Profiling a Many-core Neuromorphic Platform

Indar Sugiarto, Luis A. Plana, Steve Temple, Basabdatta S. Bhattacharya, Steve B. Furber
School of Computer Science, University of Manchester, United Kingdom
Email: {indar.sugiarto, basab, luis.plana, steven.temple, steve.furber}@manchester.ac.uk

Patrick Camilleri
Crypto Quantique Ltd., United Kingdom
Email: pcamilleri@cryptoquantique.com

***Abstract*—This paper presents a unified profiling platform for a many-core machine that gives realtime information about the current state of the system. Such information is valuable for the observation and evaluation of running programs on the machine, as well as the health status of the machine itself. The information from the profiler can also be used for tuning the system's kernel operational parameters to maintain its performance and reliability. In our work, the profiler framework is developed for SpiNNaker, a many-core neuromorphic platfrom. The profiler framework provides realtime information such as power consumption, chip temperature, processor frequency, core utilization, and network connectivity. Many researchers and institutions have been using SpiNNaker not only for simulating spiking neural networks, but also for general purpose and energy-aware computing. Therefore, it is important to provide a reliable and comprehensive profiling platform for helping SpiNNaker users in developing their programs.**

***Index Terms*—system profiling, many-core, SpiNNaker, run-time management.**

I. Introduction

SpiNNaker (spiking neural network architecture) is a many-core neuromorphic platform developed originally for simulating large-scale spiking neural networks (SNNs) in biological real time [1]. However, it can also be used as a generic platform that provides a research infrastructure in the domain of many-core systems and high performance computing.

As a research platform for exploring the emerging many-core technology, the SpiNNaker development environment is in a progressive state where everything from the hardware upwards cannot be assumed to be totally reliable. Therefore, a reliable and comprehensive profiling tool is needed to help users and developers to deploy and optimize their programs on SpiNNaker.

Reading SpiNNaker's states such as temperature, clock frequency, and cores utilization is very useful in a scenario such as a runtime management system (RTM). Such a system is designed to optimize the performance of programs and to optimally make use of available resources. Dynamic Voltage and Frequency Scaling (DVFS) and Dynamic Power Management (DPM) are two hardware techniques for reducing power consumption commonly used in multi-core embedded system [2], [3]. We are interested on investigating such RTM techniques for many-core systems such as SpiNNaker. For this, we have developed a profiling tool that supports future RTM implementation on SpiNNaker.

In this paper, we introduce a unified and comprehensive profiling framework that provides the most important information about the current state of a running machine. The profiler framework will collect and present the following information from a SpiNNaker system: power consumption of several hardware modules (CPU-cores, SDRAM, FPGA, etc.), chip temperature, current clock frequency, CPU utilization, and on-chip networking activity. This information provides essential metrics for runtime system optimization. The aims of this paper are three-fold:

1) To give readers and the SpiNNaker user community an understanding of mechanisms for measuring various SpiNNaker performance metrics.
2) To demonstrate the importance of profiling SpiNNaker performance during software development.
3) To provide a measurement baseline for further complex analysis such as application-independent power usage modelling in a SpiNNaker machine.

The structure of this paper is organized as follow. Section II describes the platform development including hardware and software mechanisms with focus on a SpiNNaker board. Section III describes profiler functionalities in example use case scenarios. Section IV discusses the current state of the profiler framework, and a conclusion is presented in Section V.

II. Platform Development

A. SpiNNaker Hardware

Hierarchically, the SpiNNaker machine is constructed on several layers. At the lowest layer, there is a SpiNNaker chip, which is a many-core system-on-chip (SoC) with 18 ARM968 cores and 128 MB SDRAM. The second layer is the SpiNNaker board, which has 48 SpiNNaker chips mounted on a single PCB. The third layer is the SpiNNaker frame, which is constructed from 24 SpiNNaker boards. The fourth layer is the 19" SpiNNaker cabinet, which is constructed from 5 SpiNNaker frames. The final SpiNNaker machine will be assembled from ten SpiNNaker cabinets.

