System Level Analysis and Accurate Prediction of Electromigration

Tushar Gupta, Clément Bertolini, Olivier Héron and Nicolas Ventroux
CEA, LIST, PC94, F-91191 Gif-sur-Yvette, France
email: {first.last}@cea.fr

Thomas Zimmer and François Marc
Université Bordeaux I, IMS, F-33405 Talence, France
email: {first.last}@ims-bordeaux.fr

ABSTRACT

A single chip or a system can have more than billions of transistors, and billions of via connected through miles of interconnections. This paper aims at analyzing dynamic variations and then hard failures due to electromigration at functional level and to estimate the accuracy of the reliability aware *ArchC* based processor simulator. We use this simulator to provide the cumulative failure rate for a processor simulated at functional level, at 373MHz and 1.21V. The simulations are assumed to be under consideration of ideal environment, with no humidity and no process variability.

1. INTRODUCTION

At different levels of abstraction, there are different trade-offs to calculate power consumption and predict reliability. We get more accurate data at lower level of abstraction than higher. But the simulation is faster at higher level of abstraction. Multi-Processor System-On-Chip (MPSoC) are complex digital circuits but are very attractive for embedded computing intensive applications. Such circuits are composed of up to hundreds of processor cores, memories and interconnect. Their design space exploration (memory sizes, processor pipeline depth, interconnect bandwidth, task scheduling, etc.) for performance or power consumption objectives requires fast and accurate simulators. Performance, power and temperature modeling and simulations for MPSoCs are still subject to intensive research works. We have many kinds of failure mechanisms that may result in intermittent and permanent errors in integrated circuits (IC). The main ones are electromigration (EM) in interconnect, hard/soft oxide breakdown, hot carrier injection (HCI), and bias temperature instability (BTI) in MOS transistors. These failure mechanisms are still extensively studied at the transistor level, and manufacturers provide technology dependent parameters for each failure mechanism **[7]**. Designers need to verify if a design is robust and can handle memory sizes, task scheduling and others for performance and reliability. Relatively to other works related to reliability simulation at front-end such as **[6]**, the reasons we develop a reliability aware ArchC based simulator are: 1. Need of speed during simulation: the processor lifetime reliability was simulated at cycle accurate level (pipeline step) which was too slow for MPSoC simulation, 2. Need of a powerful language description for processor cores at functional level (*ArchC*) with lifetime reliability evaluation capabilities to be readily integrated in an MPSoC simulator and 3. Need to distinguish the effect of different benchmarks on lifetime reliability of the processor and explore the effect of different task scheduling techniques in an MPSoC, very early in the design flow.

Starting point of our work to develop this reliability simulator are (i) *PowerArchC* **[5]**, an enhanced *ArchC* **[2]** based ISS that embeds block-level power estimation capabilities at functional level, (ii) *RTME*, a methodology to get failure models and a reliability simulator at block level, and (iii) *Hotspot* **[3]**, a block-level temperature simulator. The technical contribution of this paper is a trace-based tool chain (power-temperature-reliability) – that is fully parameterized – for exploring reliability in a MIPS processor, at functional level. In this reliability simulator, we can plug any technology library from manufacturers, packaging library and failure library (typically failure models). Reliability is expressed as the Cumulative Failure Rate (CFR) over time for each digital block and each failure mechanism **[1]**. This simulator could highlight the main failure detractors and the weak parts of the design that are prone to these detractors.

In this paper, we show how to use this tool chain at functional level for exploring the effects of different benchmarks for different power values for 3 different floorplans at functional level. Results will be derived for a MIPS processor in TSMC 40nm. Section 2 will motivate our methodology. Section 3 will present our methodology to estimate power consumption and reliability at functional level. We apply the proposed methodology to the MIPS case study and provide results in Section 4. Finally, Section 5 concludes the paper.

2. RELATED WORK

There is a strong relationship between power consumption (both dynamic and static), temperature, environmental conditions (humidity, and ambient temperature), process variations and operating conditions (operating supply voltage and frequency) causing dynamic variations, then timing violations and catastrophic failures (permanent faults) in an integrated chip. Various authors explained this relationship and discussed that most system-level (one block or a complete processor) reliability models are empirical models which can benefit greatly from calibration and validation.

