Runtime Energy Management for Many-Core Systems

André L. M. Martins, Anderson C. Sant'Ana, Fernando G. Moraes
PUCRS University, Computer Science Department, Porto Alegre, Brazil
{andre.del, anderson.santana.001}@acad.pucrs.br, fernando.moraes@pucrs.br

***Abstract* —The on-chip power dissipation limits the number of active transistors on recent CMOS technologies nodes. Designs have to adopt power techniques as power-gating and/or DVFS for respecting the restricted power budget. As a result of the limited power budget scenario, the management of many-core systems must support power techniques for improving the energy efficiency of such systems while avoiding power emergencies. Due to the complexity of developing a power management solution, researchers adopt high-level models which abstract characteristics of real systems. On the other hand, low-level proposals are not scalable or limited to small systems. This work proposes a Runtime Energy Management (REM) for many-core systems using fine-grain DVFS as the primary power control policy. Scalability is ensured by a distributed management architecture, responsible for power monitoring and actuation on individual cores to respect the power constraints. The proposal is validated in a many-core system, described in a clock-cycle accurate model, running real applications. Results show that the proposed REM reduces energy while guaranteeing scalability.**

Keywords—DVFS; many-core system; energy management; monitoring; characterization

I. Introduction and Relate Work

On the latest electronic circuit manufacturing technologies, the growing of power dissipation prevents the full utilization of the chip, resulting in the problem known as "utilization wall" or "dark silicon" [1]. A many-core system is a SoC with dozens of Processing Elements (PEs). The management of a many-core system includes the access to input/output devices, application mapping, application migration, system monitoring, among others. The complexity of these tasks and the scalability issue is a challenge and an important research field for the academy and the industry. The utilization wall imposes a power budget and leads the system to have some policy for power control. Therefore, the system management requires power/temperature monitoring and actuation policies.

Table I summarizes the reviewed works according to a classification proposed for power management (PM). According to Table I, the main techniques used for PM include Dynamic Voltage Frequency Scaling (DVFS) [3][4][5][8][9][11], and power gating [1][4][6][7][9][10][11]. However, latency and overheads for supporting both power techniques in real systems are abstracted as well as the monitoring data generation and transmission. High-level models make assumptions related to the hardware characteristics, leading to incompatibility between the model and hardware. For instance, Hanumaiah et al. [3] report problems to embedded their power management because the target platform does not support all their power techniques. The PM targeting variability [5] and reliability [7] are also proposed using high-level models. Raghunathan et al. [6] and Pagani et al. [10] works use mapping patterns for their system that increase the hop distance between the tasks/threads of the same application. There is no discussion about performance losses or traffic increasing generate by this approach. The power management systems embedded in real platforms [2][3] and FPGAs [8] present a small number of cores, less than ten cores, with no scalability evaluation. Haghbayan et al. [4] work is close to our proposal, but larger systems require cores grouped into clusters and distributed management, and also the Authors do not consider low-level aspects, as monitoring and power data generation.

TABLE I. STATE-OF-ART IN POWER MANAGEMENT FOR MANY-CORE SYSTEMS (PG: POWER GATING, CG: CLOCK GATING)

