A Method for NoC-based MPSoC Energy Consumption Estimation

André L. M. Martins, Douglas R. G. Silva, Guilherme M. Castilhos, Thiago M. Monteiro, Fernando G. Moraes
PUCRS University, Computer Science Department, Porto Alegre, Brazil
{andre.del, douglas.roberto, guilherme.castilhos, thiago.monteiro}@acad.pucrs.br, fernando.moraes@pucrs.br

***Abstract*— NoC-based MPSoCs are well suited for applications requiring high performance while maintaining a low-power profile. Therefore, it is important to estimate the energy consumption at early stages of the design flow due to the limited power budget imposed by the batteries. State-of-the-art proposals estimate the energy due to the NoC or the processing elements. Few works address the energy modeling and estimation for a NoC-based MPSoCs. This paper presents a method to estimate the energy/power for NoCs and processors from an RTL description, applying the proposed method to an MPSoC (36 processing elements interconnected by a 2D-mesh NoC) executing 4 real applications controlled by a multi-task operating system. The paper presents a set of results, identifying the energy due to the applications, to each element of the system, as well as the effect of two low power strategies.**

Keywords—MPSoC; NoC; modeling; low power; energy estimation

I. INTRODUCTION

NoC-based MPSoCs provide massive computing power, presenting a high flexibility and high power efficiency compared to other distinct platforms. NoC-based MPSoCs are well suited for applications requiring high performance, such as multimedia and network architectures while maintaining a low-power profile. For such applications, it is important to estimate the energy consumption at early stages of the design flow due to the limited power budget imposed by the batteries.

It is possible to estimate the energy of the applications running in the MPSoC at the transistor or gate level. However, as precise as this technique can be, this is a very time consuming task due to the number of components and the system complexity. The natural trend points out to the use of high-level models, since they abstract the platform low-level characteristics, accelerating the platform design, with a simulation speedup at least two orders of magnitude faster compared to VHDL [1].

To enable the energy estimation at high-level models it is necessary to determine the energy consumption for each MPSoC element, and the main parameters playing a role in the energy consumption. This *calibration* process allows inserting the low-level energy parameters into the high-level models, enabling an accurate energy estimation. The power dissipation in NoCs is a function of its communication load. The volume-based estimation model [2] computes the energy and power to transmit each packet from the source router to the target router, passing through *n* hops. This flow-based method is fast and simple to implement, however it does not consider the energy consumed by the routers when there is no packet transmission. The number of instructions executed at each processor defines the power dissipation in the processing elements.

The goal of this work is to present a general method to estimate the energy consumption of NoC-based MPSoCs from an RTL description, to be inserted at high-level models. Such method enables not only to execute design space exploration, but also estimates the energy cost of embedded applications. Contrary to most NoC energy estimation methods that consider the communication volume per flow, the method herein presented considers the communication volume per router, taking into account active and idle times. The processor energy estimation considers the consumption per instruction class (e.g. arithmetic, branch, load-store), leading to an accurate estimation. The model also accounts the energy spent in the wires (links between routers).

This paper is organized as follows. Section II reviews works related to energy estimation. Section III presents the MPSoC architectural model, to set the constraints to be applied to the energy model. Section IV corresponds to the main contribution of this work, presenting the energy model of the NoC routers and processors, evaluation the error induced by the model against the power evaluation at the gate level. In Section V the energy model is used in a 6x6 MPSoC, applying two low power strategies, enabling to estimate the energy consumption for a set of real benchmarks. Finally, section VI concludes this paper and points out directions for future works.

II. STATE-OF-THE-ART

Several works propose different models to estimate power and energy consumption in a NoC. Ye et al. [3] introduce a framework to estimate the power dissipated considering the energy cost to transmit a bit from an input port to an output port of a router. Chan et al. [4] use linear regression to establish a relationship between events occurring at each router component with the energy consumption. Meloni et al. [5] show a flow to formulate the power dissipation based on architectural components, implementation and traffic parameters. Guindani et al. [6] propose a model to estimate the power based on the average reception rates at each router buffer, constituted by a calibration step followed by its application. The purpose of these works is to provide methods for design space exploring, targeting energy minimization.

At the processor level, Tiwari et al. [7] describe a methodology to estimate a processor power dissipation according to the energy consumed by each instruction. Gupta et al. [8] extracted the instruction power values from a characterization done at gate level and integrated it in Instruction Set Simulator (ISS). Both works establish the processor power dissipation as a function of the energy consumed by each instruction.