Reliability simulators show the increase in speed and decrease in accuracy as the abstraction level goes from device to gate and functional. BERT, the Berkeley's reliability simulator, is a well known academic example at transistor level. Srinivasan et al. **[8]** presented RAMP (Reliability Aware Microprocessor) is a methodology for lifetime reliability analysis for microprocessors at micro-architecture level and performed dynamic reliability management (DRM) using this methodology. RAMP assumes a uniform device density over the chip and an identical vulnerability of devices to failure mechanisms. Later, other authors such as **[6]** introduced a structure-aware model that takes the vulnerability of basic structures of the micro architecture (e.g., register files, latches and logic) to different failure mechanisms into account. They have provided methodologies that use DTM and DPM to improve the reliability of MPSoCs, but they only assume the failure rate of the circuit to be dependent on its instantaneous behavior and to be independent of circuit usage in the past. Recently, system level simulators **[1]** presented a solution for lifetime reliability evaluation of processor-based SoCs using state of the art power and temperature simulators. They take aging effects into consideration as compared to **[6]**. In a recent paper **[9]**, authors also provided many details about mathematics of calculating aging rate – a new reliability metric – but only numerical examples for electromigration. The integration of simulators in a real MPSoC design flow with different technology libraries is not yet addressed. We provide a methodology to calculate the cumulative failure rate for a failure mechanism of a processor at functional level. The results are less accurate than transistor or gate level reliability simulators, but still close to the real world. Actually, the simulator enables to change technology libraries, packaging libraries, and failure libraries. Designer at very early stage of design can perform small or big changes and analyze their effects on reliability of the processor. The methodology we propose is ready to be applied to MPSoCs.

3. METHODOLOGY

Our reliability simulation methodology is illustrated in Figure 1 and is explained below. To estimate the reliability of a processor at functional level, we need power and temperature values at functional level, as well. For temperature, we need power consumption values, packaging characteristics and the processor floorplan. Power at functional level is obtained through Instruction Level Power Characterization (ILPC) **[5]** performed with a power simulation tool applied to a gate level description of the processor. Since we only simulate the behavior of instructions at functional level, a first step is to model the power contribution of each instruction. As shown, in the left part of Figure 1, the "RTL design" of the processor corresponds to the "functional descriptions" (micro-architecture and instruction set architecture), made in *ArchC* language in Figure 1 **[2]**. From that, a synthesis tool like "Design Compiler" (Synopsys) generates a "gate level design", based on a chosen "technology library" (e.g. TSMC). To obtain power information at functional level, "ILPC" at gate level is performed with simulation tools like "ModelSim" (MentorGraphics) and "PrimeTime" (Synopsys). The latter can provide an accurate power consumption of each gate and each digital block, at each circuit clock cycle and for different voltage and frequency conditions ("V-F"). We design a parser that outputs the average power consumption of each instruction from processor power and program traces provided by the simulation tools. We took an assumption to achieve the "Power model" that is we do not consider all possible combinations of instructions and operands.

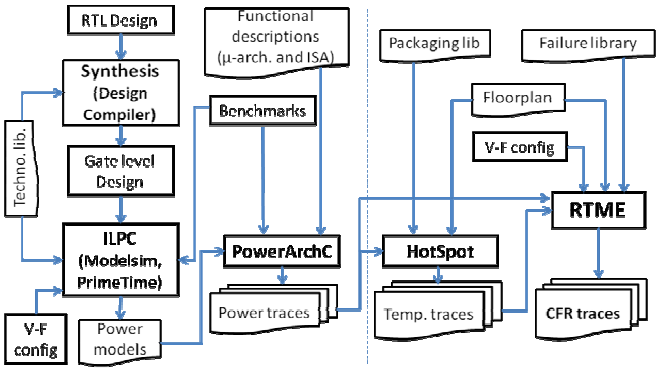

Figure 1. Reliability simulator Methodology

"Power model" is next used to back-annotate the ISS generated by *ArchC*, with power values for each simulated instruction. The behavior description of the instruction now contains a variable that points to the corresponding instruction in the power model. The ISS is now able to output both instruction and "power traces" and total consumed energy and power of the executed benchmark. The modified ISS version with power capabilities is called *PowerArchC* **[5]**. A "Power model" refers to chosen operating condition (Voltage, Frequency) and characterization campaign (benchmarks). In Figure 1, the campaigns are made with *MiBench* benchmarks. As shown in the right part of Figure 1, temperature traces are obtained using the tool named *HotSpot* **[3]**. Using a "floorplan", a "Packaging library" and "power traces" of the whole processor, *Hotspot* can derive steady and transient temperature values of each block of the "floorplan". More details are given in **[1]**. We use *RTME* (Real-Time MTTF Evaluation) which is a simulation tool we developed for predicting Cumulative Failure Rate (CFR) – reliability metric – of different blocks of the "floorplan" of an integrated circuit. It is capable to compare aging behavior for different benchmarks and architecture choices, not bound to any specific technology and power and temperature simulators. For now, it reads "V-F configuration" and "power and temperature traces" of the floorplan blocks for different failure mechanisms. CFR represents the cumulative failure rate over the time of a failure mechanism and is computed as follows:

