PowerSensor3: A Fast and Accurate Open Source Power Measurement Tool

Steven van der Vlugt[1], Leon Oostrum[2], Gijs Schoonderbeek[1], Ben van Werkhoven[3,2], Bram Veenboer[1], Krijn Doekemeijer[4], John W. Romein[1]
[1]ASTRON (Netherlands Institute for Radio Astronomy), Dwingeloo, the Netherlands
{vlugt, schoonderbeek, veenboer, romein}@astron.nl
[2]Netherlands eScience Center, Amsterdam, the Netherlands, l.oostrum@esciencecenter.nl
[3]Leiden University, Leiden, the Netherlands, b.van.werkhoven@liacs.leidenuniv.nl
[4]Vrije Universiteit Amsterdam, Amsterdam, the Netherlands, k.doekemeijer@vu.nl

***Abstract*—Power consumption is a major concern in data centers and HPC applications, with GPUs typically accounting for more than half of system power usage. While accurate power measurement tools are crucial for optimizing the energy efficiency of (GPU) applications, both built-in power sensors as well as state-of-the-art power meters often lack the accuracy and temporal granularity needed, or are impractical to use. Released as open hardware, firmware, and software, PowerSensor3 provides a cost-effective solution for evaluating energy efficiency, enabling advancements in sustainable computing. The toolkit consists of a baseboard with a variety of sensor modules accompanied by host libraries with C++ and Python bindings. PowerSensor3 enables real-time power measurements of SoC boards and PCIe cards, including GPUs, FPGAs, NICs, SSDs, and domain-specific AI and ML accelerators. Additionally, it provides significant improvements over previous tools, such as a robust and modular design, current sensors resistant to external interference, simplified calibration, and a sampling rate up to 20 kHz, which is essential to identify GPU behavior at high temporal granularity. This work describes the toolkit design, evaluates its performance characteristics, and shows several use cases (GPUs, NVIDIA Jetson AGX Orin, and SSD), demonstrating PowerSensor3's potential to significantly enhance energy efficiency in modern computing environments.**

***Index Terms*—**

I. Introduction

Power consumption is among the largest expenses in data centers and is estimated at 1-1.5% of global electricity use [1]. The recent surge in training Large Language Models (LLMs), consuming around 29.3 terawatt-hours per year—equivalent to Ireland's energy consumption [2]—has prompted companies like Amazon, Google, and Microsoft to invest billions in nuclear energy [3] to meet this demand. The Frontier supercomputer, the world's first exascale supercomputer, consumes 22.7 MW continuously [4]. Supercomputing was found to be responsible for 59.8% of the carbon emissions of the average astronomer in Australia, 3.6 times as much as air travel [5]. As less than 15% of the world's energy comes from renewable sources [6], it is crucial that we investigate how to improve the energy efficiency of these systems and applications that run on them, and reduce our carbon footprint.

Over the past decade, large improvements in the energy efficiency of data center cooling and power provisioning have been significant enough to nearly offset the growth of IT device energy use [1, 7]. As such, the crucial next step is to understand and improve energy expenditure within computer systems. Modern systems rely on many peripheral devices, including network interface controllers (NICs) and solid state drives (SSDs) that all require power. Among the peripheral devices, Graphics Processing Units (GPUs) stand out as the primary computing platform for nearly all large-scale AI and HPC applications [8, 9], delivering 99% of the compute performance in modern supercomputers [10], and consuming >64% of the total power of these systems [11]. To advance energy efficiency research, it is critically important to develop fast, accurate, and openly accessible methods for measuring the power consumption of computer components.

Many software-based methods have been developed to effectively reduce energy consumption in computing systems. For example, dynamic voltage and frequency scaling (DVFS) [12–15], power-aware scheduling [16, 17], power capping [18], and energy-efficient algorithm design [19–22] can significantly lower power usage without compromising performance. For instance, power-aware scheduling algorithms can optimally divide work between CPU and GPU depending on specific task properties [16, 23, 24], or generate optimized schedules for GPU kernels executing concurrently [17]. Additionally, code-level optimizations, such as compiler [25, 26] and function-level [11, 22] tuning, can lead to substantial energy savings. However, implementing these methods effectively requires fast (sub-millisecond) and accurate power measurements at a fine-grained level, for example down to single GPU kernels or even during the execution of individual operations. Without precise measurement tools, it is challenging to assess the impact of optimizations and guide further improvements in energy efficiency.

This paper presents PowerSensor3, a tool that measures the instantaneous power consumption of SoC development boards and PCIe cards like GPUs, FPGAs, domain-specific

This work received funding from the European Union through the RADIOBLOCKS (101093934) and MCSA-RISE Cloudstars (101086248) projects, the Dutch Research Council (NWO) through the DAS-6 (621.018.201), CORTEX (NWA.1160.18.316), OffSense (OCENW.KLEIN.209) and MLS (OCENW.KLEIN.561) grants, and from the Netherlands eScience Center through the RECRUIT (ETEC.2020.025) grant. K. Doekemeijer is funded by the VU PhD innovation program.

accelerators for AI and ML, and NICs, at 20 kHz (sub-millisecond) time scale. PowerSensor3 includes several important improvements over PowerSensor2 [27] and other non-commercial and commercially available power measurement tools, including:

- A modular design with a base board that supports up to 4 sensor board modules.
- A variety of sensor boards, with different connectors and sensors (e.g., 8-pin PCIe power, USB-C, high-current and low-current boards with terminal blocks).
- Support for measuring both voltages and currents.
- An increased sampling rate from 2.8 kHz to 20 kHz through the use of a faster microcontroller.
- The use of current sensors that are hardly sensitive to changes of the external magnetic field.
- Simplified, one-time calibration procedure through a command-line utility.
- The base board and sensor boards are released as open hardware [28] (CERN-OHL-P v2) and the firmware and host library are released as open-source software [29] (Apache-2.0).
- Cost-efficient design, a complete PowerSensor3 with 3 sensor boards costs less than € 100 in components.

This paper is structured as follows. Section II provides background and discusses related work. Section III describes the PowerSensor3 design and implementation. Section IV characterizes its performance, and in Section V, we describe some application use cases. Finally, Section VI discusses the application and extendibility and Section VII concludes.

II. Background and Related Work

Various methodologies exist for measuring power consumption within computer systems. This section provides a comprehensive overview of the power measurement tools that have been utilized and discussed in the scientific literature.

Several researchers have used commercially available tools to measure whole system power, seeking to improve the energy efficiency of GPU applications. However, these tools generally have very low sampling rates, for example the Watts Up Pro operates at 1 Hz [12, 13, 20, 30], Cray PMDB at 10 Hz [31], or Yokogawa WT230 at 10 Hz [32, 33]. While measuring whole system power might give a realistic view of power consumption for the whole application, measuring power consumption of components in isolation can give insights needed to improve efficiency in critical parts of the application.

For CPUs, several software packages exist to monitor power consumption by reading from built-in sensors. Intel's Running Average Power Limit (RAPL) provides a number of performance counters to read energy consumption of CPUs and DRAM, with a time frequency of 1 kHz [34]. LIKWID also provides the likwid-powermeter tool, which builds on top of RAPL and allows to measure power consumption on architectures from Intel, AMD, ARM, and IBM.

However, many PCIe devices such as SSDs, domain specific accelerators or NICs do not have built-in power measurement tools, and thus require external power measurement. And as such researchers have been developing their own custom-built power measurement devices for measuring power of such components [27]. One challenge with power measurement of PCIe devices is that PCIe devices receive power from different sources. Up to 75 W of power could be delivered via the PCIe slot, 10 W of which via the 3.3 V rail, the rest at 12 V. Devices that need more than 75 W can receive additional power at 12 V from the power supply unit through, possibly multiple, 6-pin, 8-pin or 12-pin PCIe connectors, or from the host motherboard using the 8-pin EPS connector. As such, measuring the power consumption of PCIe devices requires measuring current across multiple power cables. Moreover, voltages cannot be assumed to be stable under load, therefore the voltage needs to be measured for every power cable as well.

A. GPU on-board power measurement

NVIDIA has been shipping an internal power sensor in both server-grade and consumer-grade GPUs as part of the Kepler architecture [35], starting with the NVIDIA Tesla K20. While the properties of NVIDIA's current sensor have been studied widely [36–38], only a few studies have used AMD's built-in current sensor and reported on its accuracy and sampling frequency. Wu et al. [39] reported that the AMD Radeon HD 7970 estimates the chip-wide dynamic power and updates the power estimates every millisecond. Schieffer et al. [40] report to have used a sampling frequency as low as 10 ms on the AMD MI250X GPU, but did not state the lower bound.

An evident advantage of utilizing these built-in sensors is their widespread availability. However, there are two significant drawbacks. Firstly, a persistent issue with vendor-based APIs for reading the GPU internal power sensor is that they typically return only averaged power consumption values [37]. Notably, NVIDIA has addressed this with driver update 530 (May 2023), which extends their API to support instantaneous power readings [38]. Secondly, the use of on-board power measurement in GPUs is hindered by low sampling frequencies. Even with the capability for instantaneous power readings, new values are provided at a frequency of approximately 10 Hz on NVIDIA GPUs [38].

The issues with on-board power measurement in GPUs have forced researchers in GPU energy efficiency to artificially increase the execution time of their GPU kernels by several orders of magnitude to allow for enough samples to be collected and to overcome the effects of averaged power readings and low sampling rates [14, 18, 22, 41]. There are several downsides to such approaches, as evaluating power consumption in different settings or for different software implementations consumes large amounts of time and energy, and the measured execution itself can be less realistic compared to real execution scenarios.

B. External power measurement tools

Several researchers have used current clamps to determine GPU power consumption [16, 42, 43] without documenting the achieved accuracy and sampling rate. Timm et al. [44] also

used a current clamp to determine GPU power consumption and mentioned a sampling rate of 10 kHz.