Each SpiNNaker chip is composed of two modules: a SpiNNaker die and an SDRAM module, where the SDRAM is mounted on top of the SpiNNaker die. The SpiNNaker die consists of several sub-modules: 18 ARM968 processors (each

has its own local fast static memory), 32 KB static shared memory, an on-chip router, a system controller, two PLLs, and three temperature sensors. All of these components are managed by the system controller.

As an SoC, a SpiNNaker chip is also equipped with a network-on-chip (NoC). Inside the chip, a dedicated router is responsible for managing various neuromorphic-style communication protocols. SpiNNaker communication is mainly done by using small size data packets of 40 or 72 bits. There are several types of packet used in the communication, but our work uses only two of them: multicast packet (MC) and point-to-point packet (P2P). Using P2P, a higher level communication protocol is built namely SpiNNaker Datagram Protocol (SDP). We use SDP as a means for transmitting various measurement values between chips and also to the host PC.

In our work, we have developed a profiler framework targeting a SpiNNaker board. Our profiler platform is designed to collect the following information: power consumption, temperature, frequency, CPU utilization, and packet traffic. The information about temperature, frequency, CPU utilization, and network traffic can be collected internally within each SpiNNaker chip; hence, an external acquisition system is not necessary. However, for power measurement an external acquisition system using Arduino was developed.

B. Profiler Components

1) Power Measurement: A SpiNNaker board contains not only SpiNNaker chips, but also several components that support inter-board communication as well as board management. For this, each SpiNNaker board is powered by six DC/DC converters with various voltage levels. Those are: three 1.2V supply for all ARM processors in all chips (grouped into three power banks), one 1.2V supply for logic fabric in three FPGAs, one 1.8V for SDRAM in all chips, and one 3.3V for the board management system (which includes a dedicated microcontroller called BMP and two Ethernet PHY modules).

For measuring power, a data acquisition system (DAS) using Arduino was developed. This DAS is equipped with six current-to-voltage (I2V) circuits, and a multi-channel analog-to-digital converter (ADC) type MAX11633. Each I2V module contains a current sense amplifier INA213 and a shunt resistor, where the resistor is placed in series with the corresponding DC/DC converter. The logical circuit diagram of this DAS along with the SpiNN-4 board is shown in Fig.1.

2) Temperature Measurement: There are three independent temperature sensors on the chip, each with its own control and sensor read-out register. The three sensors use different sensor mechanisms to enable the temperature to be corrected for process and voltage variations [4]. The first two sensors are linear sensors with opposite polarity (sensor-1 is an NTC sensor, whereas sensor-2 is a PTC sensor), and the third sensor is a non-linear sensor.

3) Frequency Measurement: There are several clocks inside a SpiNNaker chip: processor clock, router clock, SDRAM clock, and system controller clock. The clock for processors is split into two groups: group-A for processors with odd-numbered ID, and group-B for processors with even-numbered ID. All of those clock traces are managed by two phase-locked-loop (PLL) modules, which are driven by a single 10MHz clock source. By using a divider and a multiplier inside the PLL, the PLL's output frequency can be adjusted between 25MHz and 400MHz. An external clock divider then converts the PLL's output into the required clock frequency.

4) CPU Utilization Measurement: There is no dedicated circuitry inside a SpiNNaker chip that provides direct measurement of a core utilization (or a CPU load). However, the System Controller is equipped with a register that contains sleep mode status flags of all ARM cores in the chip. Each core will send a STANDBYWFI (stand-by wait for interrupt) signal when it goes into low-power sleep mode. This signal will set a bit flag in that register. By counting how many times this bit flag is changed for a specific time duration, we can measure the utilization (or load) of the corresponding core.

5) Packets Traffic: Monitoring packet traffic is very useful for some programs, especially those that require reliable distributed processing. By monitoring packet traffic, a load balancing algorithm can be implemented. The SpiNNaker router has many registers for monitoring and controlling the packet routing. In our work, we are interested only on monitoring how many packets have been transmitted and how many of them are dropped. Our profiler program collects such information especially on multicast packets (MCs) and point-to-point packets (P2Ps). These two type of SpiNNaker packets are the most commonly used data packets in user applications [5].