$$CFR_x(n,b) = \sum_{a=1}^{n} \lambda_x(a,b) * t_a \quad (1)$$

where 'a' is the time step of duration 't_a', 'λ' is the current block failure rate at time 't_i', 'n' is the total number of steps, 'b' is the block reference and 'x' is the failure mechanism reference. In this paper, our failure library considers one failure mechanism: EM.

We now explain how the block failure rate is modeled from the knowledge of physics and failure models of EM at device level. EM is considered to be the result of momentum transfer from the electrons, which move in the applied electric field, to the ions which make up the lattice of the interconnect material. When electrons are conducted through a metal, they interact with imperfections in the lattice and scatter. Scattering occurs whenever an atom is out of place for any reason. Thermal energy produces scattering by causing atoms to vibrate. This is the source of resistance of metals. The higher the temperature, the more out of place the atom is, the greater the scattering and the greater the resistivity. Joule heating is proportional to the square of current density, the current crowding effect leads to a local temperature rise around the void that in turn further accelerates the void growth. The whole process continues till the void is large enough to break the line. The failure rate for EM of a single wire or via or contact is given using the well-known Black's law:

$$\lambda_{EM} = \frac{j^2}{A_0} * e^{-\frac{Ea_{EM}}{k*T}} \quad (2)$$

Where 'j' is the instantaneous current density that flows in the item, 'A_0' is the combination of technology-dependent constants and the second part is from *Arrhenius* equation, where, 'E_a' is activation energy, 'k' is *Boltzmann* constant and 'T' is the junction temperature. The instantaneous current density 'j' that flows through a via/contact during a clock cycle can be expressed as follows:

$$j = \frac{i}{S} = \frac{C_{ox} * V_{dd} * f}{S} \quad (3)$$

Where 'i' is the instantaneous current, 'S' is the wire/via/contact cross-section area, 'V_{dd}' is the operating voltage and 'f' is the operating frequency. From equation (2), we derive the failure rate of a digital block composed of 'N' items by applying the following assumptions: *(i)* the failure rate of a block is a constant value during a cycle 't_i'. Hence, the Mean Time to Failure (MTTF) is the reciprocal of the failure rate; *(ii)* the equivalent failure rate of 'N' components is based on a Series model in which the first device failure always causes the block failure; *(iii)* EM is predominant in contacts/vias located in power rails and CMOS gate outputs; we neglect the effects of EM in inter-gate wiring *(iv)* same transistor geometries and doping with same via/contact cross-section areas and hence same capacitances; *(v)* block area is proportional to the number of transistors and hence via/contact (N); *(vi)* we assume that all CMOS gates have an identical fanout; *(vii)* and the number of switching transistors is proportional to the switching probability. Therefore, the instantaneous current density (Black's law) of a block is assumed to be replaced by the mean current density bringing into play the switching probability, the mean dynamic power and the number of components. From that, the failure rate of block 'b' for EM at time 't_a' can be expressed as follows:

$$\lambda_{EM}(a,b) = N * \frac{[\alpha(a,b) * j(a,b)]^2}{A_0} * e^{-\frac{Ea_{EM}}{k*T(a,b)}} \quad (4)$$

Where 'α' is the input switching probability at time 't_a' and 'N' is the total number of via/contact of the block. Here, an identical current

density flows through all the via/contact at each cycle and the value is proportional to 'α'. Similarly, the instantaneous dynamic power 'P_{dyn}' of a digital block can be expressed as follows:

$$P_{dyn} = C_{block} * V_{dd}^{\ 2} * \alpha * f \quad (5)$$

Where 'C_{block}' is the equivalent gate capacitance of the block and C_{block}= N*C_{ox}. Hence, from equations (3) and (5) we can derive the following relation,

$$j = \frac{P_{dyn}}{V_{dd} * S * N * \alpha} \quad (6)$$

And from equations (1) and (5), we can derive the failure rate of the block as an expression of the dynamic power of the block and the input switching probability,

$$\lambda_{EM}(a,b) = \frac{P_{dyn}(a,b)^2}{B_0} * e^{-\frac{Ea_{EM}}{k*T(a,b)}} \quad (7)$$