PowerMon2 [45] is a custom-built power monitoring device for voltage and current measurements with a 1 kHz sampling frequency and a relatively low, -6.6% / +6.8%, current measurement accuracy. It also uses a difficult to obtain custom implementation and cannot handle 150 W PCIe power cables. Another tool, PowerInsight [46], measures both voltages and currents, but has a sample rate of less than 1 kHz. Consequently, it cannot capture the detail required to conduct precise PCIe power measurements. The exact sample rate is not given in the paper. Finally, NVIDIA has also produced the power measurement device Power Capture Analysis Tool (PCAT), which is not for sale nor is its design documented. The PCAT documentation suggests a sampling rate of 10 Hz[1].

In addition to research-oriented PCIe power sensors, several commercial options are available. Most prominent are PMD ($ 60) and Powenetics V2 (€ 975). The Powenetics V2 is expensive and has a sampling rate of up to 1 kHz according to their website. PMD was recently used by Yang et al. [38] to perform a large study of the accuracy of power measurements reported by NVML on over 70 different GPUs. They mention that while PMD has an internal sampling frequency of 34 kHz, PMD's (Windows-only) host library limits updates to a sampling frequency of 10 Hz. Yang et al. [38] developed their own data logger to achieve a sampling frequency of 5 kHz. To the best of our knowledge, their data logger is not openly available.

III. Design and Implementation

To measure power for modern PCIe accelerators, it is essential to monitor both PCIe slot power (3.3 V and 12 V, up to 75 W) and up to two external power connections (up to 600 W) for PCIe gen 5 and 6. Accurate measurement requires monitoring voltage and current for each power supply with isolation to prevent coupling between the device and the sensors. Modern accelerators, like GPUs, need fast sensors due to their high processing speeds and low kernel execution times. In order to reduce power loss over cables, the sensor must be close to the accelerator, installed within the server. This imposes size and safety constraints, requiring stable connectors and ensuring no contact with other components in a noisy environment. The PowerSensor3 is designed to meet the needs of PCIe gen 4, 5, and 6 accelerators. Its flexible design makes it suitable for high-power PCIe devices (e.g., GPUs) as well as lower-power or standalone devices such as SoC boards.

A. Hardware Design

The core of the PowerSensor3 system is a baseboard that accommodates the "Black Pill" STM32F411 microcontroller module and up to four sensor modules. The STM32F411 [47] was selected due to its ability to sample up to sixteen analog inputs, enabling the support of four sensor modules, its USB data transfer capabilities, and the availability of software development tools. To cater to different power ranges and connectivity, several different sensor modules are developed. The designs are open-source and can be modified to meet a specific power range, accuracy or connector type. This modular approach allows users to select the most appropriate power monitoring sensors for their specific applications. Additionally, a small display is integrated into the baseboard to show instantaneous power consumption.

Fig. 1 illustrates an example of the PowerSensor3 in operation. In this example, the PowerSensor3 is equipped with a PCIe sensor module that measures the power supplied to the PCIe card via the external power input as well as two sensor modules to measure the 3.3 V and 12 V PCIe slot power. By utilizing a modified riser card, where the power connections for both 3.3 V and 12 V are interrupted and routed through two sensor modules, it is possible to measure the power consumption of both the slot and the external connection without affecting the PCIe signal integrity.

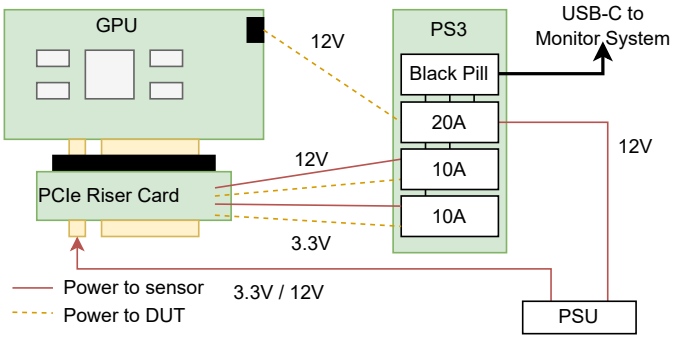


Fig. 1: Schematic of a PowerSensor3 measurement setup.

Each sensor module measures both voltage and current. To mitigate the effect of ground loops on the measurements, the circuit connected to the Device Under Test (DUT) is isolated from the measurement logic connected to the microcontroller. This isolation is achieved using a differential Hall sensor, the Melexis MLX91221 [48], for current measurement, and an optically isolated voltage sensor, Broadcom ACPL-C87B [49], for voltage measurement. The Hall sensor family supports a variety of pin compatible devices with different current ranges.

The resistance loss in both the power and return paths of the power sensor, which can cause measurement inaccuracies and affect the power delivered to the DUT, is a critical design parameter. To minimize resistance, the sensor is designed to be compact. To reduce the impact of voltage loss within the sensor and the connecting wire to the DUT, a remote sense connector is integrated into the sensor module. This allows for the measurement of voltage directly at the DUT rather than at the input port.

PowerSensor3 currently comes with five different designs for sensor modules:

- 20 A PCIe 8-pin: With a connector for easy integration with the external power connector on PCIe cards.
- 10 A: Designed to measure power between the PCIe slot and the PCIe card.

[1]https://developer.nvidia.com/nvidia-power-capture-analysis-tool

- USB-C: Suitable for USB-powered systems.
- 20 A: General-purpose power measurement for medium power applications, with terminal block connectors.
- 50 A High-Current: For high-power applications.

These sensor modules can be combined in various configurations within a single setup, providing a comprehensive and adaptable power measurement solution. Fig. 2 shows a 3D rendering of a populated PowerSensor3 module. The design of the baseboard and sensor modules has been made available [28].

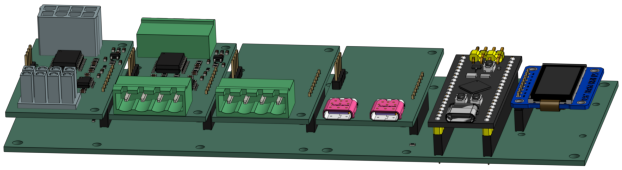


Fig. 2: 3D rendering of PowerSensor3 with PCIe 8-pin, 20 A, 10 A, USB-C sensor modules, "Black Pill" module and display.

The analog signals from the sensor modules are passed to the STM32F411, where the 10 most significant bits of the Analog to Digital Converter (ADC) are utilized. The bandwidth of the current and voltage sensors is well above the ADC's output sample rate of 20 kSamples/s. The maximum time resolution of the current sensor is specified at 300 kHz, while the maximum time resolution of the voltage sensor is 100 kHz.

The measured power is calculated with:

$$P = (U + E_u) \cdot (I + E_i)$$

where $U + E_u$ is the combination of the voltage and the error in the voltage reading, and $I + E_i$ is the combination of the current and the error in the current reading. This gives the error in the power reading as:

$$E_p = \sqrt{(U \cdot E_i)^2 + (I \cdot E_u)^2 + (E_i \cdot E_u)^2}$$

This formula shows that for small loads, the noise is dominated by the error in the current reading, while for low voltage high current sensors, the noise is dominated by the error in the voltage reading.

The error in the current reading (E_i) consists of the error due to the quantization noise of the ADC, combined with the sensitivity of the Hall sensor and inherent noise of the Hall sensor. Among these factors, the noise in the Hall sensor, which is 115 mArms for the 10 A sensor, is the dominant factor, resulting in a peak-to-peak error of 4.1 Wpp.

The error in the voltage reading (E_u) is caused by quantization noise and the inherent amplifier noise. The noise on the voltage reading is increased due to the voltage divider. For a 12 V / 10 A sensor module, the noise on the voltage at high currents is estimated to be 0.2 Wpp.

Table I provides an overview of the theoretical worst case accuracy of the sensor modules.

TABLE I: Theoretical worst case accuracy of PowerSensor3 modules.

Module	Voltage	Current	Power
12 V / 10 A	± 28.6 mV	± 0.35 A	± 4.2 W
3.3 V / 10 A	± 19.9 mV	± 0.35 A	± 1.2 W
USB-C (20 V / 10 A)	± 28.6 mV	± 0.35 A	± 7.0 W
Ext (12 V / 20 A)	± 28.6 mV	± 0.41 A	± 5.0 W

B. Firmware Design

This section details the firmware design using the STM32F411 microcontroller. Its primary function is to read current and voltage sensors at a high, constant rate and transmit the data to the host via USB. The ADC continuously reads the sensors, and the DMA controller transfers the values to RAM. Once all sensor data is in RAM, an interrupt is generated. The interrupt handler reads the sensor values, adds metadata, and creates a data package for the host. A main loop checks for data to be sent and transmits it as needed.

We use the STM32 low-level library through *STM32duino*[2]. *STM32duino* provides a simple interface, integrating it into the Arduino ecosystem, and as such facilitating firmware development, compilation, and uploading through the widely-adopted Arduino tools.

Ideally, the ADC would operate at the highest possible clock speed, transmitting data directly to the host. However, the data rate would exceed the capacity of the USB controller on the Black Pill, which supports up to USB 1.1 full speed (12 Mbit/s). Although a USB 2.0 controller can be added, we opt to reduce the sampling rate instead, to minimize the cost and complexity of the PowerSensor3 hardware.

Each sensor board contains a sensor pair with current and voltage sensors. Within each pair the sensors are connected to consecutive ADC channels, minimizing the time difference between measurements. The ADC operates at a clock speed of 24 MHz, with the CPU averaging several samples to reduce the final sampling rate. The ADC is configured with a 10-bit resolution and a sampling time of 15 clock cycles. Each bit requires one cycle to read, resulting in a total ADC sampling time of 25 clock cycles or 1.04 μs. Reading 8 sensors (4 modules) and averaging 6 consecutive samples on the CPU amounts to a 50 μs interval, corresponding to a sampling rate of 20 kHz. For each sensor, we transmit 2 bytes of data to the host. With 10 bits per sensor value and 6 bits for metadata: the sensor index, a marker, and one bit in each byte to differentiate the first byte from the second.

