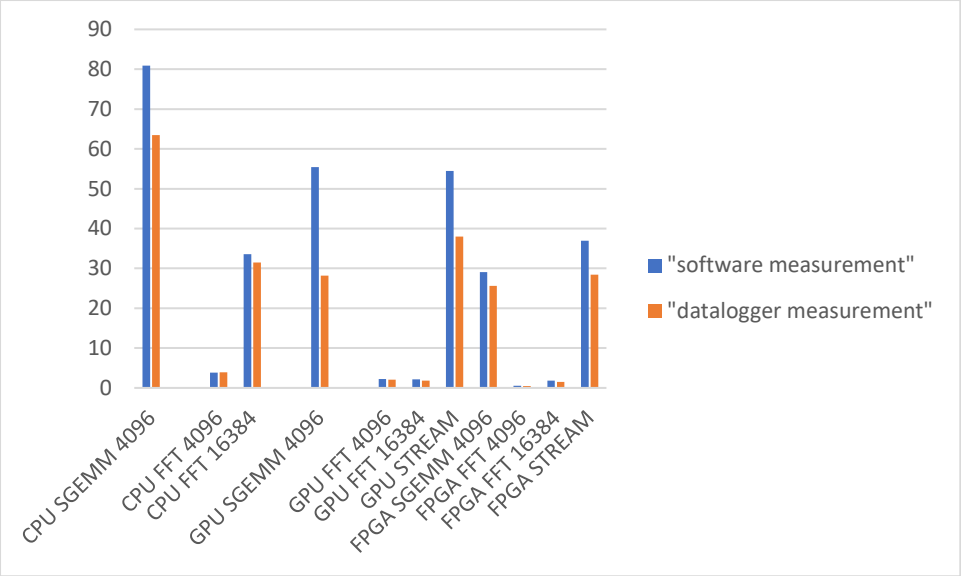


Figure 3: Software & Datalogger measurements for benchmarks

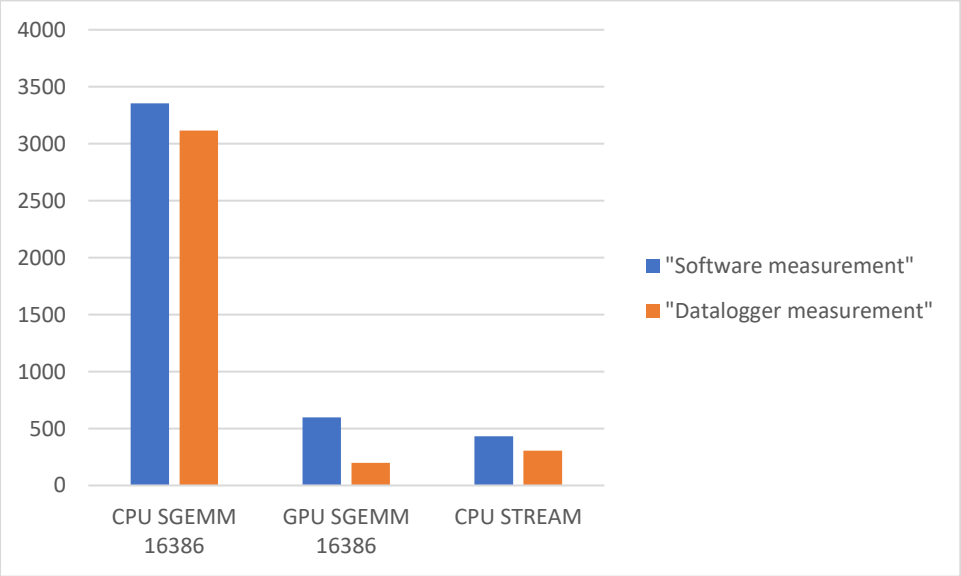


Figure 4: Software & Datalogger measurements for benchmarks

Firstly, there may be electromagnetic interference (EMI) from other nearby conductors (specifically other cables) which may affect data being logged. Although, care was taken to isolate the cables while measuring the power. Additionally, Software-based measurements rely on algorithms to estimate power based on the readings from sensors which may lead to overestimations.

6. Analysis of Energy Consumption for Representative Benchmarks on Heterogeneous Systems

The project aimed to test whether the use of heterogeneous architectures could significantly reduce energy to solution of typical HPC workloads and thus the energy consumed by UKRI data centres. In order to determine the potential role of architectures in reducing energy-to-solution, we tried to identify prototypical algorithms representing a wide range of applications and apply micro-benchmarks on relevant target hardware (both in selected existing data centres and stand-alone controlled compute environments).

As part of this process, we employed selected benchmarks to quantify the energy consumption on different architectures, and determined the optimal hardware configuration to reduce energy-to-solution. To identify particularly significant algorithms (and thus inform the choice of benchmarks) we choose to characterise sample applications in terms of Berkley's thirteen dwarves' symbolic computation [Asanovic et al], a categorisation that describes various types of parallel computations relevant to HPCs. The sample applications were drawn from a list of applications which use significant amounts of HPC resources in the UK, mainly on the UK's national service ARCHER2 and its predecessor ARCHER. While this leads to an ideal list for benchmarks, a more pragmatic approach was taken considering the short life span of this project; we focused on the algorithms that are present in the HPC challenge benchmark suite [hpcc] that have complete implementations targeting our architectures of interest: CPU, GPU and FPGA. The main challenge was finding appropriate ports available for the specific FPGA cards used by ENERGETIC and reliance upon some level of optimisation having been performed. The project timescales excluded porting and optimising, and the debugging that was still necessary was constrained by FPGA development requiring a long synthesis time, in our cases sometimes taking several hours.

While this approach provides a set of benchmarks to focus our testing upon, we additionally considered convolutional neural network as computer vision is becoming mainstream, can require massive amount of computational effort and can be considered representative of emerging deep learning paradigms. Additionally, further experiments were conducted on embedded systems for all three target architectures.

In this work we have used four different representative benchmark algorithms, namely, SGEMM (representing matrix multiplication), FFT (Fast Fourier Transformation representing spectral methods), STREAM (measuring memory bound computation and communications) and CNN (Convolutional Neural Network representing emerging machine learning techniques for computer vision applications). Further details of the algorithms and their association with thirteen Dwarfs are given in Table 2.

Algorithm	Description	Dwarf association
SGEMM	Single precision GEneral Matrix Multiply measures the floating-point rate of execution of single precision real matrix-matrix multiplication.	#1 Dense Linear Algebra
FFT	Fast Fourier Transforms convert data from a real domain to a spectral (or frequency) domain.	#3 Spectral Methods
STREAM	A simple synthetic benchmark program that measures sustainable memory bandwidth (in GB/s or MB/s) and the corresponding computation rate for a simple vector kernel.	#1 Dense Linear Algebra
CNN	Convolutional Neural Network provides the framework for emerging deep learning based methods for computer vision tasks, e.g. classification and segmentation	#6 Unstructured Grids #10 Dynamic Programming

Table 3: Benchmark algorithms used in ENERGETIC, their description and association with Dwarfs.

