An Open-Source Scalable Thermal and Power Controller for HPC Processors

An Open-Source Scalable Thermal and Power Controller for HPC Processors

Giovanni Bambini*, Robert Balas†, Christian Conficoni*, Andrea Tilli*,
Luca Benini*†, Simone Benatti*, Andrea Bartolini*
*_Università ALMA Mater Studiorum_, Bologna, Italy, †_ETH Zürich University_, Zürich, Switzerland
*{giovanni.bambini2, christian.conficoni3, andrea.tilli, luca.benini, simone.benatti, a.bartolini}@unibo.it,
†{balasr, lbenini}@iis.ee.ethz.ch

***Abstract*—In the last decade, high performance multi-core processor designs have followed an increase in number of cores, interfaces, heterogeneity and System-on-chip (SoC) complexity. HPC applications also require tailored chip designs with specific operating points and performance indexes. In this scenario, an advanced and configurable Power Controller System (PCS) is necessary to meet power and thermal constraints, without the necessity of static ultra-conservative margins on the operating points. In this paper, we propose an open-source PCS design, based on a parallel ultra-low power microcontroller with RISC-V cores, and an open-source software environment based on a Real-time operating system (RTOS) with a configurable Power-thermal control algorithm. Considering a 1ms control interval, the overhead of the RTOS is about 6% of the cycles in the nominal case. The control algorithm is able to limit temperature and power consumption within given bounds, while maximizing performance. The PCS is able to control up to 76 different cores/computing units with headroom for larger core counts.**

***Index Terms*—Real-time OS, HPC Processor, Power Control, Thermal Control, Scalable, Parallel microcontroller**

I. Introduction

Modern HPC processors comprise tens of cores, with vector units, specialized Hardware co-processors, and memory subsystems. Today, the OS handles power management by a dedicated driver, which selects the HW operating points based on cores' utilization. In HPC environment, the applications are composed by multiple processes running in parallel on distributed computing nodes, each featuring multi-core CPUs. Applications often use all the cores available, and request the fastest possible computation. Since the application's processes need to synchronize on barriers, ensuring the same computational performance on all the cores enables higher utilization and shorter execution time [1]. The HW selects autonomously the operating point for each computing resource leveraging on/off-chip power, voltage, and temperature (PVT) sensors.

To this aim, processors in this field embed dedicated HW resources (Power Controller Systems, PCS) to control the power consumption dynamically, to prevent thermal hazards (thermal capping), to ensure the TDP power budget (power capping), and ultimately to increase the energy efficiency. Hardware vendors use different PCS architectures: IBM Power9 processors use a heterogeneous SoC based on a PowerPC 405 and four additional General Purpose Engines[1]. ARM defines the System Control Management Interface (SCMI) standard with an M3 core[2] to implement the PCS. The firmware is based on several tasks, scheduled by a SW state machine. Both vendors release their PCS firmware open-source, but the implementation depends on close-source hardware. The PCS has to: (i) interface with several on/off-chip sensors, power management interfaces and actuators; (ii) perform complex computational tasks, like automatic controls, signal processing, optimisation and machine learning algorithms; (iii) support a large number of processing elements; (iv) interface with the OS and the board management controller (BMC) [2].

Naive thermal and power capping algorithms are not capable of delivering the desired energy-efficiency on a wide set of working conditions and workloads for the processors in this market segment [1], [3]. For this reason, advanced controllers and advanced policies are needed. More computationally-intensive tasks require more powerful microcontrollers (uC) to comply with the tight timing constraints of the thermal control in HPC. More capable uC also allow the implementation of more sophisticated software, like RTOS or parallel computation, while still decreasing the discrete-time output update interval. Shorter and more deterministic intervals increase the quality and the responsiveness of the control, the robustness of thermal and power capping, thus allowing for more aggressive control parameters and improved performance.

This work proposes a novel PCS based on an open-source multi-core architecture, namely PULP [4]. The PULP SoC has been recently compared [4] with M4 and H7 ARM cores on computationally intensive tasks, where it shows 19.6x lower latency and 8.67x higher energy efficiency. A configurable algorithm for power and thermal control is implemented relying on an open-source RTOS. By leveraging an open-source SoC, RTOS and firmware we aim to provide a flexible and powerful PCS reference design ready for the current and future HPC systems.