Where $B_0 = A_0 * V_{dd}^{\ 2} * S^2 * N$. Therefore, the CFR for EM of a digital block 'b' at time 'n' can be derived from (6) and (1) as follows:

$$CFR_{EM}(n,b) = \sum_{a=1}^{n} \frac{P_{dyn}(a,b)^2}{B_0} * e^{-\frac{Ea_{EM}}{k*T(a,b)}} * t_a \quad (8)$$

EM results depend on power consumption and temperature variations. Similarly to power and temperature, CFR is computed at each instruction execution i.e. each time step of *PowerArchC* simulator. Note here that we do not yet consider the static power contribution. Parameter 'n' of CFR formula is so equal to the number of instructions in the executed benchmark. Parameter 't_a' is a constant value (let say 'T') related to the frequency at which power and temperature are recorded. The 'a^{th}' line in a power or temperature trace corresponds to the value measured at time step'(a - 1)*T'. Power and temperature are assumed to remain constant during time 'T'. At each time step, *RTME* produces a CFR value for each failure mechanism and for each block of the chip floorplan.

4. RESULTS

We implement the methodology discussed in Section 3 on a MIPS 32-bit processor. We use *ArchC v2.0* to generate the MIPS ISS that supports the full *R3000* ISA. We synthesize an open source RTL description of MIPS (HMC-MIPS) with *TSMC 40nm* standard cell libraries for typical case scenario. ILPC and reliability simulations are performed with the following operating conditions: 373MHz and 1.21V @25°C. We assume an ideal environment with no humidity and no process variability and we consider results for the whole processor without system memories. However, we design a detailed floorplan of MIPS, as described in **[1]**, composed of 7 blocks denoted as follows: *fetch, decode, execute, writeback, memory, control and clock*. ILPC is performed with a program built randomly: instruction opcode selection and execution order are random (script in *Perl*). The number of instructions and operands in the bench vary every time. The program executes approximately 300,000 instructions. Due to instruction cache and time limitations, the program is split into 20 sub-programs, each composed of 16000 instructions. We provide one final power model with more accuracy compared to the model used in **[5]**.

Table 1 shows the power simulator performances and accuracy. Third column compares the execution time of a benchmark at RTL (PrimeTime) and at functional level (PowerArchC). The last column shows the percent error, where Dynamic power from RTL provides the theoretical value and Dynamic power from PowerArchC is calculated using power model obtained with the help of the random program.

Table 1. Power simulator performance and accuracy

Benchmark	# instr.	Simulation time (min)		%error
		PowerArchC	PrimeTime	
Qsort	4741	<1	~60	6.24
Motion	30558	<1	~240	1.57

Figure 2 shows the dynamic power values for the different classes of MIPS instructions in our power model. Figure 3 shows the dynamic power of a collection of *MiBench* benchmarks [4] which are typically used in embedded systems. With the help of a benchmark profiling (Figure 4), *Rijndael* has a higher percentage of 'boolean' instructions and less of 'arithmetic' instructions, that explains the high dynamic power of 'Rijndael bench. In contrary, *GSM* has high percentage of 'arithmetic' instructions and other 'miscellaneous' instructions (e.g., nop, mfhi), and less of 'boolean', and hence less dynamic power compared to others. To conclude, with these results we can say that, higher the percentage of arithmetic and miscellaneous instructions, lower the dynamic power consumption.