The specific ports we used are shown in Table 4, below:

	CPU	GPU	FPGA
GEMM	SGEMM sample benchmark [Intel-gemm]	NVIDIA port [cublas]	Xilinx Vitis benchmark [vitis_gemm]
FFT	FFT sample benchmark [FFT]	NVIDIA sample [cuFFT]	Xilinx Vitis benchmark [vitis_fft]
STREAM	Original CPU version [stream]	Cumming port [cuda-stream]	HPCC_FPGA port [hpcc-fpga]
CNN	Pytorch [pytorch]	Pytorch CUDA [pytorchC]	Xilinx Vitis port [cnn]

Table 4: Detail of implementation ports used for benchmarks.

The CPU clock frequency on both the nextgenio and Myriad CPUs is fixed. For the workstation runs, 10 measurements were made for each benchmark and the mean calculated and reported. Given the size of the benchmark problems, the benchmarks were measured on the HPC systems only once each and those runtime and energy measurements used in the analysis. GEMM specifically was tuned using the benchmarks' exposed parameters to optimise performance on the target HPC systems. Care was taken to have exclusive use of the relevant HPC nodes.

A full set of the scripts employed by the project is available from the Energetic github site [Energetic]. We have also published scripts and raw data at [e-space].

Data Summary

We've obtained results for the benchmarks defined above. For each of these algorithms, we have used multiple parameters to test the stability of the outcome. The algorithms were executed on two different HPC systems, EPCC's NextGenIO and UCL's Myriad, and ENERGETIC's standalone computing infrastructure hosted at Newcastle. Wherever applicable, HPC hardware architectures include Intel Xeon Gold 6240 CPU, NVIDIA A100 GPU and Xilinx/ARM Alveo U280 FPGA. For the standalone computing platform at Newcastle, we have used Intel i9-11900KF CPU, NVIDIA A2000 GPU and Xilinx/ARM Alveo U50 FPGA. Run time, software and physical energy measurements are carried out as described in Section 5.

Hardware	Clock
CPU: i9-11900KF	3.50 GHz
CPU: Xeon Gold 6240	2.60 GHz
GPU: Nvidia A2000	1200 Mhz
GPU: Nvidia A100	1095 Mhz
FPGA: Xilinx Alveo U50	Kernel: 300 Mhz
FPGA: Alveo U280	Kernel: 300 Mhz

Table 5: Clockspeed of hardware used.

SGEMM

The single precision general matrix multiply benchmark is a micro-benchmark that tests one of the most basic operations in linear algebra, an operation which is used as part of many algorithms in HPC. It helps inform when accelerators are appropriate for processing computations involving dense linear algebra.

This benchmark was run using two different problem sizes, a matrix size of 4096x4096 and a large size of 16384x16384. Only the smaller problem size was able to be run on the smaller FPGA, the U50, and only the larger problem size was run on the larger GPU, the A100. When using accelerators, it is possible to measure the energy used by the accelerator and the CPU independently as the benchmark runs, giving a useful measure of how much energy a host CPU uses while supporting a computation on an accelerator.

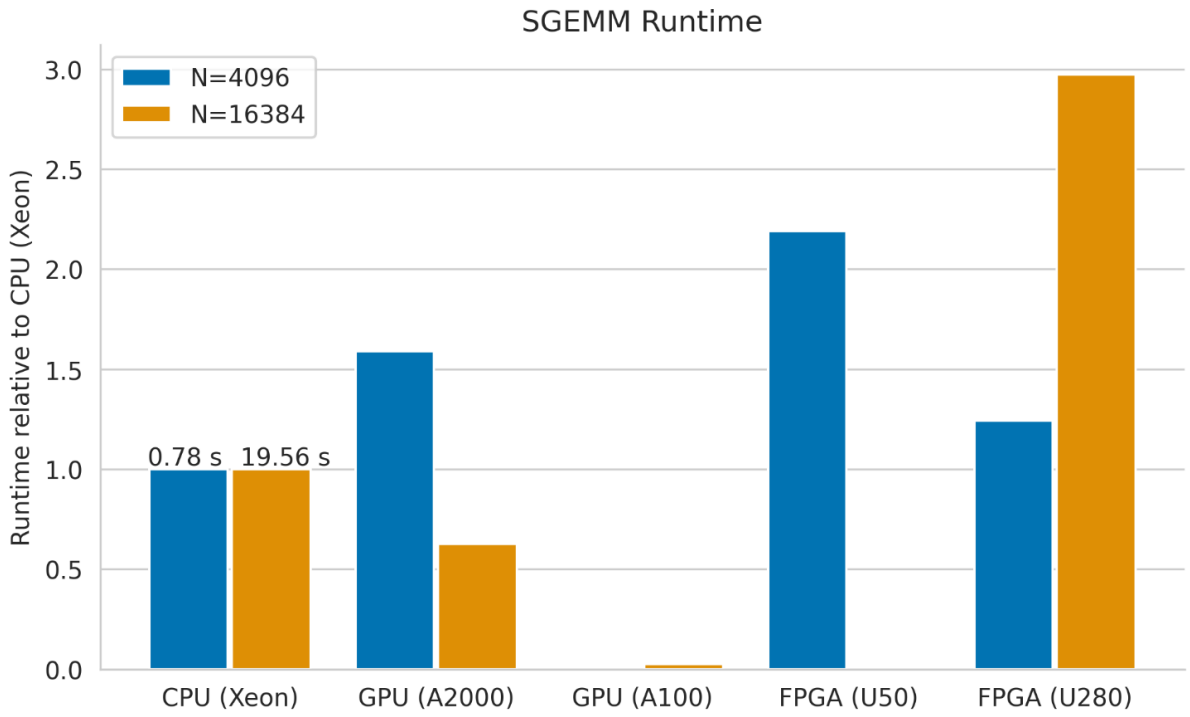


Figure 5: Relative run times of SGEMM benchmark.

Figure 5 shows the runtime of each benchmark on each target architecture normalised to the CPU runtime for comparison between devices and problem size. Since this is execution time, lower is better. The measured runtimes for the CPU benchmarks are displayed above the respective bars. Clearly the FPGAs struggle in this benchmark, suggesting the particular algorithm used in this benchmark does not suit the dataflow parallelism inherent in the FPGA. The GPU sees significant speedup over the CPU only for the large problem size, indicating that the smaller problem size cannot efficiently utilise the massive parallelism of either GPU. The maximum speedup we found when using the larger problem size was when using the A100 GPU, giving a speed up of nearly 40x (3912%) relative to the CPU implementation.