Atilallah et al. [9] provides a generic model to estimate the power dissipation early in the design flow for MPSoCs. The Authors couple the power models into an architectural simulator. The processor modeling considers two states, running and waiting, being the power dissipation for these two states different. The Authors also models the cache memory, and a crossbar as the interconnection infrastructure. Experiments are presented using a H.263 coder application with systems having 4 up to 16 processors. The goal of their experiment is to evaluate the trade-off between cache sizes, execution time and total power dissipation.

To the best of our knowledge, there is a gap in the literature concerning energy models for NoC-based MPSoCs, using message passing as the communication mode between applications. The present work fulfills this gap, modeling the energy consumption for the NoC and processing elements.

III. MPSoC Architectural Model

An MPSoC consists of a set of processing elements (PEs) interconnected by a given network topology. This work adopts common features found in MPSoCs [10]:

a) each PE contains at least one processor with a private memory (scratch-pad memory);
b) communication model is message passing;
c) there is no shared memory in the system;
d) applications are modeled as task graphs;
e) a multi-tasking operating system (OS) runs at each PE;
f) a mapping function maps tasks onto PEs, being possible to have more than one task per PE.

The NoC adopted in the present work adopts 2D-mesh topology, input buffering with parameterizable depth, credit-based flow control, round-robin arbitration, and XY routing algorithm.

The method presented in the sequel may be applied to different PEs and NoCs architectures. The method to estimate power and energy is *general*, because it is based in a *calibration* process to define the energy/power values, followed by the use of the obtained values.

IV. Energy Model

This section corresponds to the main contribution of this work, the energy model for the NoC and the PE, with the evaluation of the estimation model error.

A. NoC and Links Energy Model

The process to characterize the NoC comprises four steps: (*i*) traffic generation; (*ii*) logic synthesis; (*iii*) simulation with different injection rates; (*iv*) power analysis.

The router main internal components include input buffers, crossbar, and control logic (responsible for arbitration and routing). To characterize a given component, it is necessary to have the maximum switching activity in the internal gates. Therefore, to characterize the power profile of a given router it is necessary to excite all internal components, and provide a payload with an important hamming distance between flits to induce a large switching activity in the router logic gates. Fig. 1 presents the traffic flows used to characterize router 11 (central router), in a 3x3 NoC. This router has five input buffers, each one receiving flits from a given flow.

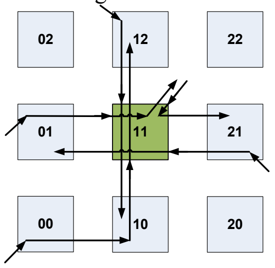

Fig. 1. Traffic flows to characterize the router consumption.

Each traffic flow source injects 1,000 32-flit packets, with a hamming distance between flits superior to 80%. The power dissipation of the router is a function of the reception rate in its buffers [6]. The method herein proposed creates 6 simulation test cases, with injection rates varying from 0% (idle) to 50% of the link bandwidth. For example, for an injection rate equal to 50%, for a 32-flit packet, each packet is injected at 64 clock cycles.

The central router is synthesized (*rc* from Cadence), for a given technology applying as constraints automatic clock gating insertion and power optimization. It is also important to consider in the logic synthesis the wire model, to obtain a correct estimation of the parasitic capacitances due to the routing. The result of the synthesis is the netlist of the central router, which replaces the original VHDL RTL router description.

The third step of the method is to simulate the NoC (*incisive* from Cadence), with the netlist of the central router and the SDF file (annotated delay data and timing checks file obtained after logic synthesis). Each simulated test case generates a TCF (Toggle Count Format) file, with the switching activity of the central router.

The fourth step corresponds to evaluating the power dissipation (*rc* from Cadence), using the TCF files. The power evaluation reports 97.35% of activity in the nets, demonstrating a correct traffic generation. The power report contains detailed power dissipation for each NoC internal component. An example of the power evaluation is summarized in Table I, with the power dissipation for the router internal components, and total average power dissipation for a 5-port router.

Table I. Average power in mW@100MHz, for each NoC component, library CORE65LPLVT (65nm), 1.2V, 25^{o}C.