The controller algorithm is can manage up to 76 processor tiles with an overhead of 6% in the cycle count. The power constraint is enforced with a 500us delay considering perfect prediction of the workload, and with a 42ms delay considering high varying (every 50ms) mixed workloads, with the aid of a Recursive Weighted Least Mean Square algorithm. The thermal constraint is enforced with a 7.5°C margin and a maximum settling time of 41ms. All the results have been obtained with HW-in-the-loop simulation.

This research was partially funded by the European Union's Horizon 2020 research and innovation programme "European Processor Initiative" under grant agreement No 826647.

[1]http://developercongress2017.openpowerfoundation.org/wp-content/uploads/2017/05/Part3-On-Chip-Controller-OCC-Tutorial-1.pdf

[2]https://connect.linaro.org/resources/bkk19/bkk19-pm05/

II. Background

In this paper we consider an HPC processors composed of many processing units (cores). We assume each core to be controlled by varying the frequency of its clock, and its temperature reading is available by means of PVT sensors. We also consider the processor total power consumption to be provided from power gauges. Roughly speaking, the aim of a thermal/power control system is to dynamically choose the "best operating point" for each core[3], while keeping the entire system within its thermal, power and structural bounds expressed by the following inequalities:

$$\begin{aligned} &\sum_{i=1}^{n_s} P_i \leq P_{\text{budget}}, \qquad T_{\text{Si},i} \leq T_{CRIT}, \\ &f_{\text{min}} \leq f_i \leq f_{\text{max}}, \qquad V_{dd\_\text{min}} \leq V_{dd\_i} \leq V_{dd\_\text{max}}, \\ &\qquad\qquad \text{with: } i = 1, \dots n_s. \end{aligned} \tag{1}$$

where n_s is the number of cores, and T_{CRIT} the critical temperature threshold not to be exceeded for the silicon devices whose temperature and consumed power are denoted with $T_{\text{Si},i}$, and P_i, respectively. The core thermal behavior can be characterized by an ordinary differential equation (ODE)

$$\dot{T}_{\text{Si},i} = f(P_i, T_{\text{Si},i}, z_i) \tag{2}$$

depending on the core power and temperature, as well as other external variables collected as z_i (e.g., neighbour cores and ambient temperature [3]). Also, the core power depends mainly on its voltage and frequency, as well as other parameters we denote with w_i (such as the workload), with a map

$$P_i = g(f_i, V_{dd,i}, w_i) \tag{3}$$

Additional constraints could be added to the problem, such as one on the power consumption per chip section, differentiated by the connection to different external power lines, or cores "binding constraints". The latter refers to the need to have always identical frequencies for cores executing parallel computations, to avoid workload imbalance. These requirements can be formulated as:

$$\sum_{i=1}^{n_j} P_i \leq P_{quad\_j\_max}, j = 1, \dots, n_{quad} \tag{4}$$

$$f_{ti} = f_{tj} \text{ if } B(i,j) = 1 \tag{5}$$

where $B(i,k)$ is a $n_s \times n_s$ symmetric matrix whose entries are 0 if the cores are not subject to binding constraints, 1 otherwise.
Finally, the control strategy can be cast as an optimization problem

$$\begin{aligned} &\min_f \sum_{j=0}^{N-1} |f_t(k+j|k) - f(k+j|k)|_R^2 \\ &\text{subject to: (1), discretized (2), (3), (4) (5)} \end{aligned} \tag{6}$$

where k denotes the (discrete) time variable, N is the number of steps assumed as the horizon, $f = (f_1, \dots, f_{n_s})^T$ represent the decision variables of the problem, i.e. the frequencies to be assigned to the cores by solving the problem above with the set of target frequencies $f_t = (f_{t,1}, \dots, f_{t,n_s})^T$ and $R \in \mathbb{R}^{n_s \times n_s}$ is a symmetric positive definite weight matrix and $|\cdot|_R$ is the corresponding norm.

Problem (6) can be transformed into a convex Quadratic Problem (replacing f_i with P_i as decision variables and relying on the linearity of thermal models (2)). However, this problem cannot be solved offline, because the workload is not available for the entire time horizon and the optimization relies on the exact knowledge of the chip thermal and power models (2), (3). Moreover, even though the optimization problem could be solved on-line, it would be computationally demanding to carry out such operation on a single conventional micro-controller (the Power Controller) for a large number of cores.