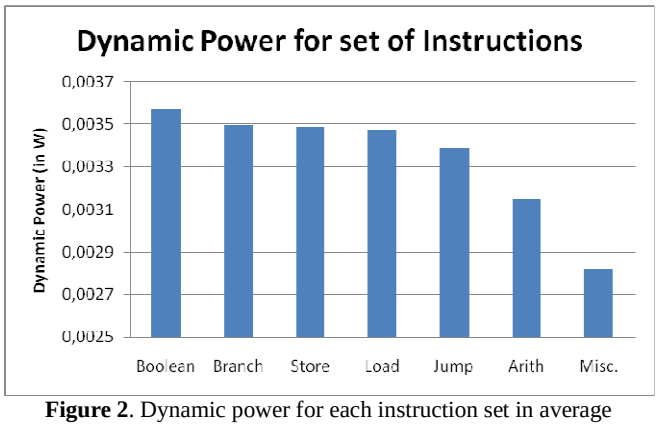

Figure 2. Dynamic power for each instruction set in average

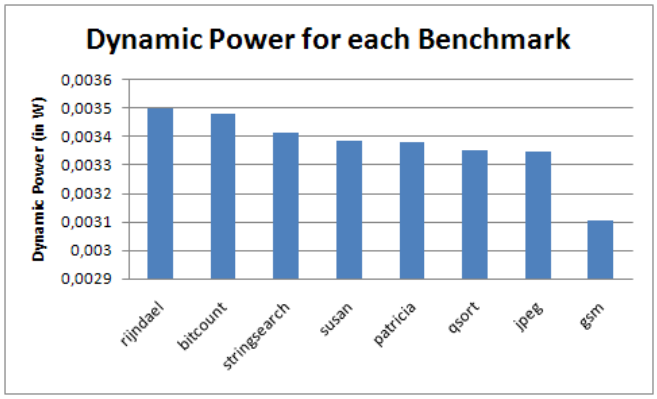

Figure 3. Total dynamic power for each benchmark

Figures 5, 6 and 7 are the graphs depicting effect of Power on Temperature and hence on failure rate (λ) and CFR_{EM}. The graphs are shown with respect to time. Each sample on x-axis corresponds to approximately 27 µsec. The variation in temperature in each benchmark is due to difference in type of instructions executed in time. Each benchmark is executed in a loop for approximately 122ms. When we create loops in benchmarks, we observe that the temperature values from *HotSpot* follow the same pattern for each loop. The difference between minimum and maximum of

temperature is found while changing some internal parameters in *HotSpot*, but it follows the same pattern in all cases. One can remark that the position of steady temperature values (average) of each benchmark is similar to the one of total power (Figure 3).