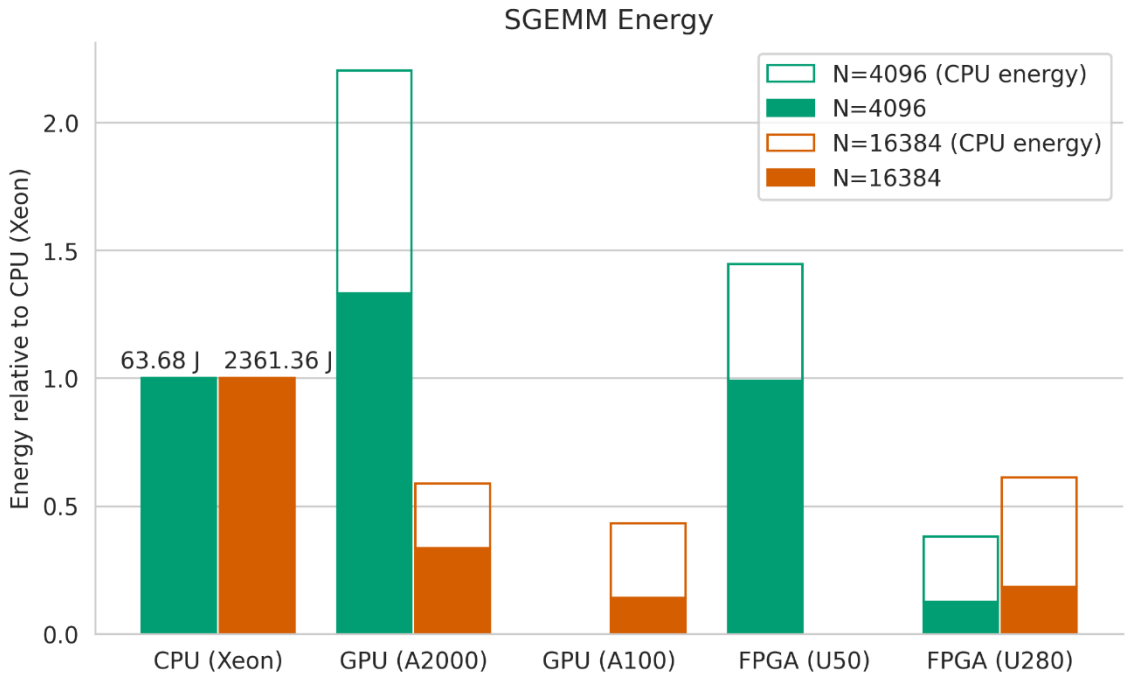


Figure 6: Relative energy consumption of SGEMM benchmark.

Figure 6 shows the relative energy of the same benchmarks as those shown in the runtime plot, again normalised to the energy consumed by the CPU during the CPU-only benchmarks. For each architecture we plot two bars, the green bar on the left represents the energy used while running the smaller problem size (N=4069) and the orange bar on the right, the larger problem size (N=16394). During the accelerator benchmarks both the accelerator and CPU are active and use energy; the energy used by the accelerator is represented by the solid portion of each bar and that used by the CPU is represented by the clear portion. In the CPU benchmarks all energy is used by the CPU. It is clear the CPU uses significant energy while the relevant computation is performed on any of the accelerators, indeed it is so significant when using the U280 FPGA that the CPU uses nearly three times as much energy as the FPGA itself, despite not performing any of the computation. Despite this, and despite the relatively long runtime of the U280 benchmarks, it still manages to use far less energy than the CPU-only benchmark. Again, we see the effect of the problem size on the A2000 where the long runtime in the small problem size case results in more energy being used by both the GPU and CPU. This is further compacted by the host CPU (an i9 consumer grade CPU; results not shown) being less energy efficient than the presented Xeon. For the large problem size, the GPU is both faster and more energy efficient. The smaller FPGA, the U50, struggles to compete in either runtime or energy efficiency, suggesting that if more powerful hardware is available to a user, and their problem size is large enough to make efficient use of it, they should opt for the more powerful (and in this case, newer) hardware. The maximum energy efficiency gain when using the large problem size was when using the A100 GPU, showing a reduction factor of 2.32x compared to the CPU implementation.

FFT

Like the SGEMM benchmark, the 2D FFT benchmark was run using single precision on all devices and two problem sizes were tested, again using matrices of size 4096x4096 and 16384x16384. This particular benchmark was only run on the Newcastle standalone machine. It was performed 10 times per run and presented here are the mean runtime and energy of those ten runs.

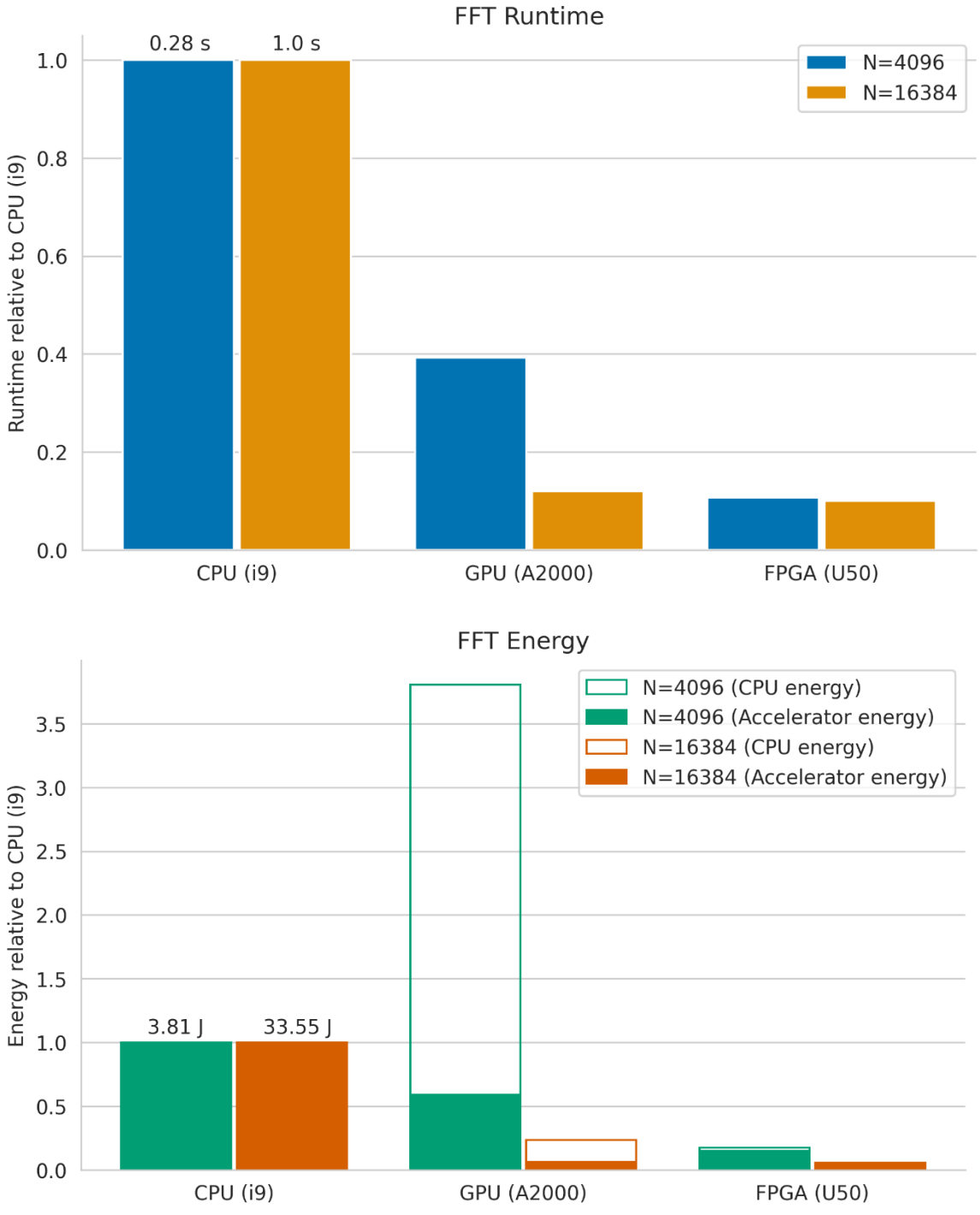


Figure 7: Relative run times (top) and energy consumption (bottom) for FFT benchmark.