C. Software Structure

SpiNNaker system does not have a mainstream operating system (OS) such as Linux to maintain its operation; rather, it has its own kernels that manage the entire operation of a SpiNNaker system. There are two kernels: SpiNNaker control and monitoring program (SC&MP), and SpiNNaker runtime application kernel (SARK). The SC&MP runs exclusively in core-0 of every SpiNNaker chip, whereas SARK runs in other cores. Every application program should be compiled using the SARK library, so that it can access SpiNNaker resources through the SARK kernel. Our profiler platform also uses the same mechanism; it uses SARK functionality in order to collect information for various measurements. There is also an application program interface (API) than can be used to write an application for SpiNNaker. Fig.2 shows how our profiler framework is deployed.

As seen in Fig.2, we run the profiler program in every chip. On chip (0,0), the profiler program becomes the root profiler that manages the communication with the main profiler program (with GUI) in a PC through an Ethernet link. This root profiler also manages the communication for gathering all information from other profiler programs in different chips before sending the combined information to the PC. The root profiler sends a signal as an MC packet to all profilers, which instructs them to start sending the information to the root

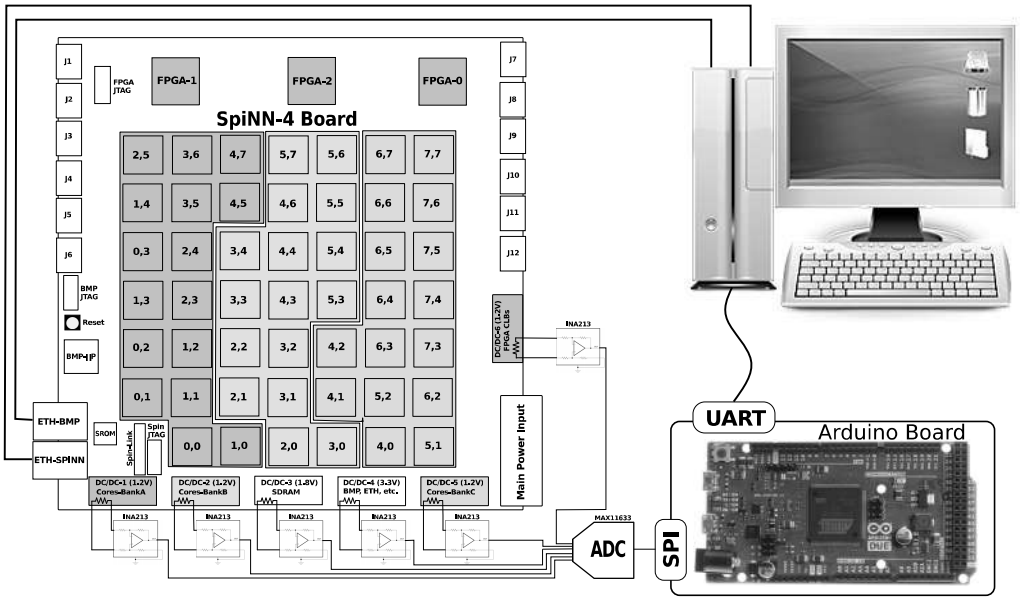

Fig. 1. The overall logical circuit diagram showing unified measurement system: an Arduino-based DAS for power measurement, and internal SpiNNaker profiler program for measuring internal state of the SpiNNaker chips.