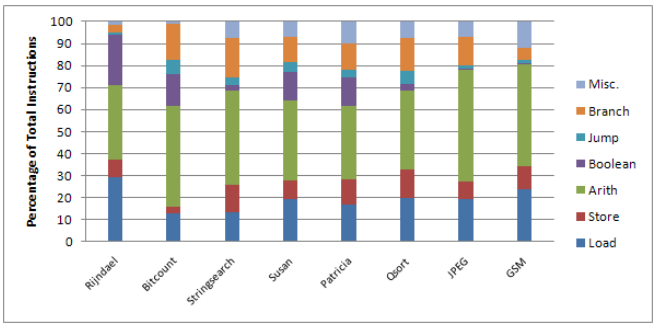

Figure 4. Instruction distribution in *MiBench* benchmark suite

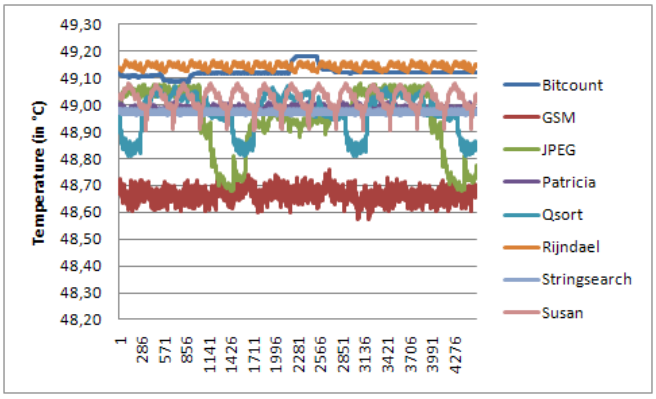

Figure 5. Temperature profiling for all benchmarks

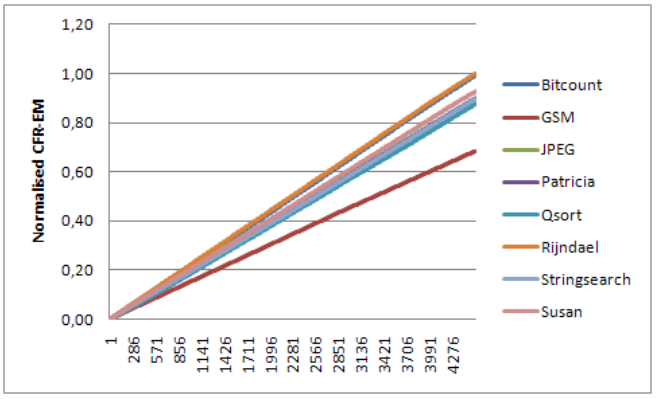

Figure 6. Normalized CFR_{EM} for all benchmarks, from 0 to 122ms

We normalize all of benchmark CFR values with the maximum CFR value, this is because of the lack of knowledge about reliability related parameters from the manufacturer, and hence the constant values are replaced with typical constants found in recent studies **[7]**. Failure rate in Figure 6 shows the variations due to power and temperature both, but largely due to exponential dependency on temperature. CFR_{EM}@122ms tends to increase in linear manner, with curve in the beginning. The user of such reliability simulator can analyze CFR due to different benchmarks and can decide a threshold according to the purpose of the specific processor. The equal time of simulations are shown for clarity, since Patricia is very long in comparison to other benches, but this does not change the behavior of CFR and will continue in the same manner, if the user simulates a loop of same application. Then, the results of Figure 6 can be easily extended to one year or other times. Whatever the time is, 'Rijndael' has the most effect on processor EM compared to 'GSM'. Table 2 shows the variation in CFR for *StringSearch* because of activation energy 'Ea'. The accuracy by which this parameter is determined has a great impact on the reliability level. A 50% variation induces a variation of CFR with a factor of 10^5 @122ms.

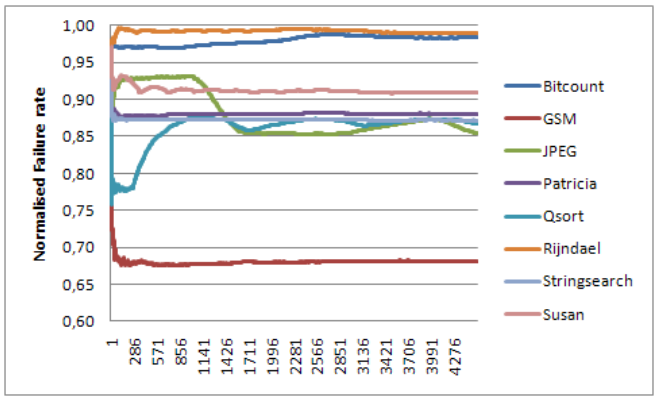

Figure 7. Normalized failure rate variations for all benchmarks

Table 2. Variations in CFR_{EM} with respect to activation energy

Ea	0.6	0.7	0.8	0.9
Normalized CFR_{EM}@8ms	~0,8	~2E-02	~6E-04	~1E-05

5. CONCLUSION

In this paper, we have continued the analyses with *RAAPS* **[1]**, providing some details regarding the failure model used for electromigration at functional level. Our simulator can provide analysis with the execution of same applications, using e.g. different floorplans and scheduling policies, and find the most reliable combination. In this paper, we have also shown the effect of dynamic power on reliability and explored it with time. We showed that, power consumption and the type and number of occurrences of executed instructions have a big effect on MIPS processor that could be the reason of various failure probabilities.

6. REFERENCES

[1] T. Gupta *et al*, "RAAPS: Reliability Aware ArchC based Processor Simulator," *IEEE IRW 2010*, pp.153-156.

[2] S. Rigo *et al.*, "Archc: a systemc based architecture description language". IEEE SBAC-PAD 2004, pages 66-73.

[3] K. Skadron *et al.*, "Temperature-aware microarchitecture: Modeling and implementation", ACM Trans. Archit. Code Optim., vol. 1, issue 1, pp. 94-125, 2004.

[4] M. R. Guthaus *et al.*, "MiBench: A free, commercially representative embedded benchmark suite", IEEE WWC-4, pp. 3-14, 2001.

[5] T. Gupta *et al.*, "High Level Power and Energy Exploration using ArchC", To appear in IEEE SBAC-PAD, 2010.

[6] A. K. Coskun *et al.*, "A simulation methodology for reliability analysis in multi-core SoCs", in ACM GLS-VLSI, 2006.

[7] JEDEC Publication, "Failure Mechanisms and Models for Semiconductor devices", JEP122E, March, 2009.

[8] J. Srinivasan *et al.* "Lifetime Reliability: Toward an Architectural Solution", IEEE Micro 2005, May-Jun, pp. 70-80.

[9] L. Huang and Q. Xu, "AgeSim: A Simulation Framework for Evaluating the Lifetime Reliability of Processor-Based SoCs", IEEE/ACM DATE'10, pp. 51-56, March 2010.