RAAPS: Reliability Aware ArchC based Processor Simulator

T. Gupta, C. Bertolini, O. Heron, N. Ventroux
CEA, LIST, PC94, F-91191 Gif-sur-Yvette, France
Email: tushar.gupta@cea.fr

T. Zimmer, F. Marc
Université Bordeaux I, 351 cours de la Libération
33405 TALENCE cedex, Bordeaux, France
Email: thomas.zimmer@ims-bordeaux.fr

ABSTRACT

In semiconductor industry, designing a SoC is a complex process. Designing reliable SoCs includes study of various configurations involving different operating conditions and considering both hard and soft errors. Designers at higher level of abstraction already have many ways to remove or handle soft errors. This paper aims at analyzing hard errors at functional level. We propose a methodology using state of the art failure models and simulators to provide the cumulative failure rate for a processor simulated at functional level.

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 application. 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, such as *SESAM* **[6]**.

We have many kinds of failure mechanisms that may result in intermittent and permanent errors in integrated circuits. The main ones are electromigration (EM) in interconnects, TDDB in the gate oxides, hot carrier injection (HCI), and negative bias temperature instability (NBTI) in PMOS transistors. These failure mechanisms have been extensively studied at the transistor level in the past, and manufacturers provide technology dependent parameters for each failure mechanism [19]. Designer needs to verify if a design is robust and can handle memory sizes, task scheduling etc for performance and reliability. Relatively to other works related to reliability simulation at front-end such as **[1, 4]**, the reasons we develop a Reliability Aware ArchC based Processor Simulator (RAAPS) 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 a MPSoC, very early in the design flow.

Starting point of our work to develop RAAPS are (i) *PowerArchC* **[7]**, an enhanced *ArchC* based ISS that embeds block-level power estimation capabilities at functional level, (ii) *RTME* **[1]**, 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 RAAPS, we can plug any technology library from manufacturers, packaging library and failure library (typically failure models). Power values at functional level are obtained by an extensive Instruction-Level Power Characterization of technology library [7]. Relatively to a packaging library, temperature values before synthesis are estimated by solving electrical equations in an equivalent RC network that represents the lateral and vertical paths of temperature dissipation over the integrated circuit [3]. Reliability values of the integrated circuit are derived from power, temperature and failure library. In this paper, it is expressed as the cumulative failure rate over time for each failure mechanism [1]. RAAPS could highlight the main failure detractor and the weak part of the design that is prone to this detractor.

In this paper, we show how to build this tool chain at functional level for exploring the effects of different benchmarks for different power values at functional level. Results will be derived for a MIPS processor in 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, a conclusion will be provided in Section 5.

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 failures (permanent faults) in an integrated chip [4]. Brooks et al. [10] 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.

BERT [11] at transistor level and GLACIER [12] at gate level are reliability simulators, studying their results shows the increase in speed and decrease in accuracy as the abstraction level goes from device to design [13]. Srinivasan et al. [4] presented RAMP (Reliability Aware Microprocessor) model for lifetime reliability analysis for microprocessors and performed dynamic reliability management (DRM) using this model. RAMP assumes a uniform device density over the chip and an identical vulnerability of devices to failure mechanisms. Later, Shin et al. [13] 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. Coskun et al [15] 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, RTME [1] and AgeSim [14] presented a solution for lifetime reliability evaluation of processor-based SoCs using state of the art power and temperature simulators. They both take aging effects into consideration as compared to [15]. In [1], we proposed the methodology at RTL, but for MPSoC, it still seems slower for simulations. In [14], Huang et al provided many details about mathematics of calculating aging rate – a new reliability metric – but only numerical examples for electromigration. The integration of AgeSim in a real MPSoC design flow with different technology libraries is not yet addressed. In comparison, we provide a methodology to calculate the cumulative failure rate for 4 failure mechanisms of a processor at functional level. The results are less accurate than BERT or GLACIER, but still close to the real world. Actually, RAAPS 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. RAAPS Methodology

RAAPS 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 **[7]** 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 *micro-architecture and instruction set architecture (ISA)* descriptions, made in *ArchC* language at functional level **[2]**. From that, a synthesis tool like *Design Compiler* **[8]** generates a gate level description, based on a chosen *Technology library*. To characterize power, *ILPC* **[7]** at gate level is performed with *ModelSim* **[9]** and *PrimeTime* **[16]**. It can provide an accurate power consumption of each block, at each clock cycle. We design a parser that outputs the average power consumption of each instruction from power and program traces provided by the simulation tool. We took an assumption to achieve the *Power model* that is we do not consider all possible combinations of instructions and operands.