The sensor data sent from the PowerSensor3 to the host is preceded by a device timestamp. The timestamp is generated after processing 3 out of the 6 samples to be averaged and is stored as a 10-bit value in microseconds. Since there is no room in the sensor data packets for the timestamp, it is sent separately. To differentiate the timestamp data from sensor data, a combination of the sensor and marker bits is used: a real marker bit can only be set in the sensor data of sensor 0. A marker bit set to one with a nonzero sensor index is unused and can be repurposed for other data. For the timestamp, the maximum sensor index of 7 (binary 111) is used.

[2]https://github.com/stm32duino

The firmware supports several options through the host:

- Start or stop streaming of sensor data.
- Send or receive configuration values (Section III-B1).
- Send a marker with the next sensor data.
- Send the firmware version as a string.
- Reboot the device, optionally to DFU mode which is used for uploading new firmware.

1) Sensor configuration values: The PowerSensor3 firmware is generally agnostic to the type of sensor module used. However, the host software must know how to convert raw sensor values to accurate voltage and current readings. These conversion values are stored on the device and communicated to the host library, so the user does not need to keep track of the specific sensors used. The STM32 supports a virtual EEPROM implementation that stores data in flash memory. The following data is stored for each sensor:

- Sensor name.
- Reference voltage.
- Sensitivity (current sensors) or gain (voltage sensors).
- Sensor state (enabled or disabled).

2) Display: PowerSensor3 is equipped with a compact display for real-time visualization of sensor values when the sensor is not in use by the host system. This display prominently features the total power consumption, while individual current, voltage, and power measurements for each sensor pair are shown in smaller fonts.

The display is connected through an SPI interface, controlled by the open-source Adafruit ST7735 library[3]. To enhance display update speed and reduce CPU load, we expanded the library with two features: 1) we enable DMA for transferring the display buffer from RAM to the SPI controller, and 2) we pre-compute graphics for all necessary characters in all used color and size combinations, storing the resulting fonts in the program memory. A Python script for automatic font generation is included in the PowerSensor3 repository.

C. Software Design

The PowerSensor3 host library is implemented in C++, with an optional Python wrapper[4]. The device is accessed via a PowerSensor C++ class, which, upon initialization, connects to the microcontroller and reads the sensor configuration values. Methods are available to read or set these values. A lightweight thread continuously receives sensor values from the device, and the library internally tracks the cumulative energy consumption measured by each sensor.

PowerSensor3 can operate in two modes: interval-based and continuous, both of which can be active simultaneously. In interval-based mode, the user requests sensor states at two different times to calculate total energy consumption and average power. This mode can be accessed through a standalone executable or via the C++ or Python interface, allowing precise control over the measurement period but requiring source code modification. The standalone executable, *psrun*, connects to PowerSensor3, runs the provided application (executable), and reports the total energy consumed after execution. In continuous mode, PowerSensor3 records all sensor data to a file at 20 kHz resolution. The library supports custom marker characters in the output file, time-synced with the microcontroller, to correlate timestamps with specific parts of the application code.

[3]https://github.com/adafruit/Adafruit-ST7735-Library

[4]implemented using pybind11, https://github.com/pybind/pybind11

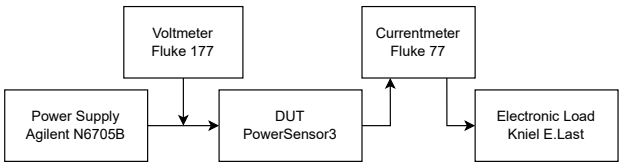


Fig. 3: Measurement setup for accuracy assessment.

The PowerSensor3 host library comes with three additional executables for easier interfacing with the device:

psconfig reads or writes the sensor configuration values and optionally reboots the device. After installing the firmware, this tool is used to configure the device.

psinfo shows the configuration values of each enabled sensor, as well as the latest measurements and the total power.

pstest measures and reports power and energy at increasing intervals for testing purposes.

D. Calibration

The sensor modules are calibrated using a known power supply, such as the system's power supply unit or a laboratory power supply. During calibration, the sensor modules are unloaded (no power dissipation), and the voltages on the voltage sensors are measured. By taking 128 k samples and calculating the average current and voltage readings, the offset error of the Hall current sensor and the gain error for the voltage are determined. These corrections are then stored in the microcontroller. Python scripts are available to guide through this process. Based on the measurements described in Section IV, calibration is only required once at production.

IV. Evaluation

To verify the functionality of the PowerSensor3, the test setup illustrated in Fig. 3 was utilized. A laboratory power supply (Keysight N6705B) served as the power source for the DUT. An electronic load (Kniel E.Last) was employed for loads up to 10 A. The voltage at the sensor and current through the load were measured using a Digital Multimeter (Fluke 177 for the voltages and Fluke 77 for the current). Data was captured using *pstest*.

A. Sensor accuracy assessment

To evaluate the sensor's accuracy, a measurement was conducted where the load current was swept in 1 A steps from the minimum (-10 A) to the maximum current (+10 A). At each step, 128 k samples were collected using the *pstest* tool. This data allowed the determination of the accuracy and variability of the current and voltage readings, and thus the power calculation.

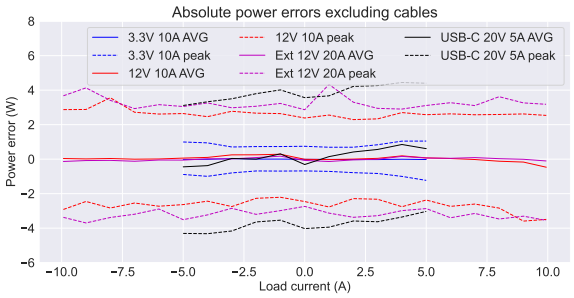


Fig. 4: Power error for four types of sensors with dotted lines indicating min and max values per measurement point.

TABLE II: Overview of error values for different samples rates for 0.5 A and 1 A loads.

F_s	0.5 A load				1 A load			
	min	max	p-p	std	min	max	p-p	std
kHz	W	W	W_{pp}	W_{rms}	W	W	W_{pp}	W_{rms}
20	2.78	9.16	6.381	0.718	7.79	15.48	7.685	0.722
10	4.04	8.22	4.173	0.507	9.42	14.53	5.109	0.511
5	4.85	7.69	2.842	0.358	10.54	13.68	3.142	0.362
1	5.66	6.85	1.183	0.16	11.62	12.9	1.285	0.163
0.5	5.85	6.67	0.821	0.113	11.92	12.73	0.814	0.117

In Fig. 4, the results are shown. The continuous line indicates the difference between the expected power and the measured power. The dotted lines in this figure represent the minimum and maximum difference within the 128 k samples at each measurement point. As can be seen in this figure, the accuracy of the 3.3 V sensor is better in comparison with the 12 V sensor, where the error in the current sensor is multiplied by 12 instead of 3.3.

Detailed inspection of the data indicates that at low currents, noise originates primarily from the current sensor, while at higher currents, the voltage sensor noise becomes more significant. Averaging the samples can reduce the noise but also lowers the time resolution (F_s). Table II provides an error overview for a 12 V, 10 A sensor with an 8 A load, where blocks of samples are averaged. In the table, the minimum and maximum values after averaging, the peak-to-peak range between these two values, and the standard deviation are shown.

B. Long term stability

The long-term stability of PCIe 8-pin sensor modules was assessed using the setup in Fig. 3 with a 7.5 A load. Over 50 hours, 128 k samples were taken every 15 minutes using *pstest*. Average, minimum, and maximum power values were calculated for each point. Marginal fluctuations ($\pm$ 0.09 W) were observed in the average values, with more noise in the minimum and maximum values. These results indicate that the PowerSensor3 remains stable and does not require recalibration after production.

C. Step response

To measure the step response of the PowerSensor3, a 12 V / 10 A sensor, sampling at 20 kHz, is connected to the electronic load. The load is configured to 8 A, with a frequency modulation of 100 Hz and a modulation depth of 50%. The

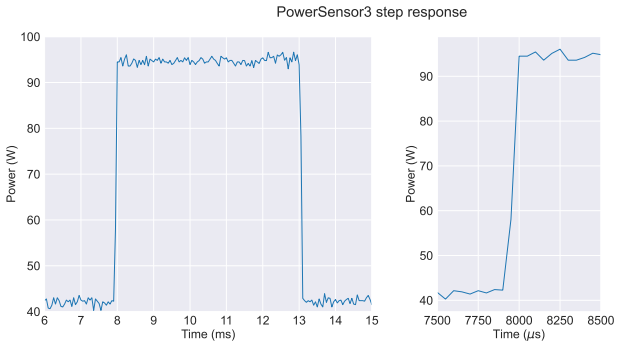


Fig. 5: Step response of PowerSensor3: load stepped from 3.3 A to 8 A plotted in ms scale (left) and μs scale (right).

results are shown in Fig. 5. The step response is clearly visible, illustrating that the PowerSensor3 is well suited to measure power transients, such as the start and stop of a GPU kernel.

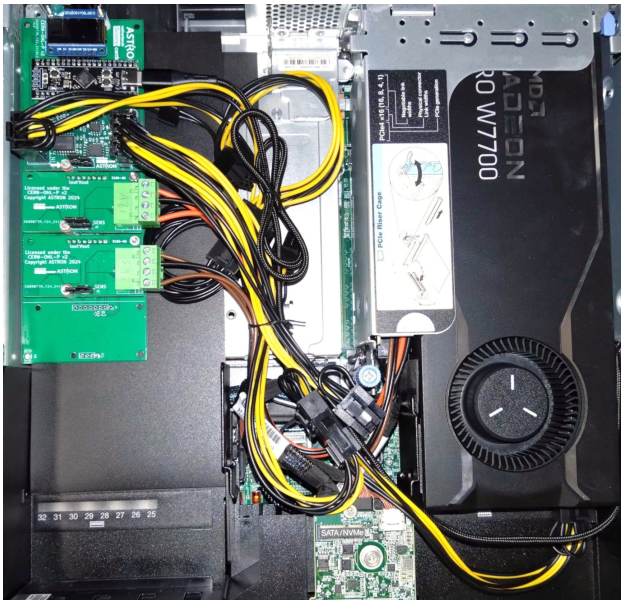
V. Application Case Studies

This section presents three use cases of PowerSensor3 demonstrating its capability to give highly-detailed insights into the power consumption of peripheral and embedded devices. The three case studies are: (1) discrete GPUs, (2) SoC boards, and (3) SSDs.