For these reasons, our proposed approach is based on a sub-optimal, yet efficient two-layer algorithm[4]

III. Controller Design

The proposed two-layer strategy approach consists of:

- a **Power Dispatching Layer** which distributes the requested power P_{budget} among cores, according to power limitations and other constraints;
- a **Thermal Regulator** which allocates core frequencies (and associated power) according to values computed by the Power Dispatching Layer, and to meet the core temperature constraints T_{CRIT}

A. The Power Dispatching Layer

The first step that the Power Dispatching Layer has to perform is the power consumption estimation. We compute the estimated power consumption with equation (7), where the parameters I_{cc_i} and C_{eff_i} vary from chip to chip and depend on the current workload.

$$P_{ti} = I_{cc_i} V_{dd_i} + C_{eff_i} V_{dd_i}^2 f_i \tag{7}$$

The estimated power consumption of the chip is obtained by summing all the power consumption of the cores. The total value is then compared to the input power budget P_{budget}.

$$\Delta = \sum_{i=1}^{n_s} P_{ti} - P_{budget} \tag{8}$$

If $\Delta > 0$, an algorithm should be deployed to reduce power consumption. To this aim, a simple heuristic strategy is proposed to achieve power reduction based on the thermal room of each core ($T_{CRITi} - T_{Si,i}$). In other words, the frequency reduction is not uniform but inversely proportional to the core temperature. A naive power dispatcher algorithm would share an equal budget to all the cores. This would limit the performance of CPU intensive tasks under imbalance workload, which are the ones for which the overall performance depends mostly on core's frequency.

[3]Indeed, also the core voltage has to be suitably scaled as well. However, since core frequency and voltage values are coupled, in this work we consider frequency as the main control knob, assuming voltage is scaled accordingly.

[4]It is further to note that the binding constraint can be enforced aligning the frequencies of all the coupled cores to the one with the lower value.

Specifically, the main power dispatching steps are:

1) Compute weights to reduce cores power: $\alpha_i = (T_{CRITi} - T_{Si,i})^{-1}$
2) Evaluate and enforce binding constraint as: $\alpha_i = max\{\alpha_i, \alpha_j\}$ if $B(i,j) = 1$;
3) Normalize the weights α_i as: $\bar{\alpha}_i = \alpha_i \cdot (\sum_{i=1}^{n_s} \alpha_i)^{-1}$
4) cut cores powers as follows: $P_{ti}^* = P_t - \bar{\alpha}_i \Delta$

B. The Thermal Regulator

This layer imposes the temperature limitation based on the thermal dynamics of each core, which are disregarded at the power dispatching level. To this aim, a Proportional-Integral (PI) temperature regulator for each core is proposed. The PI law is computationally light, and, besides its tuning, takes into account the core dynamic thermal model without strongly relying on it for the control action. Furthermore, each PI is distinct and independent of others, allowing flexibility and decentralization in the code execution.

The thermal layer's philosophy is to decrease the power indication P_{ti}^* received by the Power Dispatcher layer only when the core's temperature is approaching its critical value. To this purpose, the reference input of the regulator is set to be a constant value corresponding to the temperature limit T_{CRIT} reduced by a safety margin. The PI output is saturated between $[P_{ti}^*, 0]$ obtaining a null control action if the temperature is less than the reference input value, and negative control action in case the error is negative. This action is subtracted to the target power P_{ti}^* received by the Power Dispatch layer to obtain the final output, which is then converted into the corresponding core frequency exploiting the power model (3). Clearly, this part relies on knowledge of the core power characteristic. However, uncertainties can be managed thanks to the robustness of the output feedback PI algorithm. Furthermore, online adaptation of the power model can be plugged into the strategy, exploiting power measurements available from the voltage regulators supplying the different chip sections.

C. System infrastructure

The proposed control structure requires a hard real-time deadline regarding the output of the control values (frequencies and voltages); this implies that the computation of the power controller has to be time-bounded and executed with a precise periodicity; it also requires the ability to communicate with multiple agents without interfering with the control action.

To achieve this requirements we deployed our power control firmware over a RTOS. This brings other benefits, such as: (i) scalability and portability, given by the abstraction that an RTOS provides; (ii) modularity, upgradability and maintainability of the code; (iii) capability to execute multiple functions "simultaneously" by using multiple tasks and exploiting preemptive scheduling; (iv) more flexibility and the possibility to execute code with mixed computation requirements (periodic, event-driven, soft and hard real-time).

However, implementation over an OS also poses some challenges and raises concern over the required performances (considering the characteristics of the HPC systems over which the PCS is performing its control action), and the safety and reliability of the entire system.