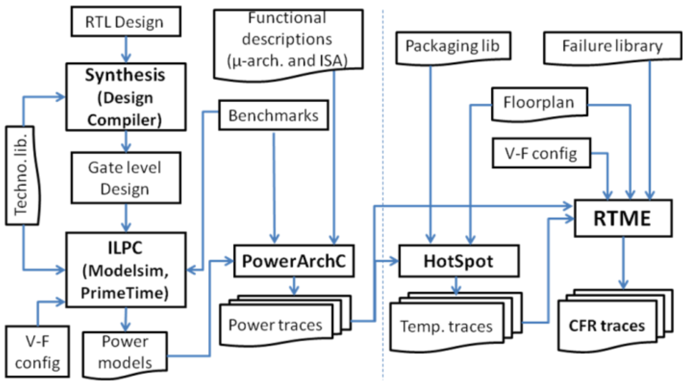

Figure 1. RAAPS Methodology

The power model is next used to back-annotate the ISS generated by *ArchC* with power values for each 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 power of the executed program. The modified ISS version with power capabilities is called *PowerArchC* **[7]**. A *Power model* refers to chosen operating condition (Voltage, Frequency) and characterization campaign (Benchmark).

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]**.

As explained 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: $CFR_x(n, j) = \sum_{i=1}^{n} \lambda_x(i, j) * t_i$ **, w**here 'i' is the time step of duration 't_i', 'λ' is the current failure rate at time 't_i', 'n' is the total number of steps, 'j' is the block reference and 'x' is the failure mechanism reference. In this paper, our failure library considers 4 failure mechanisms: NBTI, HCI, TDDB and EM. Their failure model at block level was detailed in **[1].** Relatively to power and temperature at block level, EM and HCI results depend on power consumption and temperature variations, while NBTI and TDDB results are based on temperature variations. In this paper, we make two assumptions: (i) each failure mechanism occurs in an independent way; (ii) for each separate failure mechanism, the failure rate of a block is a constant value during a cycle 't_i'. Therefore, we use the exponential distribution for representing the reliability behavior of a block. The Mean Time to Failure (MTTF) of a block is the reciprocal of the failure rate; (iii) the equivalent failure rate of a block is based on a series model in which the first device failure always causes the block failure.

Similarly to power and temperature, CFR is computed at each instruction execution i.e. each time step of *PowerArchC*. Parameter 'n' of CFR formula is so equal to the number of instructions in the executed benchmark. Parameter 't_i' is a constant value (let say 'T') related to the frequency at which power and temperature are recorded. The 'i^{th}' line in a power or temperature trace corresponds to the value measured at time step'(i-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 2 on a MIPS 32-bit processor. We use *ArchC v2.0* to generate the MIPS ISS that supports the full *R3000* ISA. We gather an open source RTL description of MIPS that is HMC-MIPS **[18].** With *TSMC 40nm* standard cell libraries for typical case scenario, we synthesize at gate-level the processor description from HMC-MIPS. In this paper, we only consider results for the whole processor. However, we design a detailed floorplan of MIPS, as described in **[1].** The floorplan of the micro-architecture is composed of 7 blocks denoted as follows: fetch, decode, execute, writeback, memory, control and clock. Power, temperature and reliability simulations are so performed at block level.

Three power models (PM1, PM2 and PM3) are built from three different ILPC campaigns: first one is obtained with 'Motion' benchmark **[17]**, second one is obtained with 'Qsort' **[5]** and third one is an average of both. Each ILPC considers the same values for power supply and frequency, i.e., 1.21V and 373MHz respectively. **Table 1** shows the total power consumption for some of many instructions of MIPS for the three power models. More details about the power consumption of other instructions can be obtained in **[7]**.