Figure 7 shows the runtime and energy for the 2D FFT benchmark across the architectures available in the standalone machine, where again the measurements are presented normalised using the relevant

measurements from the CPU-only benchmark. Both accelerators are more time and energy efficient than the CPU for this particular benchmark with the exception of the large energy used by the A2000 for the small problem size. It is unclear why the CPU uses significantly more energy than any other case, however the GPU itself does show significant energy savings over performing the computation on the CPU. Overall, the FPGA seems to perform the best over both problem sizes, using less total energy than the A2000 in either case. With its small runtime, the effect of CPU energy use is minimised. Overall, the best-case improvement in both energy efficiency and runtime was when using the U50, reducing the runtime by a factor of 10x and the energy by a factor of 17x for the large problem size.

STREAM

The STREAM benchmark differs from the SGEMM and FFT in that it primarily measures the effect of memory bandwidth, not computational power. In the CPU implementation, this measures the performance of the main memory, while in both accelerator implementations this measures the performance of the device memory, not the host memory. This allows us to ask questions about codes which are primarily memory-bound, that is their performance is limited by memory bandwidth.

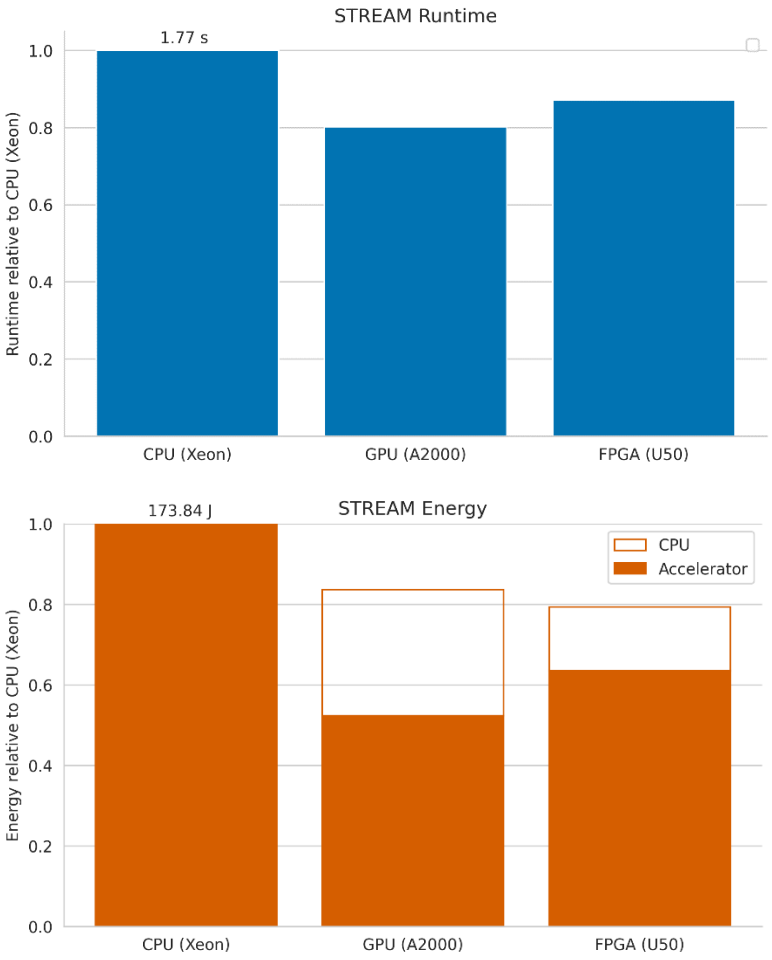


Figure 8: Relative run time (top) and energy consumption (bottom) of STREAM benchmark.

Figure 8 shows the runtime and energy use of the STREAM benchmark across three architectures, the HPC CPU (Xeon), and the two accelerators found in the standalone machine, the A2000 GPU and U50 FPGA. While the accelerator benchmarks are running there is relatively little required of the CPU, so the energy used by the CPU can be considered waste energy that could be minimised by using a lower-powered CPU. The lower runtime and better energy efficiency of the accelerator benchmarks show that memory-bound

problems can be time and energy efficient on these devices, mainly due to the (typically) higher bandwidth memory used on these devices. Due to the limited possible gains here, the accelerators are able to improve on the runtime and energy use of the CPU implementation only moderately, with the A2000 providing a 24% speedup over the CPU and the U50 providing an energy reduction of 26%.