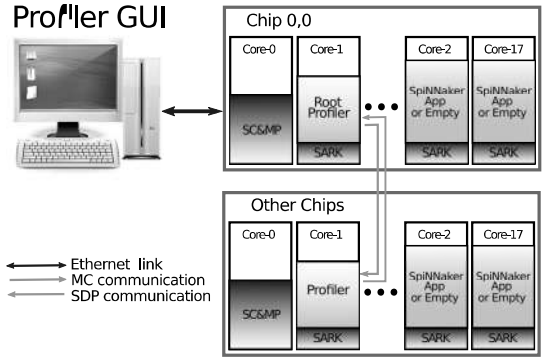

Fig. 2. Communication flow of the profiler framework. Inside every chip, a profiler program runs on core-1 and collects information from the corresponding chip. The profiler program running in chip (0,0) also behaves as the root profiler, which broadcasts a message using an MC packet periodically to all other chips. The other profiler programs will respond by sending their measurement data to the root profiler. To synchronize with Arduino data, the root profiler waits for a signal from the main profiler in the PC. This signal will be sent through the Ethernet link by the main profiler, once it gets a new data from Arduino.

profiler. Once the profiler program in other chips receives this signal, it sends the information back to the root profiler using SDP packets. The information sent by the root profiler is visualized by the main profiler in the PC.

Every SpiNNaker program in a chip can query information from the profiler in that chip for its own purpose, such as for optimizing its performance. This has been used for example in [5]. For general evaluation and monitoring of a SpiNNaker machine, our profiler framework provides visual information in a GUI (graphical user interface) that runs on a standard PC. Fig.3 shows the profiler visualization on a desktop PC.

TABLE I
SPINN-4 POWER PROFILE IN THREE DIFFERENT STATES.

Condition	Power (Watt)
Powered up, not booted	23.58
Booted, no application	27.70
Intense applications	53.65

III. EXAMPLE USE CASES

In this section, some use case examples of our profiler framework are presented.

A. Power Measurement

Measuring power consumption is a crucial aspect when developing power-aware and reliable applications. This becomes more obvious especially for embedded systems, where hardware constraints play an important role in the overall system design and performance evaluation. Recently, SpiNNaker machine has been used as a "brain" for a mobile robot [6]. The robot used in [6] is shown in Fig.4. Unfortunately, the robot was designed without measuring the real power consumption of the SpiNNaker board; hence, the robot is equipped with "plenty" of batteries to ensure operability of the robot. In future, the real power consumption by the SpiNNaker board may be taken into consideration for creating more efficient robot platforms. By using our profiler framework for power measurement, the real power consumption can be calculated precisely. Table I shows the power profile of SpiNN-4 under several conditions.

Another project that demonstrates the use of our profiler platform is presented in [5]. The authors measured the power consumption of the SpiNNaker machine that was used for