Author	Power techniques	Design constraint	Architecture /# of cores	Design goals	Benchmark	Modeling (Exp. Setup tools)
Muthukaruppan [2]	DFS, PG, migration	Power (TDP)	Asymmetric, 5 (2 A15 cores + 3 A7 cores)	Energy efficiency	H264, PARSEC, Vision	Real platform (ARM big.LITTLE with Linux)
Hanumaiah [3]	DVFS, migration, cooling	Temperature and frequency	Homogeneous 4 (8, 16, 32 estimated)	Energy efficiency	MiBench	High-level Modeling (HotSpot, PTScalar, Magma) and real platform (quad-core Intel Sandy Bridge processor)
Haghbayan [4]	DVFS, NVT, PG	Power (TDP)	Homogeneous 12x12, 11x11, 8x8	Performance	TGG, MPEG4, VOPD, UAV	Cycle-accurate simulation (McPAT, Lumos)
Maiti [5]	Mapping, DVFS, NVT, CG, variability-aware	Throughput opt. and power (TDP)	Homogeneous 6x4	Min. energy or max. perform.	PARSEC	High-level Modeling (McPAT)
Raghunathan [6]	DFS, PG, mapping, variability-aware	Power	Homogeneous 16, 24, 32	Performance	SPLASH-2, PARSEC	High-level Modeling (McPAT)
Kriebel [7]	PG, mapping	Power (TDP)	Asymmetric, 84-481 (from a lib. of cores)	Reliability	MiBench	ISA simulation
Yu [8]	DVFS, migration	Temperature	Heterogeneous, 9	Performance	Adaptive synthetic applications	FPGA emulation (Xilinx Virtex-6, HotSpot, Xpower Analyser)
Bogdan [9]	DVFS, PG	Power (TDP)	Homogeneous, 4x4	Performance	Apache Server	Trace-driven / cycle accurate simulation
Pagani [10]	mapping, PG	Temperature	Homogeneous, 8x8	Energy efficiency	PARSEC	ISA simulation (McPAT, Gem5, HotSpot)
Zheng [11]	DVFS, PG	Area and power	Heterogeneous, 5-40	Scalability and energy efficiency	SPEC2000, SPEC2000, EEMBC	Cycle-accurate simulation + low-level analysis (LLVM, Synopsys tools)
Proposal	**DVFS, CG**	**Power**	**Homogeneous, 3x3 up to12x12**	**Scalability and energy efficiency**	**MPI apps:** DTW, MWD, MPEG4, VOPD, MPEG, Dijkstra	**Cycle-accurate simulation + low-level calibration (from the gate level)**

The *goal* of the present work is to propose a Runtime Energy Management (REM) targeting NoC-based many-core systems. The PM controls the system by monitoring the energy consumption at the PE level. The REM includes hardware and software parts. DVFS is the main actuation policy for power control. Our model considers low-level characteristics of the DVFS as latency and dc-dc converter overheads in such a way to design a realistic and accurate PM. The proposal is *scalable*, able to act in systems from dozens up to hundreds of PEs while respecting the restricted power budget of the current deep submicron technologies.

II. Reference Platform and Monitoring

Figure 1 presents the reference many-core platform. It adopts a 2D-mesh NoC, with input buffering, credit-based flow control, round-robin arbitration, and XY routing algorithm. The architecture is homogeneous, with each PE having a private memory (scratch-pad memory), a processor, a DMNI module [12], and the NoC router. Applications are partitioned into tasks, being the communication between tasks done through message-passing (MPI-like API). Shared memory modules may be added to the platform to enable other communication methods, as multithreading.

The system adopts a distributed management approach to ensure scalability, by dividing the system into clusters [13]. Every cluster contains a Local Manager PE (LMP), which manages a set of Slave PEs (SPs). SPs execute the applications' tasks, activating clock-gating when there is no task to execute. The Global Manager PE (GMP) has access to external devices (e.g. application repository). The GMP works as an LMP and distributes the applications to the clusters. The OS (Operating System) running on PEs defines their role in the system.

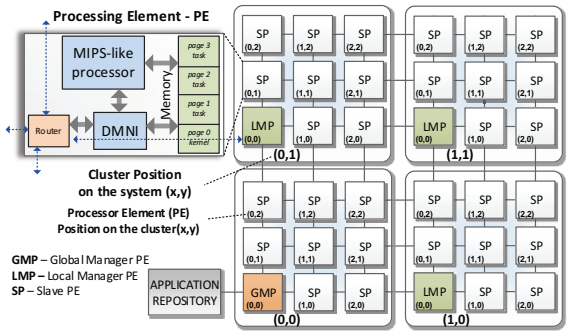

Figure 1. A 6x6 instance of the reference many-core system, with four 3x3 clusters (adapted from [14]).