Rate (%)	Buffer	Crossbar	Control Logic	5 port router
0	30.25	0.31	27.08	**205.28**
10	49.33	4.51	32.49	**322.03**
20	68.27	8.57	37.85	**437.93**
30	87.32	12.63	43.19	**554.39**
40	106.02	16.62	48.44	**668.76**
50	124.45	20.46	53.56	**781.16**

To complete the characterization process, a linear regression is made using the above results, resulting in one equation for each column of Table I (buffers, control logic, crossbar, 5 port router). The power equation for the router resulted in an r^2=0,99995, validating the linear dependency of the average power with the injection rate. Note that the buffer is the component responsible for the most power dissipation in the router. For a 5-port router, the buffers dissipate up to 80% of the total power.

In the context of MPSoCs, the following assumption is made: *when a packet is injected into the network, it is transmitted in a burst, with an injection rate equal to 100%, otherwise the link is idle, using a rate equal to 0%.* This is a correct assumption since PEs will inject packets into the NoC using a DMA module, resulting in one flit inject per clock cycle. This assumption may introduce an error at high-level estimations (e.g. TLM models) since congestion is not taking into account. This error may be neglected because the actual injection rate in MPSoCs is small (below 10%). For example, in [11] the traffic load observed by simulating the SPLASH-2 benchmarks was 0.55%.

This assumption results in two energy values: Equation (1) is active energy corresponding to a flit being transmitted through 1 buffer, while the other ones remains idle. Equation (2) is idle energy corresponding to a buffer in an idle state.

$$E_{active} = \left[(n\_ports - 1) * P_{buffer}(0) + P_{buffer}(1) + P_{crossbar}(1) + P_{control\_logic}(1)\right] * T \qquad (1)$$

$$E_{idle} = \left[(n\_ports) * P_{buffer}(0) + P_{crossbar}(0) + P_{control\_logic}(0)\right] * T \qquad (2)$$

where: n_port is the number of ports of the router, $P_{component}(0)$ the average power without traffic, $P_{component}(1)$ the average power with an injection rate equal to 100%, T the clock period.

Applying (1) and (2), using data from Table I, for a 5-port router:

$$E_{active} = 4.610\, pJ \qquad (3)$$

$$E_{idle} = 1.786\, pJ \qquad (4)$$

Therefore, to obtain the energy consumed at each router it is necessary to determine the amount of clock cycles the router is transmitting data (active), and the amount of clock cycles the router is idle. Note that (1) and (2) computes the energy per clock cycle. Therefore, the time is computed as a function of the number of clock cycles. Volume-based models, as [2], account only the flit transmission. The model herein proposed considers the whole router power dissipation, in active and idle modes. As presented next, this parcel of the energy may be higher than the active energy.

Equation (5) computes the number of clock cycles used by a given router transmitting flits. The amount of cycles is proportional to the amount of flits traversing the given router, plus the number of packets multiplied by a constant *k*, representing the number of cycles consumed to route and arbitrate a packet entering in a router, being in our NoC equal to 5 clock cycles. The remaining period is the *idle* time, equation 6, where $Cycles_{sim}$ is the total number of clock periods of the simulation.

$$Cycles_{active} = \sum flits + (\sum packets) * k \quad (5)$$

$$Cycles_{idle} = Cycles_{sim} - Cycles_{active} \quad (6)$$

Finally, equations (7) and (8) give the total energy consumption and power dissipation for a router *k*, respectively.

$$E_{rot_k} = (E_{active} * Cycles_{active} + E_{idle} * Cycles_{idle}) \quad (7)$$

$$\overline{P_{rot_k}} = \frac{E_{rot_k}}{nb\_simulated\_cycles * T} \quad (8)$$

To validate the estimation method ((7) and (8)), a traffic scenario with the following features was injected in the synthesized router: Pareto On-Off temporal distribution (burst times and idle times taken from Pareto distributions), 1.000 bursts with 34-flit packets. The Pareto On-Off was chosen since it corresponds to the behavior expected in an MPSoC, i.e., burst transmissions. The number of simulated cycles was 178,733 cycles, resulting in $Cycles_{active}$=39,000 and $Cycles_{idle}$= 139,733. According to (7), and values from (3) and (4), the estimated energy is 429,393.75 pJ (58.13% corresponding to the idle consumption). From (8) the estimated average power is 240.2431 μW. The measured average power, obtained from *rc*, was 240.26 μW, demonstrating the accuracy of the proposed method.