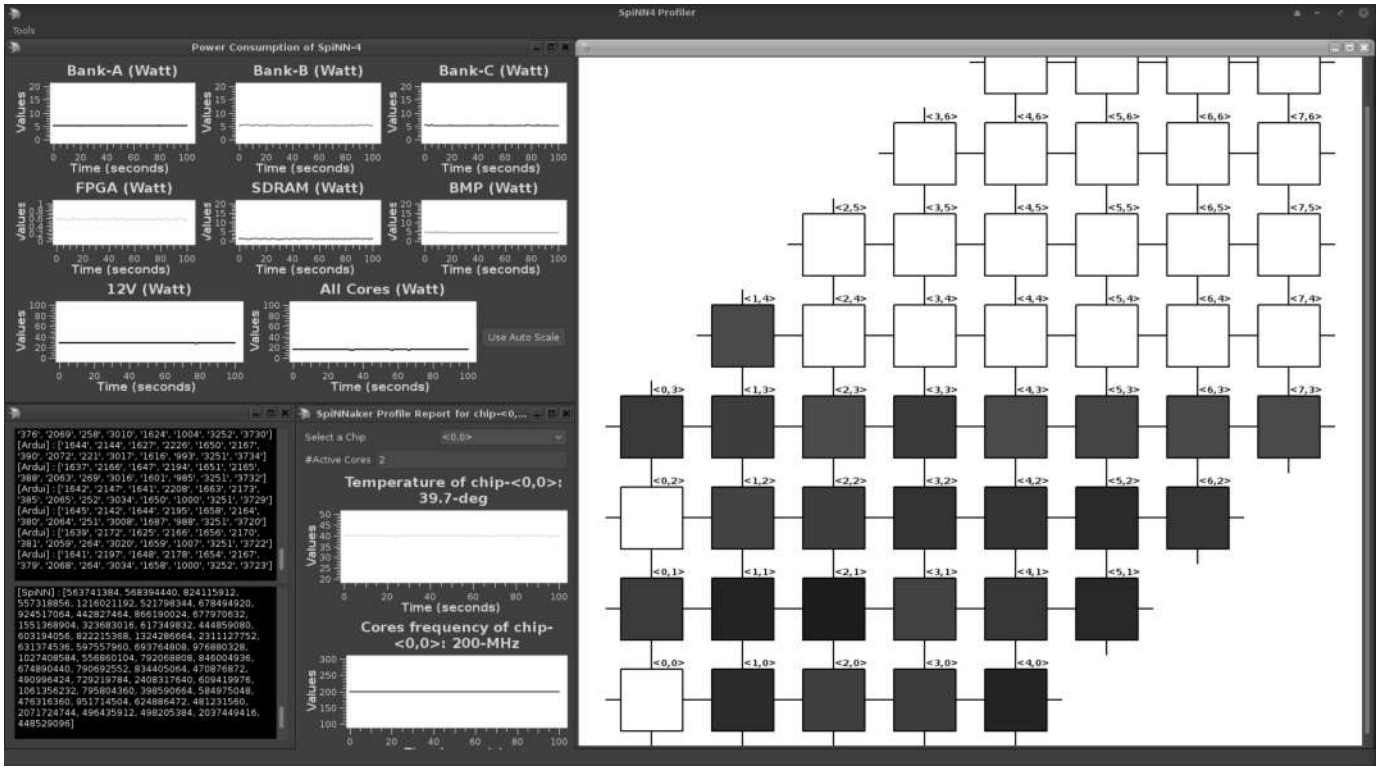

Fig. 3. The GUI of the main profiler. It is developed in Python using PyQt module. It has several sub windows that correspond to each modality in the measurement data. Shown here are the Arduino-based DAS Interface Console, Power Measurement Widget, Temperature Widget, Frequency Widget, and Network Connectivity and Activity Widget. The other sub windows are hidden: Core Utilization Widget, Frequency Selector Widget, Task Migration Widget, and the SpiNNaker Debugging Console.

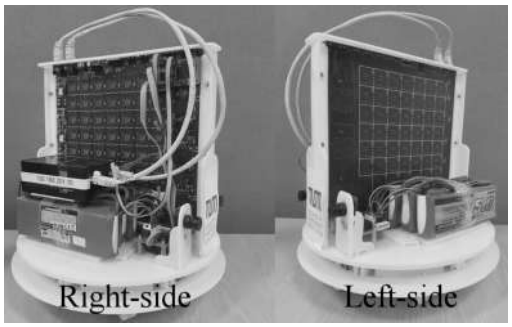

Fig. 4. SpiNNaker usage for mobile robotic research. The robot was developed by a team from Neuroscientific System Theory in Technische Universität München, Germany. The robot is equipped with six high power Li-ion batteries, much more than enough to power up the entire system.

developing an image processing application on SpiNNaker. By measuring the real power consumption, it is concluded by the author that SpiNNaker is capable of performing high resolution image processing equivalent to modern graphic cards but with much lower power consumption.

Recently, the profiler framework was also used for measuring the power consumption during a simulation of a spiking neural network (SNN). Fig.5 shows the plot generated from data produced by our profiler framework for this simulation. Initially, the SpiNNaker software developer assumed that the power should be in either a steady level or fluctuates synchronously with event processing in SpiNNaker. However, they found an interesting phenomenon where the fluctuation happens periodically every six seconds independent of event processing (shown in Fig.5 in region c). This has lead into a new investigation by the SpiNNaker software team, which is still in progress at the time of writing this paper.