CNN

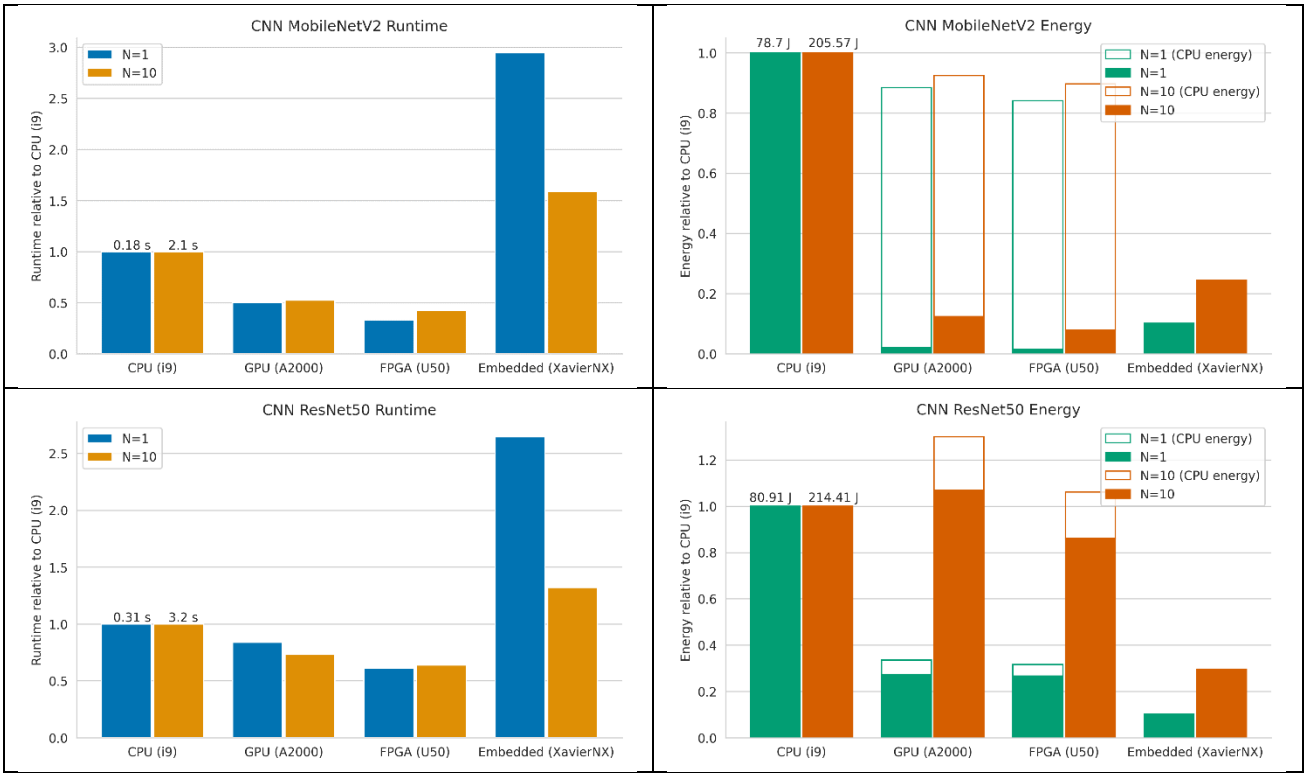


Figure 9: Comparative run times (left) and energy consumptions (right) of MobileNetV2 (top) and ResNet50 (bottom) CNN benchmarks.

Convolutional neural networks (CNN) have become ubiquitous in many fields, making them suitable as a benchmark between architectures. The convolution layers implemented in many CNN architectures are the most critical as their operations constitute the majority of the total computation time, which has the potential to be accelerated.

The CNN energy consumption and inference time for each CNN architecture on each accelerator are shown in Figure 9, above. The results also include an emerging processor called 'Nvidia Jetson Xavier', aimed at embedded systems. The technology is based on Nvidia's Ampere architecture for discrete graphics processing units and has an integrated ARM core. Finally, *N* is the number of *3840 x 2160* images processed by the CNN. In this case, the graphs show results for one image and ten images, respectively.

The results for MobilenetV2 architecture show that the FPGA has a 2.35x speedup compared to the CPU and outperforms other architectures on runtime. Although, in comparison to the GPU, there was a minor runtime difference between the FPGA. On the other hand, considering the power consumption shows that the Xavier NX leads with a 4.1x energy improvement but at a high cost to runtime due to fewer processing cores available and lower clock speed. Additionally, the CPU power consumed while transferring data or waiting for GPU or FPGA to execute the algorithms is large enough that the total energy consumed is nearly equivalent to the single I9 CPU. Finally, regarding the accuracy of the images classified on each architecture, the FPGA performs the worst at 60% classification accuracy on ten images, which can be attributed to the IP designer's optimisations (e.g., pruning) on the CNN architecture to trade-off speed for accuracy.

The figures for ResNet50 indicate that U50 FPGA implementation does outperform the CPU, GPU and XavierNX in runtime, with a 1.567x speed up. However, the ResNet50 runtime results for CPU and GPU are much closer to the FPGA than the MobileNetV2 results. The GPU consumed the highest amount of energy during inference for ten images. However, for one image, the energy consumption was much less than the CPU and closer to the FPGA, which may be due to the GPU consuming more energy as the image buffer is saturated. The XavierNX continues to consume less energy than the other architectures with a 3.38x energy improvement but at the cost of runtime.

Best case improvements when using accelerators

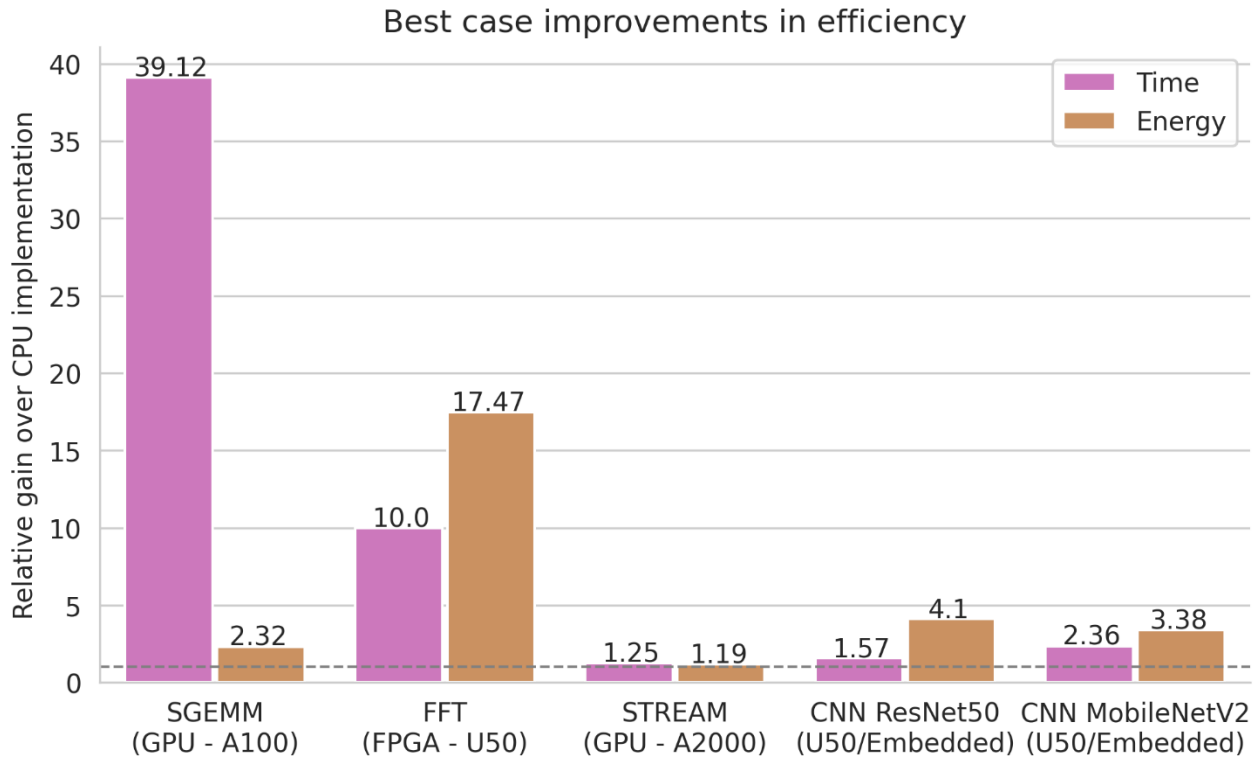


Figure 10: Relative time and energy gains (wrt host CPU) for each benchmark's largest test data runs. Note the host CPU varies.