To compute the energy consumed in the links (wires) it is necessary to determine the wire capacitance between two routers. Using the same technology, the layout of an entire PE was synthesized. Roughly, each wire has 1 mm, corresponding to a capacitance equal to 219,4 fF. With a supply voltage equal to 1.2V and a 16-bit flit, each link consumes 4.21248 pJ (E_{link}) to be fully charged or discharged. To compute the energy at each links of a given router it is necessary to multiply E_{link} by the number of flits transmitted at each output port (except the local one, since it has shorter wires), and by the average estimated switching activity at the links (α) – equation (9). The α is assumed 0.4 (worst-case measured in real traffics).

$$E_{wire_k} = E_{link} * (\sum_{north,south,east,west} nb\_of\_flits) * \alpha \quad (9)$$

B. Processor Energy Model

The process to characterize the processor energy also relies on calibration. Initially, the instruction set is divided into classes [7], and a program is written (in assembly language) to each class in such a manner to excite all instructions of the class, the internal registers, keeping an important hamming distance among the produced results to switch the internal processor modules.

The processor energy model starts with an RTL simulation, with a testbench able to count the type and number of executed instructions, as well as to count the number of clock cycles to execute the program being simulated.

Table II presents the results obtained from the RTL simulation for a Plasma processor (MIPS ISA). The testbench generates the results for the second and third columns of the Table. The fourth column of Table II corresponds to the average CPI for each instruction class, enabling to estimate the execution time at higher abstraction layers. The fifth column is a measure of the *quality* of the program to estimate the power/energy for a given instruction class. The *branch* and *jump* classes have a smaller percentage of instructions in the programs due to the insertion of NOP instructions.

TABLE II. INSTRUCTION SET CLASSES, AND THE RESULTS OBTAINED FROM THE RTL SIMULATION.

Class	number of instructions	simulation cycles	CPI	% of inst. per class
arithmetic	73,643	73,657	1.0002	98.90
logical	108,643	108,657	1.0001	99.41
shift	46,417	46,431	1.0003	99.53
move	60,025	60,039	1.0002	98.96
load_store	70,745	137,259	1.9402	94.99
mult_div	159,757	160,021	1.0017	94.52
nop	25,457	25,471	1.0005	99.58
branch	220,017	220,031	1.0001	47.72
jump	220,030	220,031	1.0000	45.45

The processor is then synthetized (*rc* from Cadence), followed by timing simulation (*incisive* from Cadence) and power estimation (*rc* from Cadence), as in the NoC calibration process. Table III presents for each instruction class the measured average power, the total energy (10) and the energy per instruction (obtained by dividing the energy by the number of simulated instructions). For the *branch* and *jump* classes the energy is computed considering the energy of the NOP instructions.

$$E_{class} = P_{class} * cycles * T \quad (10)$$

where: P_{class} is the average power for a given instruction class (second column of Table III), *cycles* the simulated cycles (third column of Table II), *T* the clock period.

TABLE III. AVERAGE POWER IN MW@100MHZ, FOR EACH INSTRUCTION CLASS, LIBRARY CORE65LPLVT (65NM), 1.2V, 25°C.

Class	Average power (mW)	Total Energy (nJ)	Energy per inst. (pJ)
Arithmetic	2.605	1,918.76	26.05
Logical	2.269	2,465.43	22.69
shift	2.175	1,009.87	21.76
move	2.110	1,266.82	21.10
load_store	2.293	3,147.35	44.49
mult_div	2.263	3,621.28	22.67
nop	1.467	373.70	14.68
branches	3.114	6,851.77	31.24
jumps	1.851	4,072.77	20.30

Equations (11) and (12) give the total energy consumption and power dissipation for a processor, respectively.

$$E_{processor} = \sum_{i=0}^{n\_classes} nb\_instr_i * E_{class_i} \quad (11)$$

$$\overline{P_{processor}} = \frac{E_{processor}}{\sum_{i=0}^{n\_classes} nb\_instr_i * CPI_{class_i}} \quad (12)$$

where: nb_instr_i is the number of executed instructions for a given class, E_{class} is the energy per instruction for a given class, CPI_{class} is the CPI for a given class.

To validate the estimation method different benchmarks are simulated (Table IV), and the measured power is compared to the estimation model using (11) and (12). The execution time estimation, based on the CPI per instruction class, resulted in an error inferior to 3%. Table IV shows that the energy estimation model is accurate, with an error smaller to 10%. The observed differences come from: (*i*) switching activity used to model the energy per instruction is not the same of the real applications; (*ii*) in real applications there are data dependencies, inducing pipeline stalls (affecting the execution time and the energy consumption).