At design time, a characterization method [14] provides the energy per instruction, according to the instruction type. The *energy per instruction class* – E_{class_i} (e.g. arithmetic, logical, load, store, branches) is obtained from the power measured from the netlist simulation (gate level).

The design of the hierarchical energy monitoring [14] uses the data obtained from the characterization method data and assumes the many-core model presented in Figure 1. The hardware of the PE has a set of *instruction counters* (one counter per instruction class) to accumulate the number of executed instructions. On the SPs, OS functions read the instruction counters periodically to compute the energy per instruction. Thus, the monitoring scheme computes instantaneous and accumulated energy data at the PE level. On the cluster level, the LMP receives the energy per PE data of its corresponding SPs and computes the energy per cluster. At the system level of the hierarchy, the GMP receives the energy per cluster from the LMP and computes the total energy of the system. According to evaluations presented in [14], sampling windows greater than 200,000 clock cycles ensures that 95% of the packets require 150 ticks for transmission and a worst case of 6% of timing overhead in the processing.

III. DVFS Design

The model of the frequency scaling requires adaptation on the original PE structure to cope with different frequencies. To guarantee realistic DVFS support, the model of the voltage scaling considers the hardware overheads (latency, energy), standard cells library characterized for distinct supply voltages provided by the foundry, and the delays inherent of the voltage scaling for establishing the correct DVFS protocol.

A. Frequency Scaling Model

Figure 2 illustrates the hardware modifications on the PE for frequency scaling support. The frequency scaling actuates only on the processor, memory, and DMNI. The main goal is to enable processors to work at different frequencies while the NoC transmits packets using the nominal frequency. The reason for transmitting packets at the nominal frequency is to avoid PEs running at higher frequencies stall due to PEs running at lower frequencies. The new PE has a Clock Generator (red line highlighted), which creates the *scaled frequency* from the *nominal frequency*. Frequency domain line separates the blocks of hardware running at *nominal frequency* (blue color) and *scaled frequency* (gold color).

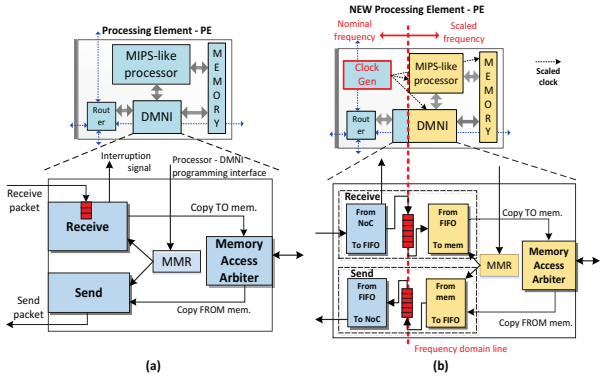

Figure 2. (a) Original PE and (b) new PE with DVFS support. The new PE has Clock Generator hardware as well as changes on the DMNI.

The DMNI synchronizes the hardware modules working at different frequencies. The original DMNI (Figure 2a) has a Send module responsible for reading the data from memory, converting it to a packet for sending to the network. The Receive module reads the packets from the NoC and copy them to the memory. The new version of DMNI (Figure 2b) has two bisynchronous FIFOs included in the Send/Receive modules to synchronize the DMNI. Both Send and Receive modules are divided into two modules, one running at the nominal frequency and the other one running at the scaled frequency. The Receive module reads the packets from the NoC at the nominal frequency, writing the flits into the bisynchronous FIFO of the receiver. If this FIFO is not empty and the memory is ready for writing, the receiver copies the packet from the FIFO to the memory, using the scaled frequency. The sending process is similar to receiving, but the data flow is in the opposite direction (read from memory, send to network).