Figure 10 shows the best-case improvement in time and energy efficiency relative to the relevant CPU implementation and the hardware used to achieve that improvement where the dashed line is at a relative gain of 1.0x. Where there are two problem sizes in the benchmark the largest has been chosen. In both the CNN benchmarks the U50 FPGA provided the greatest speedup while the embedded device, the Jetson XavierNX provided the greatest improvement in energy efficiency. The overall greatest improvement in energy efficiency was found when using the FPGA to calculate a 2D FFT, where we also found a significant improvement in runtime, showing the potential for FPGAs to be both time and energy efficient when programmed well. GPUs are well known to be efficient at dense matrix algebra and our findings reinforce that; we found a nearly 40x improvement in the runtime while improving the energy efficiency by over 2x. Of interest is the performance of the embedded platform which performed significantly worse in terms of runtime but provided excellent energy efficiency compared to other platforms.

We observed that the performance (time and energy to solution) on the FPGA was highly sensitive to the configuration/optimisation of the available port. For example, by default the linker builds a single hardware instance from a kernel. However, if the host program executes the same kernel multiple times (e.g. in the case of GEMM) then the linker forces the kernel to be executed sequentially. This impacts overall application performance. However, configuring the kernel linking stage to instantiate multiple hardware compute units

(CUs) from a single kernel would execute kernels concurrently by running separate compute units, which in turn increases the performance. Additionally, optimising kernel-to-memory connectivity was essential to maximising bandwidth and data transfers. Furthermore, assigning kernels or CUs to specific super logic regions (SLR) (important when assigning kernel ports to specific memory banks) improves placement and timing results. Even if there is only one compute unit in the device, mapping its input and output arguments to different global memory banks can improve performance by enabling simultaneous accesses to input and output data.

Thus, whilst there are energy savings observed by use of FPGA compared to either CPU or GPU, we have reason to expect further savings are available by deep(er) optimisation of the FPGA ports for the given FPGA cards used. We also suggest that CPU and GPU ports will have had time and community engagement to more fully optimise than is the case for FPGAs, hence their observed performances would be closer to the optimal possible.

7. Discussion & Conclusions

From analysis of the selected benchmarks we observed there is always, for the largest data sizes used, an energy gain to be achieved when using an accelerator rather than CPU, although the gains & choice of accelerator vary. Given there is scope for accelerators to significantly reduce the energy to solution, and that (for FPGA technology in particular) it is necessary to expend effort (& expertise) on tuning the port, we recommend UKRI consider a core team of trained experts to support and train researchers.

In terms of initial project objectives: we have shown a methodology for measuring energy for each component of a heterogeneous system comprising CPUs, GPUs and FPGAs (Sections 3 & 5) and we evaluated a range of benchmarks for use in comparing heterogeneous implementations (Section 6). The original list of benchmark codes to implement was about a dozen codes but in practice many of the FPGA ports were not runnable “out of the box” on our cards. The short period of this project excluded manual porting (or optimising) and thus the final set of benchmarks implemented (Table 2) was reduced. Therefore we have not been able to map benchmarks examined to typical loads within UKRI DRI in order to calculate energy savings possible by switching from running applications on CPUs to running on the most appropriate hardware.

Beyond the scope of the current project, further fine grained analysis could be done which includes profiling of the algorithms and components with any given algorithm. This would result into finding the fine grained suitability of the algorithms and its component to various target hardware. Additionally, further optimisation can improve the outcome through efficient native implementation as well as applying parallelism (especially on FPGAs), e.g., pipeline parallelism, task parallelism or data parallelism.

While our benchmarks shows that FPGAs are energy efficient, it is to be noted that FPGAs requires long synthesis time. However, availability of software emulators makes this problem less daunting and it is a worthy investment given key codes would be run millions of times once ported (and the comparative efforts already invested in optimising CPU/GPU implementations).

Further we note that not all applications will be applicable to data flow approach that is optimal for FPGA implementation. Unfortunately, there was insufficient time within the ENERGETIC project to examine a large range of benchmarks to give fine grained recommendations on which UKRI applications would be most suitable for porting to FPGA in terms of energy reductions.

The project, both in terms of our implementations but also via the national workshop (Section 4), has identified various challenges to energy monitoring at the application level. Uppermost, are the frequent need for system administrations to intervene to permit energy data to be accessible to users and the lack of a single interface providing easy access for energy usages across the components of a heterogeneous system. The project has a small number of Recommendations as seen in Section 9.

Our results, and other similar benchmarking efforts, suggest that, as long as a problem is parallelisable and sufficiently large to take proper advantage of a device, codes looking to minimise energy use (and thus carbon emissions) should target accelerators over CPUs. Given that most accelerated HPC systems contain only GPUs, and that GPUs are significantly simpler to program using current development stacks, it is ENERGETIC's recommendation that UKRI focus on enabling acceleration of HPC codes through GPU development. However, given the strong potential of FPGAs to be more energy efficient, and the rapid improvement of FPGA programming tools, training and resources, we also recommend UKRI continue to explore the technology as a potential future component of energy efficient HPC systems.

8. Impact of ENERGETIC Project

- Bane and Quinn each gave a presentation at the HPC-AI Advisory Council UK workshop, with Quinn also being a panel member. https://www.hpcadvisorycouncil.com/events/2022/uk-conference/agenda.php
- Bhowmik and Ali gave a plenary presentation at Computing Insight UK conference. Brown and Bane, supported by all ENERGETIC project members, ran a ½ day workshop at this conference [CIUK]
- (clean version of) all data is available (under GNU GPL licence) at MMU RDS "e-space" [e-space]
- the project's Final Report is available at e-space and the ENERGETIC project web page: https://ukri-netzero-energetic.github.io/
- original project proposal: 10.5281/zenodo.6787467

9. Recommendations

The ENERGETIC project recommends that:

- UKRI should invest in accelerator hardware and associated training & development in order to reduce energy to solution of HPC workloads. The accelerators should include GPU and FPGA technologies and further benchmarking efforts funded to improve understanding of differences in energy efficiency across different classes of algorithms.
- UKRI compute-focused DRI administrators endeavour to provide a common interface or framework to abstract underlying monitoring mechanisms and expose energy consumption data to users.
- UKRI commits to requiring user-level energy monitoring in a specified format to be available on all UKRI-funded DRI, and incorporates this into the funding and procurement processes.
- UKRI invests in raising awareness of the need to optimise for least energy consumption
- UKRI invests in upskilling researchers regarding (env.) sustainable software principles & best practises, via Green Software Engineers (GSEs) and encouraging researchers and system admin to undertake Green Software training (e.g. https://trainingportal.linuxfoundation.org/learn/course/green-software-for-practitioners-lfc131)

10. Author Contact Details

Dr. Michael K. Bane: Dept. of Computing & Mathematics, Manchester Metropolitan University (MMU), Oxford Road, Manchester. Email: m.bane@mmu.ac.uk https://helward.mmu.ac.uk/STAFF/M.Bane/

11. References

[A2DOCS] EPCC, ARCHER2 Documentation – Energy use, https://docs.archer2.ac.uk/user-guide/energy/, (2022), accessed 03/01/2023.