TABLE IV. PROCESSOR ENERGY ESTIMATION ERROR.

Becnhmark	Measured avg power (mW)	Estimated energy(nJ)	Estimated power (mW)	Error (%)
insert sort	2.236	4,185,267	2.39	6.98
bubble sort	2.548	1,716,722	2.42	-5.14
fft	2.327	838,921	2.25	-3.36
matrix	2.214	6,734,430	2.25	1.77
binary search	2.218	1,712,422	2.34	5.71
compress	2.218	4,398,619	2.29	3.27
usqrt	2.407	2,013,805	2.40	-0.11
factorial	2.093	7,404,168	2.24	6.83
switch-case	2.236	5,626,256	2.26	0.96

V. MPSOC CASE STUDY

A clock-cycle accurate SystemC RTL model describes the MPSoC [12]. The simulation speed is ten times faster than an RTL VHDL, enabling to evaluate a large set of real benchmarks. Four real applications execute simultaneously in a 6x6 instance of the MPSoC MPEG (5 tasks), DTW (6 tasks), Dijkstra (6 tasks), multispectral image analysis (14 tasks). The execution time of this scenario is 10.5 minutes, in an 8-core Intel Xeon 2.93 GHz CPU, with 32 GB RAM.

The MPSoC design adopts two low power strategies. At the PE level, the clock of the processor is disabled when there is no task to execute. When a given processor receives a task execute, the network interface detects the mapping request and the processor clock is activated. When the task finishes its execution, the clock tree of the processor is again disabled. Using this approach, the only power dissipated by the processor in idle mode is due to the leakage current, equal to 0.02 mW. At the NoC router, two frequencies are used, 100 MHz when there are flits to be transmitted, and 10 MHz when the router is in idle mode. The frequency switching is controlled at the input buffers, by signaling incoming flits to raise the router frequency. Such approach enables to reduce the consumed energy in idle mode.

Table V presents the energy spent by each processor of the MPSoC (each element of the table corresponds to a PE). Yellow PEs correspond to manager processors, responsible to map applications and monitoring functions. Their clock is not stopped. White PEs contain processors that did not receive any task to execute. They execute the boot process by the OS, and their clock tree is disabled. Green PEs contain processors executing applications' tasks. With such results, it becomes possible to estimate the energy (and power) due to each application executing in the MPSoC. Applications consumed respectively: DTW (label A) 149.5 μJ, MPEG (B) 193.16 μJ, multispectral image analysis (C) 25.02 μJ, Dijkstra (D) 259.26 μJ.

TABLE V. ENERGY SPENT IN THE 36 PES, IN UJ (100MHZ). LABELS A TO D CORRESPONDS TO THE PES EXECUTING THE APPLICATIONS.

X \ Y	0	1	2	3	4	5
5	1.41	1.41	1.41	150.15 B	1.41	1.41
4	35.01 A	1.41	1.41	35.03 B	7.98 B	1.41
3	171.66	79.53 A	34.96 A	171.69	1.41	1.41
2	3.89 C	9.80 C	2.88 C	82.72 D	1.41	1.41
1	3.69 C	2.96 C	1.41	93.32 D	1.41	1.41
0	163.48	1.80 C	1.41	171.69	83.22 D	1.41

Table VI presents the energy spent by the NoC routers (each element of the table corresponds to a router). This table shows that:

i. The energy spent by each router is a function of the number of buffers: white routers have 3 buffers, consuming 1.24 μJ, except the lower-left router which transmit flits to other routers; blue routers have 4 buffers (1.57 to 1.67 μJ); green routers have 5 buffers (1.89 to 2.01 μJ).

ii. The injection rate is small, typically 5%, explaining the small variation observed in the routers' consumption. Such small injection rate is due to the behavior of the benchmarks since most part of the time PEs are executing and not communication. A second reason explaining such small traffic comes from the absence of shared memories.

iii. The PEs consumes up to 100 times compared to the routers. In such experiment, the energy consumed by the NoC corresponds to 4.6% of the total energy consumption.

TABLE VI. ENERGY SPENT IN THE 36 NOC ROUTERS. IN UJ (100MHZ).

X \ Y	0	1	2	3	4	5
5	1.24	1.57	1.57	1.62	1.57	1.24
4	1.61	1.90	1.89	2.01	1.94	1.57
3	1.61	1.98	1.94	1.92	1.90	1.57
2	1.60	1.91	1.91	1.95	1.90	1.57
1	1.61	1.92	1.91	1.98	1.90	1.57
0	1.38	1.67	1.64	1.63	1.60	1.24

The energy consumption in the NoC links (wires) was small, being the worst-case consumption equal to 0.0402μJ. Despite the small consumption observed in the wires, it is important to evaluate the links with the highest loads since these links will be more susceptible to aging effects as electromigration.