The main hardware overheads for frequency scaling support are the Clock Generator and the bisynchronous FIFO at the DMNI. Due to the insertion of FIFOs in the new DMNI, the average execution time of the applications is penalized in 6.55%. The increasing of the power from additional hardware for supporting frequency scaling does not affect the power of the processor.

B. *Voltage Scaling Model*

A voltage regulator is an analog circuit that allows voltage scaling in a system. The method for modeling the voltage scaling on the cycle-accurate reference platform considers a set of low-level characteristics. First, standard cells characterized for 1.1V, 1.0V and 0.9V define the supply voltages supported by the system since the foundry provides liberty files only for these supply voltages (65 nm technology). Next, the selected netlist (1.1V-250MHz) is evaluated for 1.0V and 0.9V supply voltages. The minimum period to obtain a zero or positive time slack is 4.479ns and 5.229ns, for 1.0V and 0.9V, respectively.

Table II defines the DVFS protocol by linking the minimum period for scaling the voltage safely with the frequency range generated by the Clock Generator. The numbers on the yellow boxes define valid *vf-pairs*. The ascending order defines the protocol to scale the *vf-pair* down, while the descending order is the protocol to scale the *vf-pair* up. The system always starts at the nominal *vf-pair* (1.1V-4ns). If the DVFS heuristic detects higher energy values, the actuation selects the next *vf-pair* (1.1V-4.5ns), then (1.0V-4.5ns), up to reach the required energy constraint. The same procedure is followed in the other sense. Note that the DVFS protocol changes only the frequency or only the voltage, i.e., it never changes both at the same time.

TABLE II. VOLTAGE-FREQUENCY PAIRS ADOPTED BY THE DVFS PROTOCOL (65 NM BULK TECHNOLOGY).

Volt.	DVFS protocol						
1.1V						**2**	**1**
1.0V				**5**	**4**	**3**	
0.9V	**9**	**8**	**7**	**6**			
	7.0	6.5	6.0	5.5	5.0	4.5	4.0
	Period (ns)						

	Negative slack
	Non-efficient energy *vf-pairs*
	Valid *vf-pair*

The modeling of voltage scaling can be integrated into the system at the system level, cluster level or PE level. The fine-grain DVFS is a trend to overcome the limitations of coarse-grained DVFS and to maximize the energy savings [15][16]. The latency of fine-grain voltage scaling is lower than hundreds of nanoseconds [15] and allows more flexibility in controlling power. Unfortunately, the voltage scaling has an intrinsic energy overhead (around 10% of the total energy of a system at nominal voltage) by using on-chip voltage regulators [16]. Therefore, we model a fine-grain voltage scaling considering that the latency of a voltage scaling (up or down) is 100ns, and the energy overhead due to the on-chip voltage regulators corresponds to 10% of the energy per instructions values.

Once defined the DVFS scaling protocol, the PE is characterized for each voltage, considering the smallest periods (*vf-pairs* 1, 3, 6). It is not necessary to characterize each frequency individually since the power has a linear dependence to the frequency.

IV. RUNTIME ENERGY MANAGEMENT (REM)

The monitoring packets are sent by the PEs, in intervals defined at design time - *sampling window* (in clock cycles). A counter controlled by the *nominal frequency* interrupts the processor when the counter value is equal to the sampling window value. With this process, regardless the *scaled frequency* value, all PEs send their energy value to the manager PEs synchronously.

To guide the REM, it is first necessary to define the maximum energy value a PE may consume in a given period (Equation 1).

$$E_{max} = n_{cycles} * \left(\sum_{i=0}^{n_{classes}} \left(E_{class_i} \Big/ n_{classes} \right) + E_{leak} \right) \qquad (1)$$

where: n_{cycles} is the number of clock cycles executed in the sampling window, E_{class_i} is energy per instruction for a given class, $n_{classes}$ is the number of instruction classes, and E_{leak} is energy leakage per sampling window.