[Asanovic et al] Asanovic, K., Bodik, R., Catanzaro, B.C., Gebis, J.J., Husbands, P., Keutzer, K., Patterson, D.A., Plishker, W.L., Shalf, J., Williams, S.W. and Yelick, K.A., 2006. The landscape of parallel computing research: A view from Berkeley.

[Baskerville] https://www.baskerville.ac.uk/ accessed 16/01/2023

[Cirrus] Cirrus. https://www.epcc.ed.ac.uk/hpc-services/cirrus accessed 16/01/2023

[CIUK] STFC Scientific Computing, CIUK 2022 Breakout Sessions, https://www.scd.stfc.ac.uk/Pages/CIUK-2022-Breakout-Sessions.aspx, (2022), accessed 09/01/2023.

[CPUID] HWMonitor. https://www.cpuid.com/softwares/hwmonitor.html accessed 23/01/2023

[cublas]: https://github.com/hma02/cublasgemm-benchmark accessed 23/01/2023

[cuFFT]: https://github.com/NVIDIA/CUDALibrarySamples/tree/master/cuFFT accessed 23/01/2023

[cuda-stream]: https://github.com/bcumming/cuda-stream accessed 23/01/2023

[EEHPCWG] https://eehpcwg.llnl.gov/ accessed 23/01/2023

[EHSD4.2] Carpen-Amarie et al., D4.2 Design Document of GPI-Space Extension for Distributed FPGAs and DSL for Deep-Learning Applications, EPiGRAM-HS (2019). https://cordis.europa.eu/project/id/801039/results accessed 23/01/2023

[Energetic] https://github.com/ukri-netzero-energetic/benchmarks-and-scripts accessed 16/01/2023

[e-space] e-Space MMU Research Repository. ENERGETIC Project. doi.org/10.23634/MMU.00631226 accessed 24/01/2023

[FFT]: https://weww.intel.com/content/www/us/en/develop/documentation/get-started-with-mkl-for-dpcpp/top.html accessed 23/01/2023

[GEOPM] GEOPM. https://geopm.github.io/ accessed 23/01/2023

[GSF] Green Software Foundation, Software Carbon Intensity (SCI) Specification. https://github.com/Green-Software-Foundation/software_carbon_intensity/blob/main/Software_Carbon_Intensity/Software_Carbon_Intensity_Specification.md accessed 23/01/2023

[hpcc] https://hpcchallenge.org/hpcc/ accessed 23/01/2023

[hpcc-fpga] https://github.com/pc2/HPCC_FPGA/tree/60651eb40fa524218e661f7048e989ab2a106b58/STREAM

[InsideHPC] First Global Survey of Energy and Power Aware Job Scheduling and Resource Management (20 Dec 2017). https://insidehpc.com/2017/12/first-global-survey-energy-power-aware-job-scheduling-resource-management/ accessed 23/01/2023

[Intel] Intel 64 and IA-32 Architectures Software Developer's Manual, Volume 3 ("System Programming Guide"), Chapter 14. https://cdrdv2.intel.com/v1/dl/getContent/671200 accessed 23/01/2023

[Intel-gemm] Intel. Developer Reference. ?gemm. https://www.intel.com/content/www/us/en/develop/documentation/onemkl-developer-reference-fortran/top/blas-and-sparse-blas-routines/blas-routines/blas-level-3-routines/gemm.html accessed 23/01/2023

[JADE2] JADE II. https://docs.hpc.shef.ac.uk/en/latest/other-uk-hpc-resources/jade2.html accessed 16/01/2023

[Kaggle] Kaggle Inc., Kaggle, https://www.kaggle.com/, (2022), accessed 15/12/2022.

[KillAWatt] http://www.reuk.co.uk/Buy-UK-Power-Meter.htm accessed 22/02/2023

[Laros et al] Laros, J. H., Grant, R., Levenhagen, M. J., Olivier, S. L., Pedretti, K., Ward, H. L., & Younge, A. J. (2017). High Performance Computing-Power Application Programming Interface Specification Version 2.0 (No. SAND-2017-2684). Sandia National Lab.(SNL-NM), Albuquerque, NM (United States). https://www.osti.gov/biblio/1347187

[LIKWID] NHR, Likwid Performance Tools, https://hpc.fau.de/research/tools/likwid/, (2022), accessed 03/01/2023.[LIKWID] NHR, Likwid Performance Tools, https://hpc.fau.de/research/tools/likwid/, (2022), accessed 03/01/2023.

[MB3D6.9] Mantovani et al., MB3 – D6.9: Final report of applications on the project test platform: performance evaluation and optimizations, Mont-Blanc 3 (2019), https://cordis.europa.eu/project/id/779877/results accessed 23/01/2023

[Myriad] https://www.rc.ucl.ac.uk/docs/Clusters/Myriad/ accessed 16/01/2023

[Open Hardware Monitor] https://openhardwaremonitor.org/ accessed 23/01/2023

[PAPI] Jagode, H., YarKhan, A., Danalis, A., Dongarra, J. Power management and event verification in PAPI. In: Knüpfer, A., Hilbrich, T., Niethammer, C., Gracia, J., Nagel, W.E., Resch, M.M. (eds.) Tools for High Performance Computing 2015, pp. 41–51. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-39589-0_4 accessed 23/01/2023

[PowerAPI] https://pwrapi.github.io/ accessed 23/01/2023

[PowerStack] https://hpcpowerstack.github.io/ accessed 23/01/2023

[pytorch]: https://pytorch.org/vision/stable/models.html accessed 23/01/2023

[PytorchC] https://pytorch.org/docs/stable/notes/cuda.html accessed 23/01/2023

[Schone] Schöne, R., Ilsche, T., Bielert, M., Velten, M., Schmidl, M. and Hackenberg, D. Energy Efficiency Aspects of the AMD Zen 2 Architecture. https://arxiv.org/pdf/2108.00808.pdf accessed 23/01/2023

[stream]: https://www.cs.virginia.edu/stream/ accessed 23/01/2023

[Sulis] https://warwick.ac.uk/research/rtp/sc/sulis/about accessed 16/01/2023

[TOP500] Top500, Top 500 List – November 2022, https://www.top500.org/lists/top500/list/2022/11/, (2022), accessed 14/12/2022.