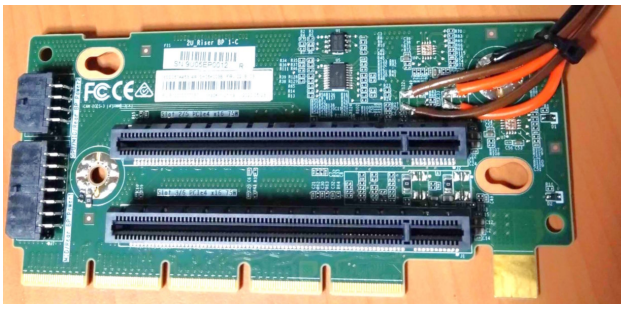
A. GPUs

In this section, we illustrate the application of PowerSensor3 in monitoring the power consumption of GPU applications. We have equipped multiple compute nodes in the DAS6 cluster [50] with a PowerSensor3 with 3 sensor boards as shown in Fig. 6(a). Two sensor boards for the 3.3 V and 12 V PCIe power channels and one for the 12 V PSU power channel. The PCIe power channels are intercepted using a modified PCIe Gen 4 riser card as shown in Fig. 6(b). By removing a 0-ohm resistor and attaching wires for each power channel, we created measurement points for the power supplies without compromising signal integrity. We consider two use cases. First, we use PowerSensor3 to monitor the power consumption of a single kernel executing on the GPU and compare the results with the GPU's internal power sensor using Power Measurement Toolkit. Secondly, we use PowerSensor3 to monitor the power consumption while automatically optimizing a realistic GPU application for both compute performance and energy efficiency using Kernel Tuner.

1) Power Measurement Toolkit: The Power Measurement Toolkit (PMT) is an open-source high-level software library for measuring and monitoring power consumption across various hardware platforms [51]. Written in C++, PMT leverages vendor-specific APIs to collect power usage data. For NVIDIA GPUs, it uses NVML and, for AMD GPUs, ROCm SMI and its successor AMD SMI are both supported, while for CPUs, it utilizes the RAPL interface or LIKWID [52]. Additionally, PMT supports profiling other architectures, such as AMD FPGAs, and its modular design allows for straightforward extension to new hardware.



(a) PowerSensor3 attached to an AMD W7700 GPU.



(b) Modified PCIe gen 4 riser card (Lenovo SR665), providing measurement points for 3.3 V and 12 V.

Fig. 6: A node in the DAS6 [50] cluster computer equipped with a PowerSensor3 to measure GPU power consumption.

PMT provides a unified interface for power measurement, catering to both C++ and Python applications. PMT is particularly suited for high-performance computing researchers and developers, who can use it to evaluate and optimize energy efficiency, but it is equally valuable for a general-purpose public requiring a simple yet effective software-based power measurement tool.

Yang et al. examined over 70 GPUs across 12 architectural generations and revealed significant inaccuracies in NVML power readings, leading to severe under- or overestimates of energy consumption [38]. Although mitigations were proposed, these issues underline the need for more reliable alternatives. PMT addresses these concerns by offering support for PowerSensor3, which provides accurate and consistent power measurements without the caveats identified in NVML.

We compare the PowerSensor3 energy measurement with NVML on an NVIDIA RTX 4000 Ada GPU in Fig. 7a and with AMD SMI on an AMD W7700 GPU in Fig. 7b. The measurement starts with a brief idle time, followed by a synthetic load of fused multiply-add instructions. A two-dimensional grid is used, where the x-dimension of the grid is set according to the number of streaming multiprocessors (SMs) or Compute Units (CU) on the NVIDIA and AMD GPU, respectively. The y-dimension is set such that the kernel runs for roughly two seconds.

On the NVIDIA RTX 4000 Ada GPU, energy consumption initially spikes to approximately 95 W before increasing to around 120 W. This behavior corresponds to the gradual ramp-up of the clock frequency, which does not reach its peak instantaneously. Distinct phases are visible in the energy profile, corresponding to the sequential execution of thread blocks along the y-dimension of the grid. The power dips between individual phases are made clearly visible by PowerSensor3, but are missed by NVML. After the workload completes, the GPU requires over a second to return to its idle power state. While NVML's instantaneous energy measurement aligns reasonably well with PowerSensor3, its time resolution cannot capture fine-grained GPU behavior. NVML's 'legacy' average power measurement is limited to coarse-grained energy estimations and completely inadequate to measure the kernel's energy use accurately.

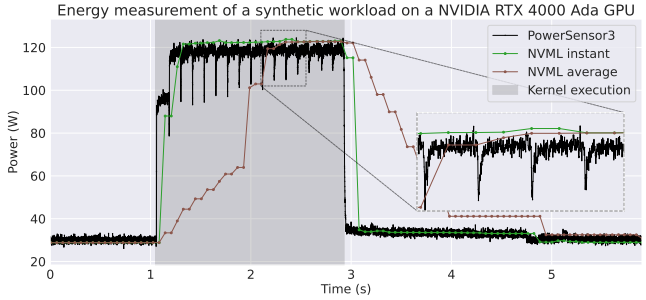
On the AMD W7700 GPU, we compare PowerSensor3 with both ROCm SMI and AMD SMI APIs, which yields identical results despite differences in their programming interfaces. Unlike NVML, the built-in energy measurements of the W7700 GPU closely match PowerSensor3, demonstrating excellent accuracy. The energy profile reveals distinct phases of the GPU's power and frequency behavior: an initial spike to the 150 W power limit is followed by a sharp drop, a ramp-up phase with brief power overshoot, and eventual stabilization at the power limit. Notably, the GPU returns to its idle power state more rapidly than the NVIDIA GPU.

In conclusion, high-resolution energy measurement tools like PowerSensor3 are critical for uncovering GPU behavior that remains invisible to standard performance profilers, particularly in capturing transient power fluctuations that are not detectable at lower sampling rates. Given the limited disclosure from vendors regarding their built-in energy measurement mechanisms, reliable external tools are essential to ensure accurate and detailed energy analyses.

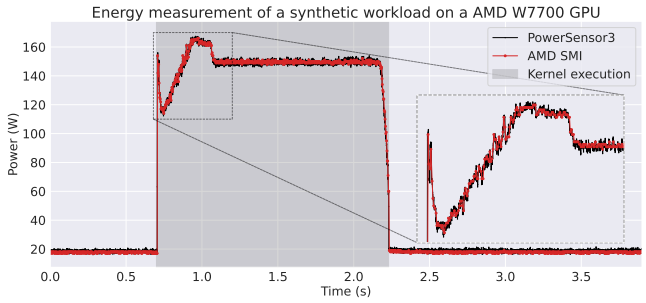
2) Kernel Tuner: We now use PowerSensor3 to provide power measurements while automatically optimizing a real-world GPU kernel for computational performance as well as energy efficiency.

We use the Tensor-Core Beamformer [53] as an example GPU application. Beamforming is a well-known technique to combine signals from multiple receivers. The Tensor-Core Beamformer has been developed for use in both radio astronomy and ultrasound imaging. The beamformer uses tensor cores on NVIDIA GPUs or matrix cores on AMD GPUs to perform complex matrix multiplications, which are not supported by vendor libraries such as cuBLAS or CUTLASS. In this case study, we use 16-bit input and output data with M=4096 beams, N=4096 samples at a time, and K=4096 elements summed.

The Tensor-Core Beamformer can be automatically tuned to achieve optimal performance on a specific GPU using



(a) NVIDIA's NVML library provides two different energy measurements, 'instantaneous' and 'average'. The former provides a better sampling rate. High time-resolution energy measurements such as PowerSensor3 uncover GPU behavior that is not visible with NVML.



(b) The AMD SMI and PowerSensor3 measurement closely align.

Fig. 7: Energy measurements for a synthetic GPU workload using PowerSensor3 and vendor supplied software-based measurements. The shaded area marks the kernel execution, and the insets highlight specific GPU behavior uncovered by energy analysis of the workload.

Kernel Tuner [54]. Kernel Tuner is an open-source GPU auto-tuner that allows users to define parameters in the code to be tuned. The auto-tuner constructs a search space of all possible functionally-equivalent code variants and automatically searches for the specific combination of tunable parameter values that achieves the best performance. During the auto-tuning process, Kernel Tuner performs many empirical measurements to obtain the execution time and power consumption of each variant. In a typical use case, the tuner compiles and benchmarks several thousands of different code variants on the GPU.

Kernel Tuner supports capturing the energy consumption of GPU kernels [22], which is typically measured using on-board current sensors, either using NVML on NVIDIA GPUs or ROCm-SMI through PMT for AMD GPUs. However, as explained in Section II and confirmed in Fig. 7a, NVIDIA's on-board current sensors typically have a time resolution of about 10 Hz, which is much too low to accurately capture the power consumed by real-world GPU kernels, which typically take at most a few tens of milliseconds. When using onboard current sensors for power measurement, Kernel Tuner therefore first executes the GPU kernel repeatedly to determine the execution time, and then runs the kernel continuously for an extended period, for example, 1 or 2 seconds, to collect sufficient measurements from the on-board sensor. As the tuner typically benchmarks several thousands of code variants, this means the tuning process is extended by several hours, which wastes both time and energy.

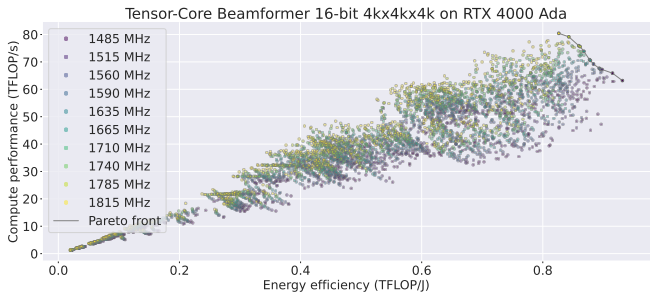


Fig. 8: Tuning results for the Tensor-Core Beamformer on the NVIDIA RTX 4000 Ada.