Instruction mnemonic	PM 1	PM 2	PM 3
addiu	3.19	3.96	3.57
nop	3.13	3.61	3.37
jal	3.05	3.59	3.32
sw	3.06	3.63	3.35

Table 1. Total power consumption in **mW** of some MIPS instructions vs. power models

After that, we simulate 8 benchmarks from MiBench collection **[5]** executed with PowerArchC, separately. For each benchmark (1st column), **Table 2** shows the number of executed instructions by the simulator (2nd column) and the total power (total energy divided by the execution time) consumed by MIPS processor regarding the 3 different power models: PM1, PM2 and PM3 respectively. As we consider functional abstraction level in ArchC, so, we have each instruction executed in 1 clock cycle. For a same power model, differences appear in the results. This is due to the type and number of execution occurrences of each instruction, as explained in **[7]**.

Benches	#ins.	PM 1	PM 2	PM 3
PATRICIA	289,194,424	2.69	3.85	3.71
BITCOUNT	45,593,644	2.43	3.66	3.53
RJINDAEL	33,714,675	2.51	3.71	3.59
GSM	32,662,350	3.16	3.45	3.32
JPEG	29,474,602	3.25	3.98	3.82
QSORT	14,402,560	2.61	4.10	3.92
SUSAN	3,458,842	2.59	3.85	3.72
STRING SEARCH	279,664	2.78	4.19	4.01

Table 2. Number of instructions of Benchmarks and Total MIPS power consumption in **mW** vs. power models

In Hotspot, we assume a heat sink and a thermal spreader having a thickness equal to 0.15mm and 6.9mm respectively. The thermal package model is based on empirical convection heat transfer theories in textbooks. All simulations start from 45°C. We use the block-model based algorithm. Due to small length of simulation time we consider here, temperature variations are very small.

Figures 2, 3, 4 and 5 show CFR results for EM, HCI, NBTI and TDDB respectively. Results are given for the whole processor (without considering memories) regarding the 3 different power models and the 8 different benchmarks. For each failure mechanism, we normalize the CFR values respectively to the maximum CFR obtained from the 8 benchmarks. For each benchmark, parameter 'n' of CFR formula is equal to the number of instructions divided by 100 (Table 2). Parameter 'ti' is a constant value equal to T= 0.27us. For all failure mechanisms, 'patricia' has much higher CFR than other benchmarks; this is solely due to much higher no. of instructions in 'patricia' than others, as shown in Table 2.

Figure 2 shows results for EM. We observe effect of power consumption on EM, since the MTTF for EM is inversely proportional to current density; CFR is directly proportional to power consumption. Temperature variations, as already discussed, do not cause an observable effect in this graph. As expected 'patricia' due to long time of execution and high temperature (46°C in average), put more stress on the processor. When we compare effect of 'gsm' and 'rijndael' benchmarks (with almost similar no. of instructions and temperature of 45°C) on EM, we observe 'rijndael' has more effect for PM1 and PM3 because it consumes more power (i.e., due to the type and executions occurrences of instructions executed).

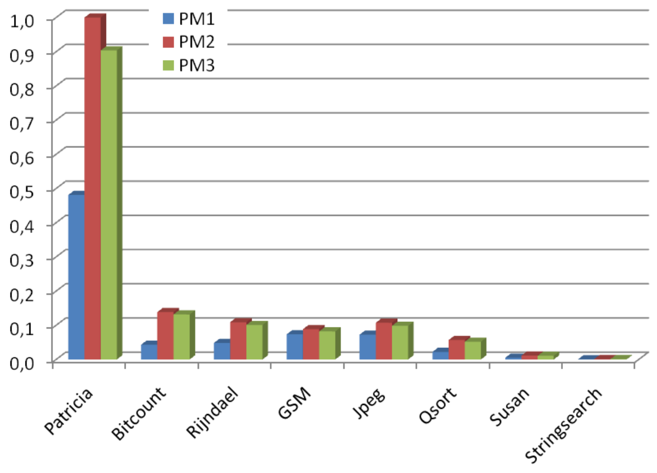

Figure 2. CFR [EM] for MIPS processor

Figure 3 shows CFR results for HCI. As discussed in **[1],** it is inversely proportional to power consumption and temperature but proportional to time. HCI also has an inverse effect of temperature than all other failure mechanisms. Due to small change in temperature, we are not able to highlight this effect. Similar to discussion for EM, when we compare 'bitcount', 'rijndael', 'gsm' and 'jpeg' (with almost similar range of number of instructions), we show that 'bitcount' for PM1 has much more effect on HCI than others, as power consumption for PM1 of 'bitcount' is less.