Item	Nominal	Worst Case
Overhead: Total Cycles	14 887	23 607
% of total cycles	5.95 %	9.44 %
Total (with Semaphores)	47 955	63 761
% of total cycles	11.24 %	15.86 %

TABLE I: Total overhead of using FreeRTOS in our Application

FreeRTOS was chosen as the real-time OS because it is a well-established RTOS with the necessary characteristics and tools to meet the above-mentioned control requirements. It is also open-source, which enabled to inspect the source code while developing, and it is aligned with the scope of the PCS project. Lastly, FreeRTOS provides a safety-critical version, called safeRTOS, which is pre-certified for all standards and can be used in medical, industrial and automotive applications.

The chosen microarchitecture is Control-PULP, which is a hardware platform based on the open-source PULP project [5].

IV. Experimental Results

The proposed PCS is tested in a Hardware-in-the-loop environment on a commercial PULP-based platform (i.e. GAP8 from GWT), with a FreeRTOS v.10.2.1 implementation. The thermal and power model computation to perform the hardware-in-the-loop test are entrusted to an STM32 F401RE Nucleo micro-controller.

A. Timings

We performed several tests to evaluate the feasibility of using a RTOS, analyze its overheads and performance. In particular, we analyzed the overhead generated from the FreeRTOS notification System, the overhead introduced by the task context switch in an RTOS environment, and the overhead of the context switch generated by a binary semaphore.

Table I illustrates the total overhead of our application in the nominal and the worst-case scenario, and the percentage of these overheads w.r.t. to the available cycles of our interval. The second part of the table shows the same indexes considering semaphores as an overhead derived from the utilization of FreeRTOS. Note that, even if the percentage of overhead cycles is acceptable if compared to the total cycles in the interval, semaphores overheads represent a significant part of the cycles occupied by our application.

B. Scalability

We ran tests to assess the scalability of our code w.r.t. the number of cores to be managed. Some parts of the code are almost invariant (the FreeRTOS overheads, the parameter adaptation, other internal management) and others are almost linearly dependent on cores number (PIDs, read/write of global core variables, power algorithm). Equation (9) shows the number of cycles occupied by our application in the periodic time interval, in function of the number of cores n_s:

$$\begin{aligned} Cycles_{Nom} &= 42685 + 1629 \cdot n_s \\ Cycles_{worst} &= 63358 + 1669 \cdot n_s \end{aligned} \tag{9}$$

From equation (9), by leaving 20% of the cycles free, and adding the cycles occupied by the sporadic overheads, the

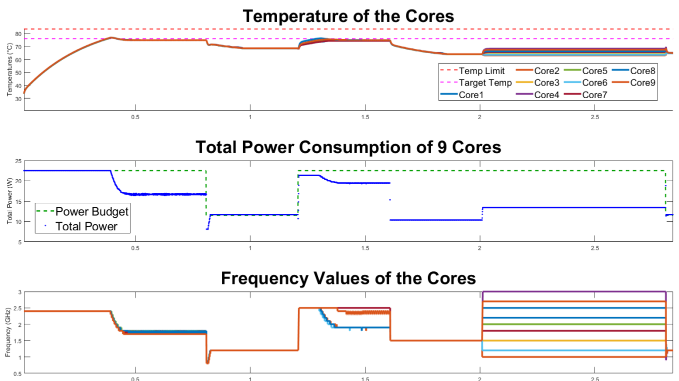

Fig. 1: Perfect Knowledge scenario test: the input commands are: high frequency w. maximum workload, a low power capping constraint, high frequency w. mixed workloads, low frequency, mixed frequencies w. DGEMM.

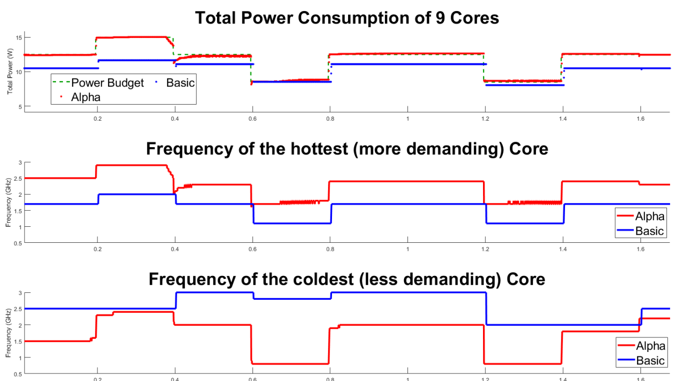

Fig. 2: Comparison test between the Alpha reduction algorithm and a basic Power Dispatching

maximum number of cores which can be controlled is 76 in the worst case scenario and 93 in the nominal case.