B. Temperature Measurement

SpiNNaker is an example of the emerging many-core technology. Running an application on a many-core platform, especially a compute-intensive one, will easily increase the temperature of devices that eventually leads into lifetime shortage of the system (which is usually measured as MTTF - mean time to failure). Hence, thermal-aware runtime management (RTM) system has become a hot topic in research recently, and many engineers and scientists try to model the behaviour of many-core system under this control mechanism [7]. For this, we need to provide a reliable temperature monitoring system. Currently, a dynamic RTM system with predictive thermal model is under development for SpiNNaker. This RTM control system will use the temperature measurement of our profiler program. Fig.6 shows the profile of three SpiNNaker temperature sensors under normal operations. The temperature of SpiNNaker chips on SpiNN-4 can also be displayed on real time, as shown in Fig.3.

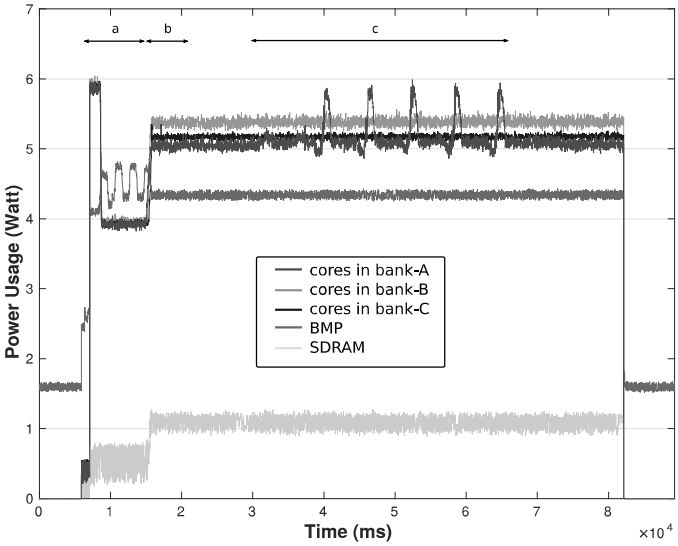

Fig. 5. An example of power measurement during SNN simulation. Three region of interests are shown: (a) during powering up the SpiNNaker board, (b) during booting up the board, (c) during running the SNN simulation. Regions a and b show SpiNNaker behavior as expected, but region c shows something new and currently being investigated thoroughly by the SpiNNaker software developer team.

C. Processor Frequency and Utilization Measurement

In addition to temperature profiling, a dynamic RTM control system will also need information about current processor frequency as well as processor loads. Our profiler platform is designed to provide such information simultaneously in real time. The RTM then will use these information to maintain the system performance at optimal point. This information can be queried directly by an application in a SpiNNaker chip from the corresponding profiler program (running in core-1), and can also be visualized in real time on the main profiler GUI. The frequency measurement plot is shown in Fig.3, whereas the processor utilization plot is shown in Fig.7.

D. Packets Traffic for Network Evaluation

As a massive many-core system, SpiNNaker has attracted researchers to deploy a distributed and parallel computing program on it using a Task Graph (TG) formalism. One particular challenge of implementing a TG on SpiNNaker is the mapping between TG nodes and SpiNNaker chips. Authors in [8] for example used an evolutionary algorithm to produce optimal mapping of a TG on SpiNNaker. To further improve the mapping result, the authors observed the number of packet drops and congestion in the network. This was done by using our profiler platform that provides information about network connectivity on the SpiNNaker board. As described in Section II-B5, the profiler program in each chip monitors packets traffic in-and-out of the chip, as well as dropped packets. As a result of using information from the profiler, a more reliable network is achieved because the TG can be remapped and the load can be balanced across the network. The networking activity of SpiNNaker chips can also be observed visually from the main profiler GUI as shown in Fig.3. The largest window in the figure shows the layout of SpiNN-4 chips and their interconnection. Each square in the diagram represents a chip, and its color represents its networking activity level.

IV. Discussion

Our profiler platform is a part of an RTM system currently under development for SpiNNaker. Initially, it has only the following measurements: temperature, frequency, and core utilization. Later on, we found that some SpiNNaker applications also require power measurement as well as network traffic monitoring. Hence, we extended the profiler program so that it covers a wider range of measurement.