In fact, E_{max} is not the maximum possible energy in the sampling window since it would be obtained with the instruction class having the maximum energy value. The adoption of an average value among all instruction classes is due to the real behavior of the execution of a program trace, where a mix of all instruction types is executed, not only one instruction type.

Figure 3 shows the behavior of the instantaneous and the leakage energy when the DVFS protocol is applied on a synthetic application. These values are obtained from the monitoring packets, at the PE level. In the first half of the execution, the DVFS controller moves from *vf-pair*(1) to *vf-pair*(9), decreasing the instantaneous energy values. In the second half of the execution, the DVFS controller returns the system to the *vf-pair*(1). The leakage is less than 10% of the total energy at the nominal voltage (1.1V), reaching almost 30% of total energy at the last *vf-pair*.

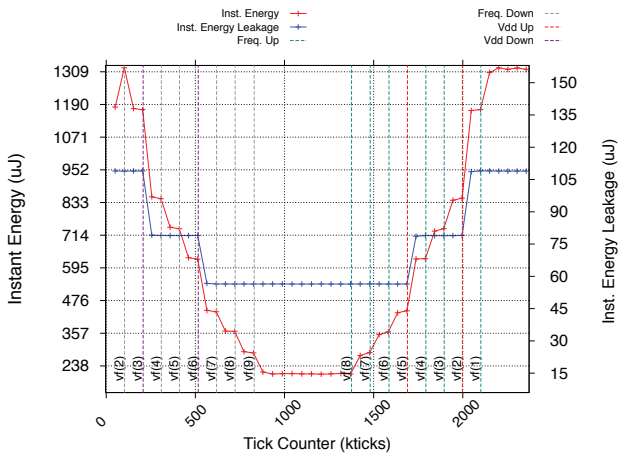

Figure 3. DVFS protocol applied to a task with maximum energy profile. The graph shows Instantaneous Energy (y-axis left legend) and Leakage Energy (y-axis right legend) as well as vertical lines highlight the *vf-pairs*. Technology: bulk 65nm.

The REM heuristic defines three energy zones, derived from E_{max}: (*i*) *hot zone*: the energy is above `VH`% from E_{max}; (*ii*) *cold zone*: the energy is below `VL`% from E_{max}; (*iii*) *warm zone*: energy between hot zone and cold zone. All these energy zones are configurable at design time.

When a manager PE receives a monitoring packet from a given SP, it verifies in which zone this SP is executing. If the instantaneous energy value of the SP is in the hot zone, a control packet is sent to the SP to scale down the *vf-pair*. In the cold zone, a control packet is sent to the SP to scale up the *vf-pair*. The control packet follows the protocol presented in Table II, rising or falling the frequency or voltage, one step at once.

The OS of the SPs has two system calls for setting a *vf-pair*: `MemWrite(DVFS,UP)`, `MemWrite(DVFS,DOWN)`. These system calls change a register mapped into the Clock Generator, which is the reference for the generation of the scaled frequency, or change the supply voltage. When the supply voltage changes, the processor holds according to the DVFS latency (100ns).

Figure 4 shows the energy profile of a PE running two tasks simultaneously. The tuple {`VH`, `VL`} is set to {85%, 60%} of E_{max}. When the simulation starts, the REM identifies a peak of energy (hot zone, red part of the graph) in the second sampling window and sends a message to scale the frequency down. As a consequence, the

energy drops, but it is still in the hot zone, making the REM change the voltage in the second actuation. The energy stays in the warm zone (yellow part of the graph) until one of the tasks terminate (400 Kticks). The REM identifies that the remaining task is working in the cold zone (blue part of the graph) and increases the voltage.

The goal of the heuristics is to avoid peaks of energy. Since the high density of transistors of the newest technologies nodes leads to the creation of hotspots in certain areas of the chip [17] our approach can move towards thermal management. Also, in a case of a restricted global power constraint, the designer can set the energy zones to keep the system below the power threshold. Only when the voltage scales down, the system saves energy. Therefore, the designer must calibrate the energy zones to allow the REM to reach the minimum voltage (sixth *vf-pair*).