C. Control Performance

In the first test, we show the control capability in a perfect-knowledge scenario: the control inputs are changed every $400ms$, and the controller is perfectly aware of the system parameters, as well as the workload to be executed. From Figure 1, we can observe that both the power and temperature limits are met, while the frequency is at its maximum value. In particular, the stricter thermal limit is always respected, the longer overshoot in our test is about 41 ms (due mainly to the slow response typical of thermal systems). The frequency reduction due to thermal control is 834 MHz on average.

Then we changed the model to reflect a more realistic scenario, adding: (i) multiplicative and additive errors in the plant parameters (ii) noise in the temperature sensors and the total power measurement, (iii) additional power consumed by other components which is unknown by the controller. We also (iv) modeled mixed workloads that vary every 50ms and (v) are not known by the controller.

In this more realistic, yet challenging scenario, thermal limits are still ensured due to the feedback control (PID) properties. Instead, the total power consumption is above the power budget of the 10% for the 68% of the time. To tackle this issue, we implemented a Recursive Weighted Least Square algorithm (RWLS) which estimates the workloads parameters in the power formula and adapts them to the varying mixed-workloads values that are running in the model [6]. Using the RWLS we obtained that: (i) The temperature with the RWLS is a little lower (2.625°C in average); (ii) The power capping requirement is better met. The total power consumption is above the Power budget limit of the 10% for only the 3% of the time, compared to the 68% of time without a parameter adaptation. (i) The average frequencies are the same (we considered only the parts in which both algorithms respect the power budget limit).

We also designed a test to stress the proposed power reduction compered to a simple equal power-division algorithm. In this test, two groups of cores have really different constant workloads, while the power budget requirement is kept low and varied every 200 ms. From the graph 2 we can observe that our power reduction algorithm has better utilization of the available budget by using on average 99.5% of it, while the basic algorithmsexploits only the 87.5%. Also, with our reduction algorithm, the cores with the higher temperature (i.e. doing the most intensive workload) have an average of 700MHz higher frequency than the basic power dispatching, and this increase goes up to a 1.8GHz boost.

V. Conclusion

This paper presents an open-source versatile design for a Power Control Subsystem, with an adaptive control algorithm implemented on a Real-Time Operating System. Our framework is based on a parallel ultra-low-power microcontroller. By virtue of the efficient task manager, a 1ms base interval requires only 6% of the cycles for the overheads, under nominal conditions. To limit the maximum temperature and to control the power consumption, a re-configurable algorithm is used showing the capability to manage up to 76 tiles. The HiL emulation framework is fully implemented using off-the-shelf digital platforms.

References

[1] D. Cesarini, A. Bartolini, P. Bonfa, C. Cavazzoni, and L. Benini, “Countdown: a run-time library for performance-neutral energy saving in mpi applications,” *IEEE Transactions on Computers*, pp. 1–1, 2020.

[2] A. Bartolini, D. Rossi, A. Mastrandrea, C. Conficoni, S. Benatti, A. Tilli, and L. Benini, “A pulp-based parallel power controller for future exascale systems,” in *2019 26th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, 2019, pp. 771–774.

[3] A. Bartolini, M. Cacciari, A. Tilli, and L. Benini, “Thermal and energy management of high-performance multicores: Distributed and self-calibrating model-predictive controller,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 1, pp. 170–183, 2013.

[4] A. Garofalo, M. Rusci, F. Conti, D. Rossi, and L. Benini, “Pulp-nn: A computing library for quantized neural network inference at the edge on risc-v based parallel ultra low power clusters,” in *2019 26th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, 2019, pp. 33–36.

[5] P. D. Schiavone, D. Rossi, A. Pullini, A. Di Mauro, F. Conti, and L. Benini, “Quentin: an ultra-low-power pulpissimo soc in 22nm fdx,” in *2018 IEEE SOI-3D-Subthreshold Microelectronics Technology Unified Conference (S3S)*, 2018, pp. 1–3.

[6] R. Diversi, A. Tilli, A. Bartolini, F. Beneventi, and L. Benini, “Bias-compensated least squares identification of distributed thermal models for many-core systems-on-chip,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 61, no. 9, pp. 2663–2676, 2014.