[UKplug] https://www.amazon.co.uk/230V-250V-Monitor-Consumption-Calculator-Analyzer/dp/B07FZZ17ZY accessed 22/01/2023

[Variorium] https://variorum.readthedocs.io/en/latest/index.html accessed 23/01/2023

[vitis-ai]: https://github.com/Xilinx/Vitis-AI-Tutorials/blob/1.4/Design_Tutorials/09-mnist_pyt/README.md accessed 23/01/2023

[vitis_fft]: https://github.com/Xilinx/Vitis_Libraries/tree/main/dsp/L1/tests/hw/2dfft accessed 23/01/2023

[vitis_gemm]: https://github.com/Xilinx/Vitis_Libraries/tree/main/blas/L3/benchmarks/gemm accessed 23/01/2023

Appendix 1: Box-whiskers plots supporting Section 5 Discussion

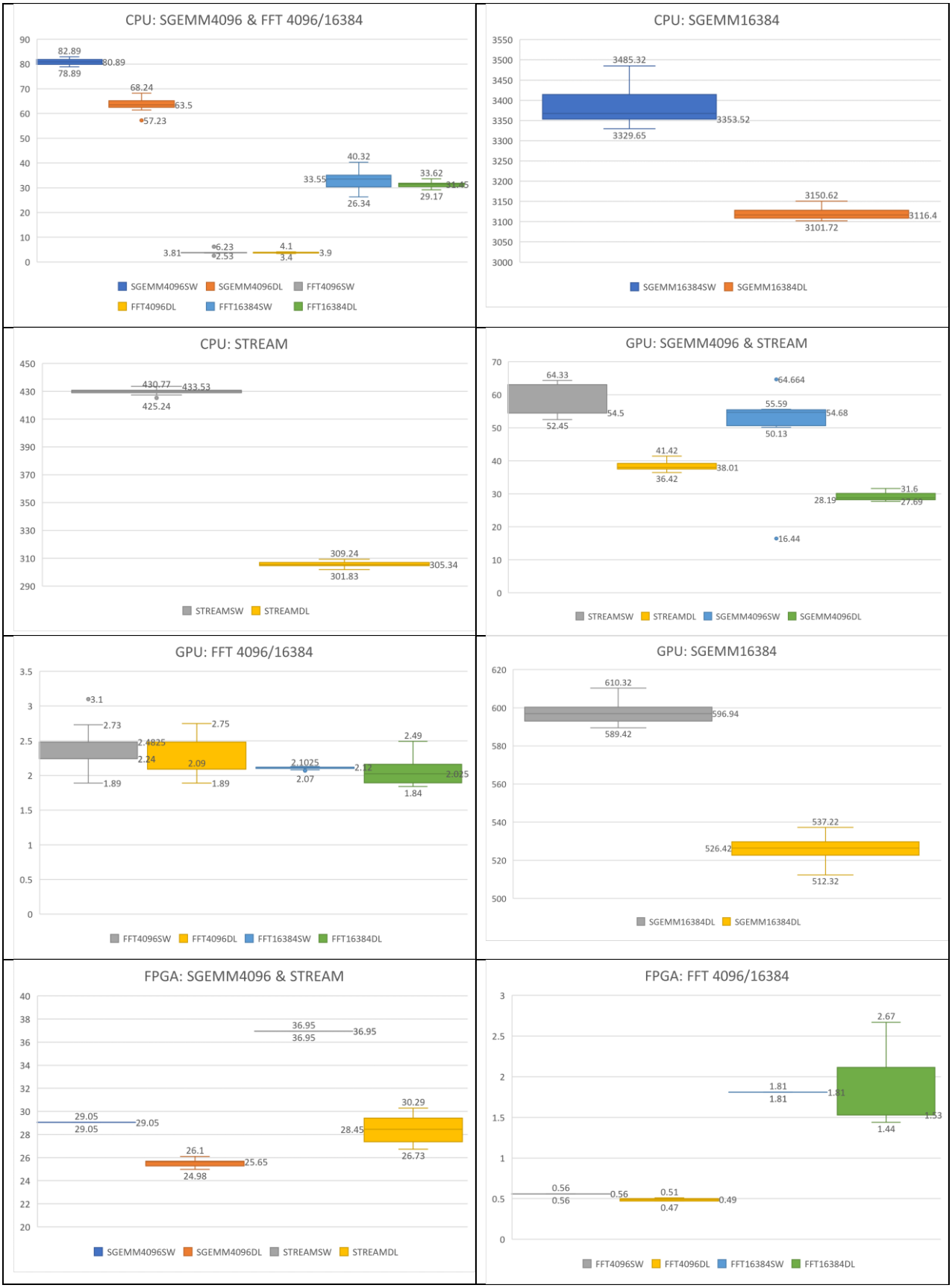


Figure 11: Box-whisker plots of energy consumed for various benchmarks for software ("SW") and datalogger ("DL") approaches as discussed in Section 5..

Appendix 2: Data supporting Section 6 Figures & Discussion

CNN MobileNetV2

Architecture	Num. Images	Runtime	Total Energy	CPU Energy	Acc Energy
CPU (i9)	1	0.18	78.70	78.70	0.00
GPU (A2000)	1	0.09	69.58	68.00	1.58
Embedded (XavierNX)	1	0.53	7.95	0.00	7.95
FPGA (U50)	1	0.06	66.18	65.00	1.18
CPU (i9)	10	2.10	205.57	205.57	0.00
GPU (A2000)	10	1.10	189.97	164.67	25.30
Embedded (XavierNX)	10	3.34	50.13	0.00	50.13
FPGA (U50)	10	0.89	184.37	168.35	16.02

CNN ResNet50

Architecture	Num. Images	Runtime	Total Energy	CPU Energy	Acc Energy
CPU (i9)	1	0.31	80.91	0.00	80.91
Embedded (XavierNX)	1	0.82	8.20	0.00	8.20
GPU (A2000)	1	0.26	27.13	5.03	22.10
FPGA (U50)	1	0.19	25.53	3.97	21.56
CPU (i9)	10	3.20	214.41	0.00	214.41
GPU (A2000)	10	2.34	278.61	49.61	229.00
Embedded (XavierNX)	10	4.23	63.45	0.00	63.45
FPGA (U50)	10	2.04	227.40	42.84	184.56

STREAM

Architecture	Runtime	Total Energy	CPU Energy	Acc Energy
CPU (Xeon)	1.77	173.84	0.00	173.84
GPU (A2000)	1.42	145.41	54.50	90.91
FPGA (U50)	1.54	137.95	27.71	110.24

SGEMM

Architecture	Resolution	Runtime	Total Energy	CPU Energy	Acc Energy
CPU (Xeon)	4096	0.78	63.68	0.00	63.68
GPU (A2000)	4096	1.24	140.27	55.45	84.82
FPGA (U50)	4096	1.71	92.02	29.05	62.97
FPGA (U280)	4096	0.97	24.20	16.20	8.00
CPU (Xeon)	16384	19.56	2361.36	0.00	2361.36
GPU (A2000)	16384	12.27	1387.66	596.94	790.72
GPU (A100)	16384	0.50	1018.31	685.67	332.64
FPGA (U280)	16384	58.15	1446.32	1014.85	431.47

FFT

Architecture	Resolution	Runtime	Total Energy	CPU Energy	Acc Energy
CPU (i9)	4096	0.28	3.81	3.81	0.00
GPU (A2000)	4096	0.11	14.50	12.26	2.24
FPGA (U50)	4096	0.03	0.66	0.10	0.57
CPU (i9)	16384	1.00	33.55	33.55	0.00
GPU (A2000)	16384	0.12	7.89	5.77	2.12
FPGA (U50)	16384	0.10	1.92	0.11	1.81