We have integrated support for PowerSensor3 directly into Kernel Tuner, which allows for instant capturing of the energy consumption of GPU kernels. In this way, there is no need for Kernel Tuner to continuously run the kernel for several seconds, effectively saving hours of tuning time.

To auto-tune the Tensor-Core Beamformer for both energy and time efficiency on the NVIDIA RTX 4000 Ada GPU, we used the performance model presented in [22] to narrow down the range of GPU clock frequencies to tune for. The other tunable parameters that can be varied in the code are the thread block dimensions, the number of submatrices (fragments) per thread block and per warp, and the extent to which double buffering in shared memory is applied. In total there are 512 different code variants, with 10 different GPU clock frequencies, this amounts to an auto-tuning search space of 5120 configurations, that are averaged over 7 trials each.

Fig. 8 shows the energy efficiency in tera-flop per joule (TFLOP/J) and compute performance in tera-flops per second (TFLOP/s) of the code variants benchmarked during auto-tuning the Tensor-Core Beamformer on the NVIDIA RTX 4000 Ada. Overall, we observe that performance and energy efficiency are correlated. However, especially among the more efficient configurations, there is a wider spread in both energy and compute efficiency. The fastest Pareto optimal configuration achieves a compute performance of 80.4 TFLOP/s at 0.83 TFLOP/J energy efficiency, whereas the most energy efficient configuration is 12.7% more energy efficient, but also has a 21.5% slowdown compared to the fastest configuration. Overall, collecting all data points from Fig. 8 using PowerSensor3 took 2274.4 seconds, which would have taken about 7394 seconds if we had used the onboard power sensor instead. Thus, thanks to PowerSensor3, we were able to perform this experiment in 3.25x less time.

B. SoC boards (NVIDIA Jetson)

The NVIDIA Jetson series System-on-Chips contain a tightly integrated CPU and GPU, and are used in GPU-accelerated edge-computing systems. Fig. 9 shows an NVIDIA Jetson AGX Orin development kit where the SoC module is combined with a carrier board. The system is powered by a USB-C connector, which is routed through PowerSensor3.

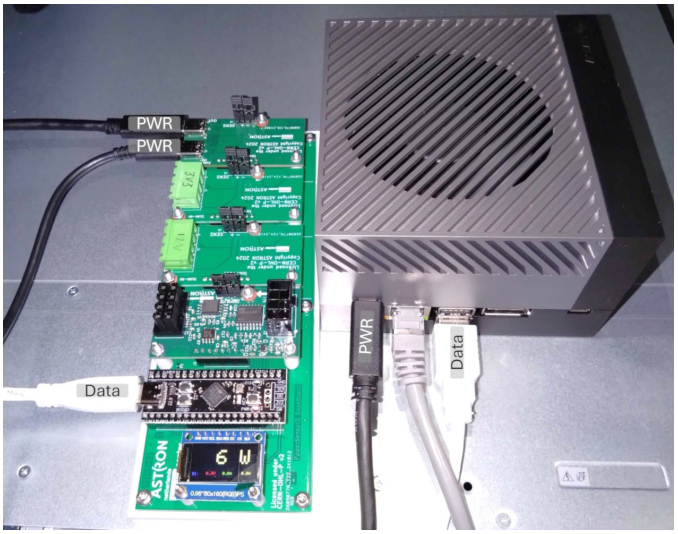


Fig. 9: NVIDIA Jetson AGX Orin with PowerSensor3 on the USB-C power supply, the display shows the idle power.

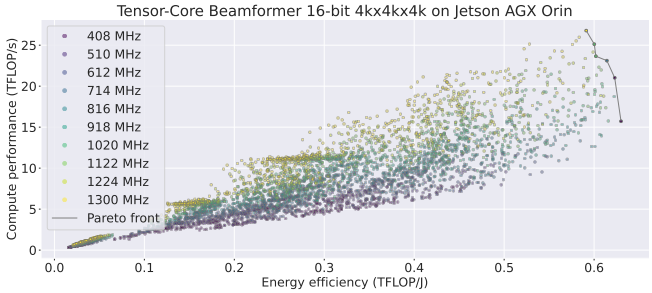


Fig. 10: Tuning results for the Tensor-Core Beamformer on the NVIDIA Jetson AGX Orin.

We repeat the same measurement as on the RTX 4000 Ada (Fig. 8). The tuning results are shown in Fig. 10. The overall behavior is similar to the RTX 4000 Ada. PowerSensor3 provides several advantages over the built-in sensor of the Jetson: the time resolution of the built-in sensor is very limited (∼0.1 second), and similar to the RTX 4000 Ada we can perform this experiment much faster with PowerSensor3. Additionally, the built-in sensor only measures the power consumption of the Jetson module, not including the carrier board that the module is inserted into. With PowerSensor3, we are able to measure the power consumption of the entire device.

C. SSDs

Apart from GPUs, another major power consumer in data centers is data storage [56], with power usage estimates ranging from 10% [57, 58] to as much as 25–30% [59]. In order to reduce the power utilization of storage, it is important to understand the power contribution of individual hardware components, such as individual SSDs. However, storage devices do not report their power usage and rely on external sensors. In this section, we demonstrate that the PowerSensor3 is an effective external sensor for modern storage devices.

Numerous investigations have been conducted to measure SSD power consumption, categories related to this work are:

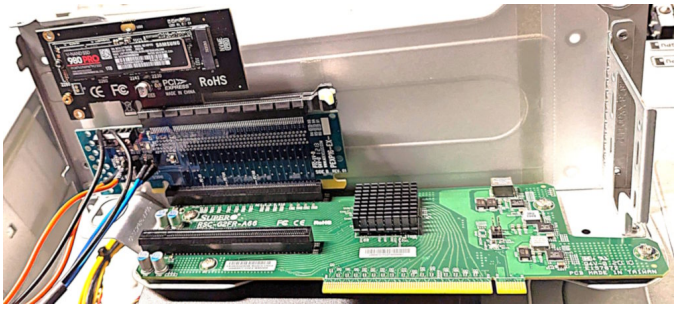


Fig. 11: The NVMe to PCIe adapter with the Samsung 980 PRO 1 TB M.2 SSD [55] in a modified PCIe (gen 3) riser card, providing measurement points for 3.3 V and 12 V.

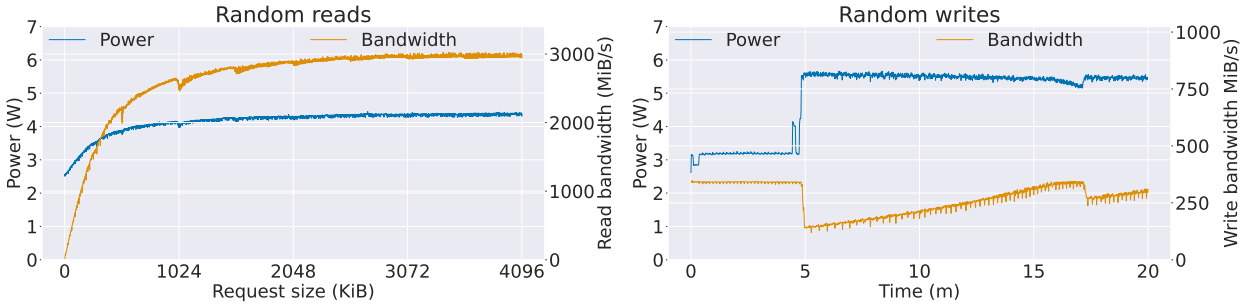
- Individual flash chips and SATA drives [60–63].
- Whole system energy of software on NVMe [64–67].
- Analysis of NVMe SSDs, using a custom external sensor [58], which samples at 1 kHz.

However, these studies either do not measure power of individual SSDs, lack standardized tooling, can not measure at the desired granularity, or do not apply to NVMe flash SSDs. PowerSensor3 allows for a standardized approach for SSDs with a configurable granularity in sample frequency (sub-milliseconds to seconds).

For evaluation we use a Samsung 980 PRO 1 TB M.2 SSD with the hardware setup visualized in Fig. 11. The Supermicro system (SYS-2029GP-TR) used in this set-up does not easily provide access to PCIe slot power. Therefore, we use an additional PCIe 3.0 riser card, modified similar to the situation described for GPU measurements, providing measurement points for the PCIe 3.3 V and 12 V power channels. We use the state-of-the-practice fio workload generator [68] with direct I/O and the *io_uring* engine with recommended performance optimizations [69]. As demonstrative workloads, we use random reads at various request sizes and use a long-running random write workload.

First, we evaluate the impact of I/O request size for random reads on bandwidth and power. It is well-known that larger requests typically lead to increased SSD bandwidth and power [58, 61] as more work can be done in parallel. To reproduce these observations, we run 10 second long random read workloads at request sizes ranging from 1–4096 KiB (Δ1 KiB). In Fig. 12a, we plot the read request size on the x-axis, and the average power usage and bandwidth on the y-axes. We confirm that power and bandwidth both increase with the request size (expected) until the device is saturated.

Second, we evaluate the bandwidth and power for a longer-running (>20 minute) random write workload. Flash SSDs (with a block interface) are known to suffer from performance variability when consistently writing randomly, which is largely due to an SSD-internal process known as garbage collection (GC). GC issues reads and writes that interfere with reads and writes issued by the host, which leads to performance variability. Past studies have observed that this variability does not necessarily translate to similar trends in SSD power [61, 63]. Such discrepancies have implications for



(a) Random reads, mean per request size.

(b) Random writes, mean over time.

Fig. 12: Power and bandwidth results for the Samsung 980 PRO benchmarking: (a) random reads; (b) random writes.

host-managed solutions that attempt to reduce SSD power or estimate SSD power usage, since bandwidth is not an accurate indicator of power. We evaluate if these observations also hold for the evaluated SSD. We first format the NVMe SSD, then precondition with 128 KiB sequential writes, and lastly issue random 4 KiB writes until the SSD is in steady-state. Fig. 12b shows the power and bandwidth (y-axes) over time (x-axis) for the random writes, using a granularity of one second for both power and bandwidth. We observe that bandwidth is highly variable, but power increases to 5 W at the first bandwidth descend, and remains relatively stable afterward. We thus confirm that bandwidth is not indicative of power consumption. Therefore, to accurately evaluate SSD power for a given workload, we recommend using an external sensor.