As demonstrated in Section III, the profiler platform works reliably to collect and visualise various measurement data. The profiler is designed such that it can be accessed directly by SpiNNaker applications to provide various information about current SpiNNaker states, or for real-time observation via a GUI on a standard PC.

Regarding SpiNNaker power distribution modelling, currently the profiler only supports power measurement of a single SpiNNaker board. As described in Section II-A, the big SpiNNaker machine will be composed of many boards and cabinets. To build a SpiNNaker frame or cabinet, several supporting circuitry called backplane modules are required. We can estimate the power consumption of these modules per frame. Thus, by combining the backplane power model and SpiNNaker board power model, we can build the complete power model of the whole SpiNNaker machine. We regard this as an opportunity for our future work.

Another possible improvement for the profiler framework is the inclusion of a monitor program for memory usage. In PC-based operating systems, such a monitoring tool is standard practice and very useful for giving information about currently available memory. Using this tool, we can monitor the performance of a program with respect to its memory usage efficiency. The information from this tool can also be used by an RTM program in a process migration scenario, where it needs to estimate the time needed to move a process and its workspace in memory from one chip to another chip. Hence, we also regard this as another opportunity for future work.

V. Conclusion

A unified and comprehensive profiling platform for SpiNNaker has been developed, and its evaluation as well as its current use cases is presented in this paper. The profiler platform is developed to collect, represent, and visualise various machine states/information, such as machine's power distribution and consumption, chip temperature, processor clock frequency and utilization, and network traffic. Such information is valuable for observation and evaluation of running programs on the machine, as well as the health status of the machine itself. The information from the profiler can also be used for tuning the system's kernel operational parameters to maintain its performance and reliability. Many researchers and institutions have been using SpiNNaker not

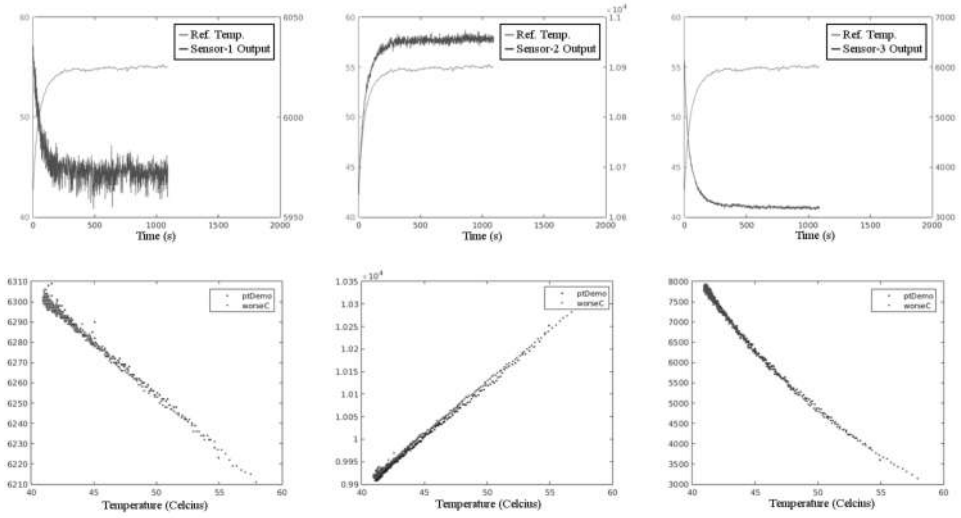

Fig. 6. The behaviour of three internal sensors of a SpiNNaker chip under normal operation. On the top row, each sensor output is plotted in the time domain alongside an external sensor reading for calibration. On the bottom row, the output of a simple mapping function between real sensor output and external sensor reading is plotted; these will give temperature readings in centigrade.