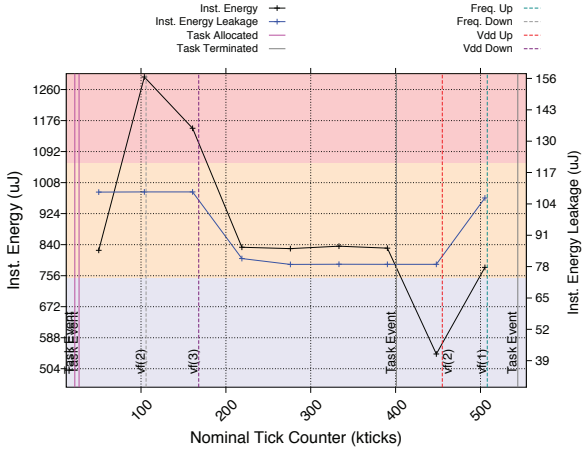

Figure 4. REM used for keeping the energy profile of the PE in the warm zone (yellow part). The tuple {VH, VL} is set to {85%, 60%} of the E_{max}.

V. Results

The benchmarks used to evaluate the proposal include (described in C language): *DTW*, *MWD*, *synthetic*, *MPEG4*, *VOPD*, *prod_cons*, *MPEG*, *Dijkstra*. The test cases run the applications in such a way that the number of tasks is equal to twice of the number of SPs, e.g., the 3x3 many-core executes 18 tasks. The platform model is clock-cycle accurate. We include in the results the 10% of the energy overhead for supporting DVFS at PE level.

Table III presents the execution time and total energy consumption for two REM configurations, compared to the baseline system. The baseline considers neither monitoring nor DVFS. Two {VH,VL} tuples are evaluated: {85%,60%} (*light REM*) and {45%,25%} (*heavy REM*) of E_{max}. The goal is to study the impact of limiting the operational zone of the system, regarding the consumed energy and execution time. The DVFS actuation is triggered when the energy of a given PE is outside of the warm zone.

TABLE III. EXECUTION TIME AND TOTAL ENERGY OF REM COMPARED TO A SYSTEM WITHOUT ENERGY MANAGEMENT (BASELINE SYSTEM).

many-core size (cluster size)	Execution Time			Total Energy		
	Baseline system	Light REM	Heavy REM	Baseline system	Light REM	Heavy REM
3x3 (3x3)	16.77 ms	19.04%	40.12%	362.03 uJ	-5.22%	-50.73%
4x4 (4x4)	17.22 ms	10.60%	34.84%	597.54 uJ	-18.12%	-55.19%
6x6 (3x3)	18.18 ms	18.11%	40.08%	1042.60 uJ	-15.74%	-51.91%
8x8 (4x4)	16.86 ms	18.35%	40.14%	966.60 uJ	-15.51%	-52.25%
9x9 (3x3)	18.20 ms	18.36%	40.02%	1377.93 uJ	-14.79%	-47.95%
10x10 (5x5)	17.87 ms	34.04%	36.73%	1686.80 uJ	-3.57%	-45.69%
12x12 (3x3)	18.31 ms	18.63%	39.74%	1859.86 uJ	-13.75%	-39.91%
12x12 (4x4)	22.86 ms	15.20%	34.09%	2383.54 uJ	-10.02%	-35.47%
	Average	19.04%	38.22%		-12.09%	-47.39%

In the *light REM*, the average energy saving is 12.09%, with an execution time overhead of 19.04%. With the *heavy REM*, the energy saving reaches in average 47.39%, with a larger penalty in the execution time, 38.22%. These results also show clearly the scalability of the proposal, whereas different system sizes generate similar penalties in the execution time and similar energy savings.