To conclude, the PowerSensor3 allows us to reproduce prior SSD energy measurements, but with the added advantage that the sensor is standardized and can be installed within servers (deployment flexibility). While we have evaluated our storage workloads at 1 ms granularity, the PowerSensor3 is able to measure at sub-millisecond granularity (e.g., >1 kHz) which will be evaluated in more detail in future work.

VI. Discussion

Calibration and ease of use: The calibration and evaluation of the sensors, as described in this work, follow standard procedures for such devices. Calibration is required only once during production, ensuring long-term reliability and accuracy. The source code repository includes comprehensive documentation detailing the productions and calibration process, making it accessible for users to understand and implement, when they wish to produce their own hardware. Despite the open nature of the device and accompanying software, we acknowledge that not everyone may be able to manufacture the device independently. To address this, we have started an initiative to explore if we can provide fully assembled and calibrated devices, allowing for broader accessibility and ease of use of the PowerSensor3 technology [70].

Extendibility of PowerSensor3: The current design of PowerSensor3 allows to measure up to four different power supplies to a device, which can range from GPU cards and SoC boards to custom boards with ASICs. The provided software is compatible with any host system running Linux, offering flexibility and adaptability to various use cases. Both the hardware and software can be tailored to specific requirements such as different power ranges, sensor accuracy, connectors types and form factor. We encourage others to make their sensor boards available under an open hardware license and to open pull requests on the hardware and software repositories.

VII. Conclusion

The application case studies presented in Section V illustrate that PowerSensor3 offers an open, cost-efficient solution for fine-grained, high-frequency power measurements on a variety of peripheral devices. Thus, enabling a deeper understanding of system-level energy consumption and guiding effective optimization strategies.

For the NVIDIA RTX 4000 Ada GPU, PowerSensor3 reveals previously undetected behavior, outperforming built-in sensors. For the AMD W7700 GPU, it shows comparable time and amplitude accuracy to the built-in sensor, which specifications are not well documented. PowerSensor3 also reduced the Tensor-Core Beamformer application auto-tuning time by 3.25x compared to NVIDIA's internal sensor.

Additionally, PowerSensor3 works with SoC boards like the NVIDIA Jetson AGX Orin, which lacks total system power reporting. The power consumption of PowerSensor3 itself is minimal, measured in milliwatts, which is negligible compared to SoC boards. Alternatively, the PowerSensor3 can be powered separately, eliminating the need to draw power from the monitored system.

A case study with a PCIe SSD demonstrated that PowerSensor3 uncovers behavior not observable from bandwidth metrics alone, proving its utility for PCIe devices without built-in sensors.

PowerSensor3 allows developers and researchers to use energy as a metric for software optimization and evaluate the efficiency of new hardware platforms. PowerSensor3 can play a pivotal role in reducing the energy footprint of large-scale AI, HPC, and data center operations.

Acknowledgment

We would like to thank Quinten Twisk for his work on an early prototype of PowerSensor3.

REFERENCES

[1] E. Masanet, A. Shehabi, N. Lei, S. Smith, and J. Koomey, "Recalibrating global data center energy-use estimates," *Science*, vol. 367, no. 6481, pp. 984–986, 2020.

[2] A. de Vries, "The growing energy footprint of artificial intelligence," *Joule*, vol. 7, no. 10, pp. 2191–2194, 2023.

[3] "Hungry for Energy, Amazon, Google and Microsoft Turn to Nuclear Power," 2024. [Online]. Available: https://www.nytimes.com/2024/10/16/business/energy-environment/amazon-google-microsoft-nuclear-energy.html

[4] "Top500," 2024. [Online]. Available: https://top500.org

[5] A. R. Stevens, S. Bellstedt, P. J. Elahi, and M. T. Murphy, "The imperative to reduce carbon emissions in astronomy," *Nature Astronomy*, vol. 4, no. 9, pp. 843–851, 2020.

[6] H. Ritchie, M. Roser, and P. Rosado, "Renewable Energy (Last revised Jan 2024)," *Our World in Data*, 2020, https://ourworldindata.org/renewable-energy.

[7] A. Shehabi, S. J. Smith, E. Masanet, and J. Koomey, "Data center growth in the united states: decoupling the demand for services from electricity use," *Environmental Research Letters*, vol. 13, no. 12, p. 124030, 2018.

[8] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[9] S. Heldens, P. Hijma, B. V. Werkhoven, J. Maassen, A. S. Belloum, and R. V. Van Nieuwpoort, "The landscape of exascale research: A data-driven literature analysis," *ACM Computing Surveys (CSUR)*, vol. 53, no. 2, pp. 1–43, 2020.

[10] "Frontier: OLCF's Exascale Future," 2018. [Online]. Available: https://www.olcf.ornl.gov/2018/02/13/frontier-olcfs-exascale-future/

[11] M. Stachowski, A. Fiebig, and T. Rauber, "Autotuning based on frequency scaling toward energy efficiency of blockchain algorithms on graphics processing units," *The Journal of Supercomputing*, vol. 77, pp. 263–291, 2021.

[12] 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 *2013 42nd International Conference on Parallel Processing*. IEEE, 2013, pp. 826–833.

[13] X. Mei, L. S. Yung, K. Zhao, and X. Chu, "A measurement study of GPU DVFS on energy conservation," in *Proceedings of the Workshop on Power-Aware Computing and Systems*, 2013, pp. 1–5.

[14] 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," *Computer Science-Research and Development*, vol. 31, no. 4, pp. 185–193, 2016.

[15] S. Akiki, Z. Yang, C. Liu, J. Tang, and S. Liu, "Energy-Aware Automatic Tuning of Many-Core Platform via Gradient Descent," in *2018 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*. IEEE, 2018, pp. 1199–1203.

[16] T. Katagiri, C. Luo, R. Suda, S. Hirasawa, and S. Ohshima, "Energy optimization for scientific programs using auto-tuning language ppOpen-AT," in *2013 IEEE 7th International Symposium on Embedded Multicore Socs*. IEEE, 2013, pp. 123–128.

[17] J. Guerreiro, A. Ilic, N. Roma, and P. Tomás, "Multi-kernel auto-tuning on GPUs: Performance and energy-aware optimization," in *2015 23rd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*. IEEE, 2015, pp. 438–445.

[18] A. Krzywaniak and P. Czarnul, "Performance/energy aware optimization of parallel applications on gpus under power capping," in *International Conference on Parallel Processing and Applied Mathematics*. Springer, 2019, pp. 123–133.

[19] K. Datta, M. Murphy, V. Volkov, S. Williams, J. Carter, L. Oliker, D. Patterson, J. Shalf, and K. Yelick, "Stencil computation optimization and auto-tuning on state-of-the-art multicore architectures," in *Proceedings of the 2008 ACM/IEEE conference on Supercomputing*. IEEE Press, 2008, p. 4.

[20] S. Huang, S. Xiao, and W.-c. Feng, "On the energy efficiency of graphics processing units for scientific computing," in *2009 IEEE International Symposium on Parallel & Distributed Processing*. IEEE, 2009, pp. 1–8.

[21] T. Dong, V. Dobrev, T. Kolev, R. Rieben, S. Tomov, and J. Dongarra, "A step towards energy efficient computing: Redesigning a hydrodynamic application on CPU-GPU," in *2014 IEEE 28th International Parallel and Distributed Processing Symposium*. IEEE, 2014, pp. 972–981.

[22] R. Schoonhoven, B. Veenboer, B. Van Werkhoven, and K. J. Batenburg, "Going green: optimizing GPUs for energy efficiency through model-steered auto-tuning," in *2022 IEEE/ACM International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS)*. IEEE, 2022, pp. 48–59.

[23] L. Li and C. Kessler, "MeterPU: a generic measurement abstraction API enabling energy-tuned skeleton backend selection," in *2015 IEEE Trustcom/BigDataSE/ISPA*, vol. 3. IEEE, 2015, pp. 154–159.

[24] E. M. Garzón, J. Moreno, and J. A. Martínez, "An approach to optimise the energy efficiency of iterative computation on integrated GPU–CPU systems," *The Journal of Supercomputing*, vol. 73, no. 1, pp. 114–125, 2017.

[25] R. Nobre, L. Reis, and J. M. Cardoso, "Compiler phase ordering as an orthogonal approach for reducing energy consumption," *arXiv preprint arXiv:1807.00638*, 2018.

[26] J. Pallister, S. J. Hollis, and J. Bennett, "Identifying compiler options to minimize energy consumption for embedded platforms," *The Computer Journal*, vol. 58, no. 1, pp. 95–109, 2015.

[27] J. W. Romein and B. Veenboer, "PowerSensor 2: A Fast Power Measurement Tool," in *2018 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. IEEE, 2018, pp. 111–113.

[28] G. Schoonderbeek, S. van der Vlugt, J. Romein, and L. Oostrum, "Powersensor3 hardware," Mar. 2025. [Online]. Available: https://doi.org/10.5281/zenodo.15039399

[29] L. Oostrum, J. Romein, B. van Werkhoven, Q. Twisk, G. Schoonderbeek, and S. van der Vlugt, "Powersensor3," Nov. 2024. [Online]. Available: https://doi.org/10.5281/zenodo.14216576

[30] W. Jia, E. Garza, K. A. Shaw, and M. Martonosi, "GPU performance and power tuning using regression trees," *ACM Transactions on Architecture and Code Optimization (TACO)*, vol. 12, no. 2, pp. 1–26, 2015.

[31] H. Anzt, B. Haugen, J. Kurzak, P. Luszczek, and J. Dongarra, "Experiences in autotuning matrix multiplication for energy minimization on GPUs," *Concurrency and Computation: Practice and Experience*, vol. 27, no. 17, pp. 5096–5113, 2015.