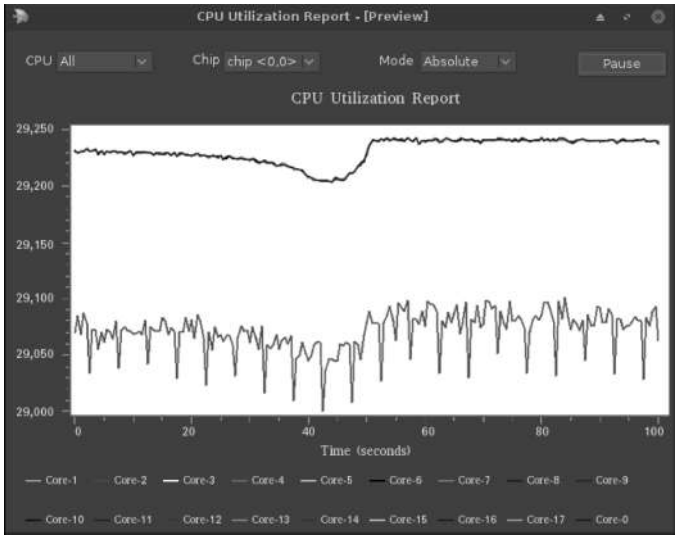

Fig. 7. A dialog window showing processor (CPU) utilization. This window actually belongs to the main GUI shown in Fig.3, but shown separately here for figure simplicity. The CPU utilization can be displayed as raw/absolute values (shown above), or relative values (by changing the mode in the Mode Menu).

only for simulating spiking neural network, but also for general purpose and energy-aware computing. Thus, the the profiler platform presented in this paper will be very useful to help SpiNNaker users and software developers in developing their program.

ACKNOWLEDGMENT

This work was supported primarily by the Engineering and Physical Sciences Research Council (EPSRC) through the Graceful project (grant EP/L000563/1). The design and construction of the SpiNNaker machine was also supported by EPSRC under grants EP/D07908X/1 and EP/G015740/1. The Arduino-based DAS was developed through the EPSRC-PRiME project (grant EP/K034448/1). Ongoing development of the software is supported by the EU ICT Flagship Human Brain Project (FP7-604102). BSB, LAP, ST and SBF receive support from the European Research Council under the European Unions Seventh Framework Programme (FP7/2007-2013) / ERC grant agreement 320689.

REFERENCES

[1] S. B. Furber, F. Galluppi, S. Temple, and L. A. Plana, "The spinnaker project," *Proceedings of the IEEE*, vol. 102, no. 5, pp. 652–665, May 2014.

[2] L. Benini, A. Bogliolo, and G. D. Micheli, "A survey of design techniques for system-level dynamic power management," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 8, no. 3, pp. 299–316, June 2000.

[3] K. Bhatti, C. Belleudy, and M. Auguin, "Power management in real time embedded systems through online and adaptive interplay of dpm and dvfs policies," in *2010 IEEE/IFIP International Conference on Embedded and Ubiquitous Computing*, Dec 2010, pp. 184–191.

[4] APT, *SpiNNaker Datasheet version 2.02*, School of Computer Science - University of Manchester, January 2011.

[5] I. Sugiarto, G. Liu, S. Davidson, L. A. Plana, and S. B. Furber, "High performance computing on spinnaker neuromorphic platform: A case study for energy efficient image processing," in *2016 IEEE 35th International Performance Computing and Communications Conference (IPCCC)*, Dec 2016, pp. 1–8.

[6] F. Galluppi, C. Denk, M. Meiner, T. Stewart, L. Plana, C. Eliasmith, S. Furber, and J. Conradt, "Event-based neural computing on an autonomous mobile platform," in *2014 IEEE International Conference on Robotics Automation (ICRA)*, 2014, pp. 2862–2867.

[7] I. Yeo, C. C. Liu, and E. J. Kim, "Predictive dynamic thermal management for multicore systems," in *Proceedings of the 45th Annual Design Automation Conference*, ser. DAC '08. New York, NY, USA: ACM, 2008, pp. 734–739. [Online]. Available: http://doi.acm.org/10.1145/1391469.1391658

[8] I. Sugiarto, P. Campos, N. Dahir, G. Tempesti, and S. Furber, "Task graph mapping of general purpose applications on a neuromorphic platform," in *Future Technologies Conference 2017 (FTC 2017, accepted)*. New York, NY, USA: ACM, November 2017.