VI. Conclusions and Future Works

The Runtime Energy Management (REM) proposed in this work uses DVFS to reduce the consumed energy in many-core systems. A *significant contribution* of the paper is the closed-loop control: energy monitoring – decision (REM) – actuation (DVFS). The realistic approach of the DVFS design and the scalability of the proposal are also *original* contributions to the state-of-the-art. The monitoring infrastructure, as well as the DVFS design, enable to propose more complex REM heuristics. The DVFS technique may be employed together with task migration and scheduling policies to manage the system energy. The monitoring system may evaluate not only the energy but also the slack time of real-time applications to guide the REM.

Acknowledgments

The Author Fernando Moraes is supported by CNPq - projects 472126/2013-0 and 302625/2012-7, and FAPERGS - project 2242-2551/14-8.

References

[1] H. Esmaeilzadeh, et al. "*Dark silicon and the end of multicore scaling*". IEEE Micro, v.32(3), pp. 122–134, 2012.

[2] T. S. Muthukaruppan, et al. "*Hierarchical power management for asymmetric multi-core in dark silicon era*". In: DAC, 2013, 9p.

[3] V. Hanumaiah, et al. "*Energy-efficient operation of multicore processors by DVFS, task migration, and active cooling*". IEEE Transactions on Computers, v.63(2), pp. 349–360, 2014.

[4] M. Haghbayan, et al. "*Dark silicon aware power management for manycore systems under dynamic workloads*". In: ICCD, 2014, pp. 509–512.

[5] S. Maiti, et al. "*Process variation aware dynamic power management in multicore systems with extended range voltage/frequency scaling*". In: MWSCAS, 2015, 4p.

[6] B. Raghunathan, et al. "*Cherry-picking: exploiting process variations in dark-silicon homogeneous chip multi-processors*". In: DATE, 2013, pp. 39–44.

[7] F. Kriebel, et al. "*ASER: Adaptive soft error resilience for reliability-heterogeneous processors in the dark silicon era*". In: DAC, 2014, 6p.

[8] H. Yu, R. Syed, Y. Ha. "*Thermal-aware frequency scaling for adaptive workloads on heterogeneous MPSoCs*". In: DATE, 2014, 6p.

[9] P. Bogdan, et al. "*Dynamic power management for multidomain system-on-chip platforms: an optimal control approach*". ACM Trans. Design Aut. of Elec. Syst., v.18(4), pp. 46:1–46:20, 2013.

[10] S. Pagani, et al. "*TSP: Thermal Safe Power: efficient power budgeting for many-core systems in dark silicon*". In: CODES+ISSS, 2014, 10p.

[11] Q. Zheng, et al. "*Exploring energy scalability in coprocessor-dominated architectures for dark silicon*". ACM Transactions on Embedded Computing Systems, v.13(4), pp. 130:1, 130:24, 2014.

[12] M. Ruaro; F. Lazzarotto; C. Marcon; F. Moraes. "*DMNI: A specialized network interface for NoC-based MPSoCs*". In: ISCAS, 2016, pp. 1202-1205.

[13] G. Castilhos; M. Mandelli; G. Madalozzo, F. Moraes. "*Distributed Resource Management in NoC-Based MPSoCs with Dynamic Cluster Sizes*". In: ISVLSI, 2013, pp. 153-158.

[14] A. Martins; M. Ruaro; F. Moraes. "*Hierarchical Energy Monitoring for Many-Core Systems*". In: ICECS, 2015, pp. 657-660.

[15] W. Kim, et al. "*System level analysis of fast, per-core DVFS using on-chip switching regulators*". In: HPCA, 2008, pp. 123–134.

[16] Y. Choi, et al. "*DC-DC converter-aware power management for low-power embedded systems*". IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, v.26(8), pp. 1367–1381, 2007.

[17] M. Shafique, et al. "*The EDA Challenges in the Dark Silicon Era*". In: DAC, 2014, 6p.