[32] I. Grasso, P. Radojkovic, N. Rajovic, I. Gelado, and A. Ramirez, "Energy efficient HPC on embedded SoCs: Optimization techniques for mali GPU," in *2014 IEEE 28th International parallel and distributed processing symposium*. IEEE, 2014, pp. 123–132.

[33] P. Schiffmann, D. Martin, G. Haase, and G. Offner, "Optimizing a RBF interpolation solver for energy on heterogeneous systems," in *Parallel Computing is Everywhere*. IOS Press, 2018, pp. 287–296.

[34] K. N. Khan, M. Hirki, T. Niemi, J. K. Nurminen, and Z. Ou, "RAPL in action: Experiences in using rapl for power measurements," *ACM Transactions on Modeling and Performance Evaluation of Computing Systems (TOMPECS)*, vol. 3, no. 2, pp. 1–26, 2018.

[35] NVIDIA. (2012) NVIDIA's Next Generation CUDA Compute Architecture: Kepler GK110/210. [Online]. Available: https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/tesla-product-literature/NVIDIA-Kepler-GK110-GK210-Architecture-Whitepaper.pdf

[36] J. Lang and G. Rünger, "High-resolution power profiling of GPU functions using low-resolution measurement," in *European Conference on Parallel Processing*. Springer, 2013, pp. 801–812.

[37] M. Burtscher, I. Zecena, and Z. Zong, "Measuring GPU power with the K20 built-in sensor," in *Proceedings of Workshop on General Purpose Processing Using GPUs*, 2014, pp. 28–36.

[38] Z. Yang, K. Adamek, and W. Armour, "Accurate and Convenient Energy Measurements for GPUs: A Detailed Study of NVIDIA GPU's Built-In Power Sensor," in *2024 SC24: International Conference for High Performance Computing, Networking, Storage and Analysis SC*. IEEE Computer Society, 2024, pp. 307–323.

[39] G. Wu, J. L. Greathouse, A. Lyashevsky, N. Jayasena, and D. Chiou, "GPGPU performance and power estimation using machine learning," in *2015 IEEE 21st international symposium on high performance computer architecture (HPCA)*. IEEE, 2015, pp. 564–576.

[40] G. Schieffer, D. A. De Medeiros, J. Faj, A. Marathe, and I. Peng, "On the rise of AMD matrix cores: Performance, Power Efficiency, and Programmability," in *2024 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. IEEE, 2024, pp. 132–143.

[41] R. S. Chen and J. K. Hollingsworth, "Angel: A hierarchical approach to multi-objective online auto-tuning," in *Proceedings of the 5th International Workshop on Runtime and Operating Systems for Supercomputers*, 2015, pp. 1–8.

[42] D. Q. Ren, "Algorithm level power efficiency optimization for CPU–GPU processing element in data intensive SIMD/SPMD computing," *Journal of Parallel and Distributed Computing*, vol. 71, no. 2, pp. 245–253, 2011.

[43] R. Suda, L. Cheng, and T. Katagiri, "A mathematical method for online autotuning of power and energy consumption with corrected temperature effects," *Procedia Computer Science*, vol. 18, pp. 1302–1311, 2013.

[44] C. Timm, F. Weichert, P. Marwedel, and H. Müller, "Design space exploration towards a realtime and energy-aware GPGPU-based analysis of biosensor data," *Computer Science-Research and Development*, vol. 27, no. 4, pp. 309–317, 2012.

[45] D. Bedard, M. Y. Lim, R. Fowler, and A. Porterfield, "Powermon: Fine-grained and integrated power monitoring for commodity computer systems," in *Proceedings of the IEEE SoutheastCon 2010 (SoutheastCon)*. IEEE, 2010, pp. 479–484.

[46] J. H. Laros, P. Pokorny, and D. DeBonis, "Powerinsight-a commodity power measurement capability," in *2013 International Green Computing Conference Proceedings*. IEEE, 2013, pp. 1–6.

[47] STMicroelectronics, "Datasheet STM32F411xC STM32F411xE," 2024. [Online]. Available: https://www.st.com/en/microcontrollers-microprocessors/stm32f411/documentation.html

[48] Melexis, "Datasheet MLX91221 Integrated Current Sensor IC," 2024. [Online]. Available: https://www.melexis.com/en/product/MLX91221/0-50A-isolated-3-3V-integrated-hall-current-sensor

[49] Broadcom, "Datasheet ACPL-C87B Precision Optically Isolated Voltage Sensor," 2024. [Online]. Available: https://www.broadcom.com/products/optocouplers/industrial-plastic/isolation-amplifiers-modulators/isolation-amplifiers/acpl-c87b

[50] H. Bal, D. Epema, C. de Laat, R. van Nieuwpoort, J. Romein, F. Seinstra, C. Snoek, and H. Wijshoff, "A Medium-Scale Distributed System for Computer Science Research: Infrastructure for the Long Term," *Computer*, vol. 49, no. 5, pp. 54–63, 2016.

[51] S. Corda, B. Veenboer, and E. Tolley, "PMT: Power Measurement Toolkit," in *2022 IEEE/ACM International Workshop on HPC User Support Tools (HUST)*. IEEE, 11 2022, pp. 44–47. [Online]. Available: https://ieeexplore.ieee.org/document/10027520/

[52] J. Treibig, G. Hager, and G. Wellein, "LIKWID: A lightweight performance-oriented tool suite for x86 multicore environments," *Proceedings of the International Conference on Parallel Processing Workshops*, pp. 207–216, 2010.

[53] L. Oostrum, B. Veenboer, R. Rook, M. Brown, P. Kruizinga, and J. W. Romein, "The Tensor-Core Beamformer: A High-Speed Signal-Processing Library for Multidisciplinary Use," in *39th IEEE International Parallel & Distributed Processing Symposium (IPDPS)*. IEEE, 2025.

[54] B. van Werkhoven, "Kernel Tuner: A search-optimizing GPU code auto-tuner," *Future Generation Computer Systems*, vol. 90, pp. 347–358, 2019.

[55] Samsung, "NVMe SSD 980 Pro Data sheet Rev 2.1," https://download.semiconductor.samsung.com/resources/data-sheet/Samsung-NVMe-SSD-980-PRO-Data-Sheet_Rev.2.1.pdf, Accessed: 2024-12-02.

[56] V. Rao and A. A. Chien, "Understanding the Operational Carbon Footprint of Storage Reliability and Management," *HotCarbon*, 2024.

[57] A. Shehabi, S. Smith, D. Sartor, R. Brown, M. Herrlin, J. Koomey, E. Masanet, N. Horner, I. Azevedo, and W. Lintner, "United states data center energy usage report," 2016.

[58] D. Xie, T. Stavrinos, K. Zhu, S. Peter, B. Kasikci, and T. Anderson, "Can Storage Devices be Power Adaptive?" in *Proceedings of the 16th ACM Workshop on Hot Topics in Storage and File Systems*, 2024, pp. 47–54.

[59] H. Cao, S. Bergman, S. Sun, Y. A. Zhou, X. Li, J. Gao, Z. Cheng, and J. Zhang, "Answering the Call to ARMs with PACER: Power-Efficiency in Storage Servers," *MSST*, 2024.

[60] M. Bjorling, P. Bonnet, L. Bouganim, and B. P. Jónsson, "uFLIP: Understanding the Energy Consumption of Flash Devices," *Bulletin of the Technical Committee on Data Engineering*, vol. 33, no. 4, pp. 48–54, 2010.

[61] S. Cho, C. Park, Y. Won, S. Kang, J. Cha, S. Yoon, and J. Choi, "Design Tradeoffs of SSDs: From Energy Consumption's Perspective," *ACM Transactions on Storage (TOS)*, vol. 11, no. 2, pp. 1–24, 2015.

[62] L. M. Grupp, A. M. Caulfield, J. Coburn, S. Swanson, E. Yaakobi, P. H. Siegel, and J. K. Wolf, "Characterizing Flash Memory: Anomalies, Observations, and Applications," in *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*, 2009, pp. 24–33.

[63] E. Seo, S.-Y. Park, and B. Urgaonkar, "Empirical Analysis on Energy Efficiency of Flash-based SSDs," in *HotPower*, 2008.

[64] B. Harris and N. Altiparmak, "Ultra-Low Latency SSDs' Impact on Overall Energy Efficiency," in *12th USENIX Workshop on Hot Topics in Storage and File Systems, HotStorage 2020, July 13-14, 2020*, A. Badam and V. Chidambaram, Eds. USENIX Association, 2020. [Online]. Available: https://www.usenix.org/conference/hotstorage20/presentation/harris

[65] ——, "When Poll is More Energy Efficient than Interrupt," in *Proceedings of the 14th ACM Workshop on Hot Topics in Storage and File Systems*, ser. HotStorage '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 59–64. [Online]. Available: https://doi.org/10.1145/3538643.3539747

[66] S. Sundar, W. Simpson, J. Higdon, C. Whitaker, B. Harris, and N. Altiparmak, "Energy Implications of IO Interface Design Choices," in *Proceedings of the 15th ACM Workshop on Hot Topics in Storage and File Systems*, 2023, pp. 58–64.

[67] C. Whitaker, S. Sundar, B. Harris, and N. Altiparmak, "Do We Still Need IO Schedulers for Low-latency Disks?" in *Proceedings of the 15th ACM Workshop on Hot Topics in Storage and File Systems*, 2023, pp. 44–50.

[68] Jens Axboe, "Fio," https://github.com/axboe/fio, Accessed: 2024-12-02.

[69] Z. Ren and A. Trivedi, "Performance Characterization of Modern Storage Stacks: POSIX I/O, Libaio, SPDK, and io_uring," in *Proceedings of the 3rd Workshop on Challenges and Opportunities of Efficient and Performant Storage Systems*, 2023, pp. 35–45.