Table VII evaluates the impact of the adopted low power (LP) strategies. At the processor level, the total energy consumption had a reduction of 65% and at the NoC level a reduction of 90%. Without the LP techniques, all processors execute almost the same number of instructions (small value of *std dev*), while with the low power techniques only the processors with tasks execute instructions (higher value of *std dev*, Table VII). The non-LP NoC consumed ten times more compared to the LP NoC, due to relationship between the active and idle frequencies (10 and 100 MHz respectively).

TABLE VII. ENERGY REDUCTION (UJ) USING THE LOW POWER TECHNIQUES.

	Processors			NoC		
	Total	Average	Std. dev.	Total	Average	Std. dev.
Without LP tech.	3,777.2	128.3	18.0	604.7	16.8	2.2
With LP tech.	1,329.4	36.9	58.9	61.5	1.7	0.2

VI. CONCLUSIONS AND FUTURE WORKS

This paper presented a general method to define the energy/power parameters for the NoC and processors, applying these to a high-level model. The obtained results showed that: (*i*) it is possible to estimate the energy per application; (*ii*) simple low power strategies brings important energy savings; (*iii*) most of the consumed energy comes from the processors (roughly 90%); (*iv*) even if the NoC is underused (typically the link loads are below 5%), the NoC allows parallel transactions, short wires (1 mm), being scalable compared to busses; (*v*) the energy consumed in the wires is small (around 1%).

Nonetheless, several directions for future works are identified: (*i*) model the consumed energy due to local memories, DMA and NI modules; (*ii*) model the effect of congestion in the NoC when two PEs tries to communicate with the same target PE; (*iii*) compare the MPSoC estimated energy at higher abstraction levels to a gate-level estimation (feasible only for small MPSoC instances); (*iv*) extend the method by including the back-end step of the synthesis.

ACKNOWLEDGMENT

The author Fernando Moraes acknowledges the support granted by CNPQ, processes 472126/2013-0 and 302625/2012-7; and CAPES process 708/11.

REFERENCES

[1] Petry. C.; et al. *A Spectrum of MPSoC Models for Design and Verification Spaces Exploration*. In: RSP, 2012, pp. 30-35.

[2] Hu. J.; Marculescu. R. *Energy-aware mapping for tile-based NoC architectures under performance constraints*. In: ASP-DAC, 2003, pp. 233-239.

[3] Ye. T.; Benini. L.; De Micheli. G. *Analysis of power consumption on switch fabrics in network routers.* In: DAC, 2002, pp. 524–529.

[4] Chan. J.; Parameswaran. S. *NoCEE: energy macro-model extraction methodology for network on chip routers.* In: ICCAD, 2005, pp. 254-259.

[5] Meloni. P.; et al. *Area and power modeling for networks-on-chip with layout awareness.* In: VLSI Design. Article ID 50285. 2007. 12 pages.

[6] Guindani. G.; et al. *NoC power estimation at the RTL abstraction level.* In: ISVLSI, 2008, pp. 475–478.

[7] Tiwari. V.; Malik. S.; Wolfe. A. *Power analysis of embedded software: a first step towards software power minimization.* IEEE Transactions Very Large Scale Integration Systems, v.2(4), 1994, pp. 437–445.

[8] Gupta. T.; et al. *High level power and energy exploration using ArchC.* In: SBAC-PAD. 2010. pp. 25–32.

[9] Atitallah. R.; Niar. S.; Dekeyser. J. *MPSoC power estimation framework at transaction level modeling.* In: ICM, 2007, pp. 245–248.

[10] Garibotti. R.. et al. *Simultaneous Multithreading Support in Embedded Distributed Memory MPSoCs.* In: DAC, 2013, 6p.

[11] Kao. Y.; Yang. M.; Artan. S.; Chao. H. *CNoC: High-Radix Clos Network-on-Chip.* IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, v.30(12), 2011, pp. 1897 - 1910.

[12] Carara, E.; Oliveira, R.; Calazans, N.; Moraes, F. HeMPS - a Framework for NoC-based MPSoC Generation. In: ISCAS, 2009, pp. 1345 - 1348.