[70] NWO, "NWO Take-Off phase 1 grant: Commercial Feasibility of PowerSensor (CFPS)," 2024. [Online]. Available: https://doi.org/10.61686/FRXJD41196

[71] S. van der Vlugt, L. Oostrum, G. Schoonderbeek, B. van Werkhoven, B. Veenboer, K. Doekemeijer, and J. W. Romein, "Powersensor3 results," Mar. 2025. [Online]. Available: https://doi.org/10.5281/zenodo.15037451

[72] S. van der Vlugt, L. Oostrum, G. Schoonderbeek, B. van Werkhoven, B. Veenboer, K. Doekemeijer, and J. Romein, "ispass-2025-powersensor3-ssd-data," Mar. 2025. [Online]. Available: https://doi.org/10.5281/zenodo.15019311

[73] A. NLeSC. (2025) PowerSensor3 Documentation. [Online]. Available: https://powersensor3.readthedocs.io/en/latest/

APPENDIX

Appendix containing Artifact description.

A. Abstract

In this work, we introduce the PowerSensor3, a novel tool comprising custom-developed hardware, firmware, and software components.

The hardware architecture of the PowerSensor3 includes a base board and a sensor board, both of which have been meticulously designed and released as open hardware under the CERN Open Hardware License (CERN-OHL-P v2) at https://doi.org/10.5281/zenodo.15023417 [28]. This open hardware approach ensures transparency, reproducibility, and the potential for community-driven enhancements.

Complementing the hardware, the firmware and host software for the PowerSensor3 have been developed and released as open-source software under the Apache License 2.0 at https://doi.org/10.5281/zenodo.7941162 [29]. This open-source software framework facilitates seamless integration with existing systems and promotes collaborative development.

To validate the performance and accuracy of the PowerSensor3, we conducted extensive evaluations using the Power Measurement Toolkit (PMT) [51] and KernelTuner [54]. These tools were employed in conjunction with a Tensor Core Beamformer application [53] and storage benchmarks, enabling comprehensive analysis and benchmarking.

The results and findings from these evaluations are made available at https://doi.org/10.5281/zenodo.15037450 [71] and https://doi.org/10.5281/zenodo.15019310 [72], both licensed with Apache 2.0. By providing access to these results, we aim to foster further research and development in the field of power measurement.

B. Description

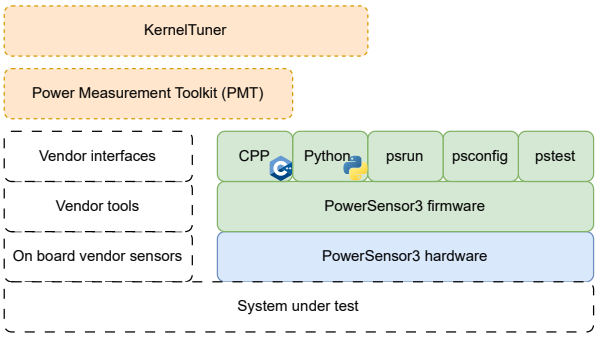


Fig. 13: Overview of components involved in the PowerSensor3 design and evaluation.

Fig. 13 illustrates the organization of the various components in the PowerSensor3 hardware and software stack:

System Under Test (SUT): The SUT, which can be a CPU, GPU, SoC, or other hardware components, often comes equipped with its own sensors, tools, and interfaces which can be compared to or combined with the PowerSensor3 measurements. These sensors, tools and interfaces are not part of this work, but are (when available) used in comparison to our PowerSensor3.

PowerSensor3 hardware: The PowerSensor3 hardware [28], shown in blue in Fig. 13, consists of the baseboard and sensor modules. The baseboard houses the STM32F411 microcontroller and supports up to four sensor modules, which can be customized to measure different power ranges and types of connectivity. Fig. 14 illustrates an example of the PowerSensor3 in operation. In this example, the PowerSensor3 is equipped with a PCIe sensor module that measures the power supplied to the PCIe card via the external power input as well as two sensor modules to measure the 3.3 V and 12 V PCIe slot power. A modified riser card, where the power connections for both 3.3 V and 12 V are interrupted and routed through two sensor modules, enables to measure the power consumption of the PCIe slot. The PowerSensor3 hardware design has been released as open hardware under the CERN Open Hardware License (CERN-OHL-P v2) at https://doi.org/10.5281/zenodo.15023417 [28].

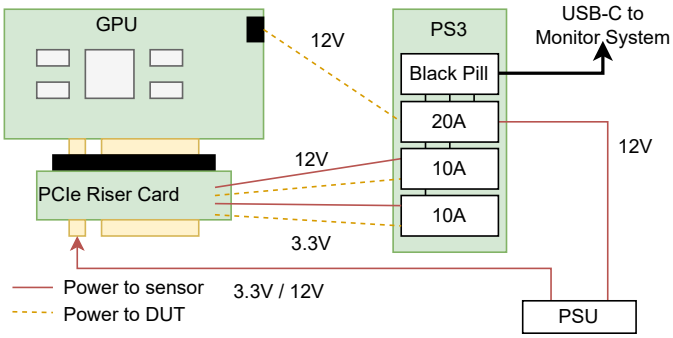


Fig. 14: Schematic of a PowerSensor3 measurement setup for PCIe devices.

PowerSensor3 firmware and software: The PowerSensor3 firmware and software [29], depicted in green in Fig. 13, play a vital role in the system's functionality. The firmware is programmed onto the PowerSensor3 hardware, enabling it to process and transmit power measurement data. The host system interacts with the hardware through the PowerSensor3 software, which provides a user-friendly interface for configuring, monitoring, and analyzing power consumption data. For easy use, the executable *psrun* can be used with a command line interface to report the power utilization of an existing application. For measurements with very high time resolution, Python and CPP interfaces are offered for integration with user applications. The PowerSensor3 firmware and software have been released as open-source software under the Apache License 2.0 at https://doi.org/10.5281/zenodo.7941162 [29].

Power Measurement Toolkit (PMT) and KernelTuner: PMT [51] and KernelTuner [54], shown in orange in Fig. 13, are versatile tools that can be used with both vendor-specific sensors and the PowerSensor3. PMT is designed for comprehensive power measurement and analysis across various devices, while KernelTuner facilitates the optimization of GPU

kernel performance across a broad range of parameters. These tools enhance the capability to evaluate and fine-tune the power consumption and efficiency of the SUT. PMT and KernelTuner are not part of this work, but are used in evaluation of our PowerSensor3. We have contributed the PowerSensor3 specific extensions of PMT and Kernel Tuner back to these projects.

For evaluation of PowerSensor3 with Kernel Tuner we carefully selected representative kernels that align with real-world high-performance GPU workloads where power efficiency is a critical concern. While vendor-provided reference implementations (e.g., CUTLASS and cuBLAS) may serve as performance baselines, our goal was to analyze power behavior in a use case from our application domains. The speedup achieved in tuning these specific kernels with Kernel Tuner are similar with the PowerSensor3 and the vendor tooling, however the tuning itself required 3.25× less time with PowerSensor3.

Evaluation results and SSD dataset: The results presented in this work [71] and the supplementary SSD dataset [72] provide valuable insights into the performance and accuracy of the PowerSensor3. These datasets are made available for evaluation purposes and include detailed examples on how to effectively utilize the PowerSensor3 for various applications. By sharing these results, we aim to support further research and development in power measurement technologies. The results and findings from these evaluations are made available at https://doi.org/10.5281/zenodo.15037450 [71] and https://doi.org/10.5281/zenodo.15019310 [72].

Documentation: The installation and use of the PowerSensor3 hardware, firmware and software is documented at: https://powersensor3.readthedocs.io/en/latest/ [73] and described in readme files in the individual repositories. Fig. 15 shows an example of assembly instructions as found in the hardware repository and Fig. 16 shows a fully assembled PowerSensor3 with three sensor modules populated.

Hardware dependencies: The PowerSensor3 firmware and software depend on the PowerSensor3 hardware.

Software dependencies: Software dependencies are described in the PowerSensor3 documentation [73] and are managed through git submodules and cmake files in the repository, the hardware design works with KiCAD and the software has been designed for Linux.

How to contribute: We encourage others to contribute to the development of PowerSensor3. Possible forms of contributions include: integration of the PowerSensor3 library with your software, pull requests for extensions to the hardware, firmware, software and documentation and design of your own sensor boards, made available under an open hardware license.

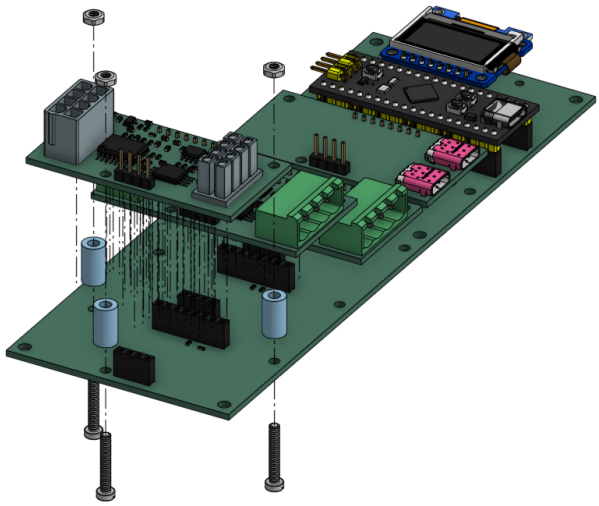


Fig. 15: Assembly instructions for the PowerSensor3 baseboard and sensor modules.

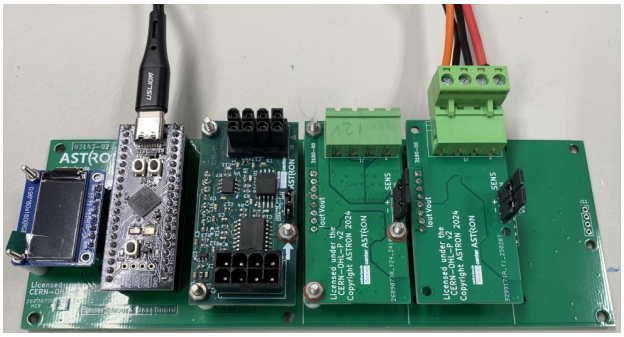


Fig. 16: Assembled PowerSensor3 baseboard with three sensor modules populated.