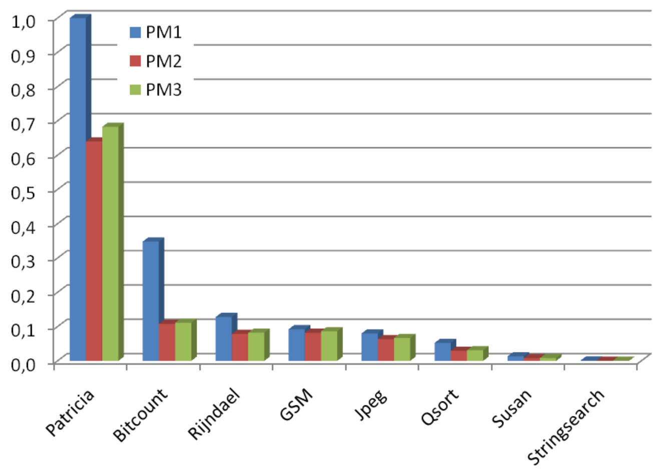

Figure 3. CFR [HCI] for MIPS processor

Figure 4 and 5 show results for NBTI and TDDB respectively. Degradation exhibits dependence on time and temperature. Since there is no effect of power consumption on NBTI, hence for all two power models, we have same results. As expected, TDDB presents the similar behavior as NBTI, since we did not consider all the parameters from the failure models.

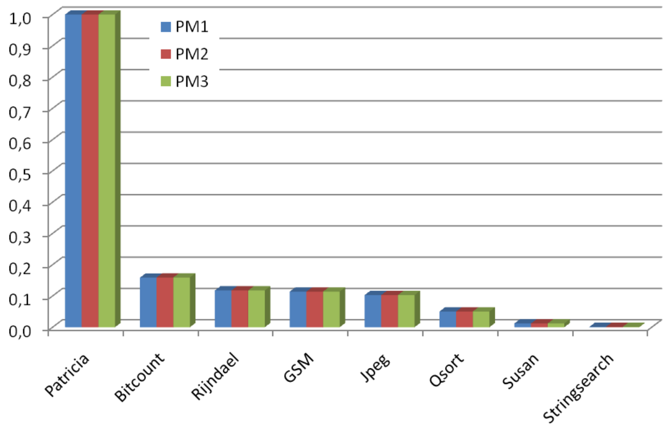

Figure 4. CFR [NBTI] for MIPS processor

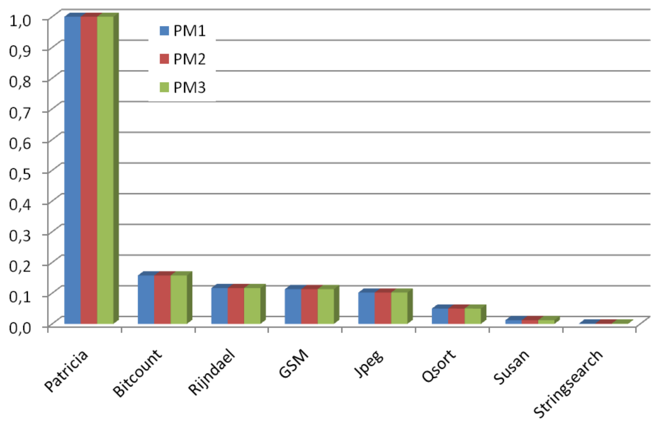

Figure 5. CFR [TDDB] for MIPS processor

As an additional feature, it is also possible to generate CFR results for each block of a processor shown in Figure 6, to analyze deeply the effects of power consumption and temperature on CFR, but with a higher speed than at RTL.

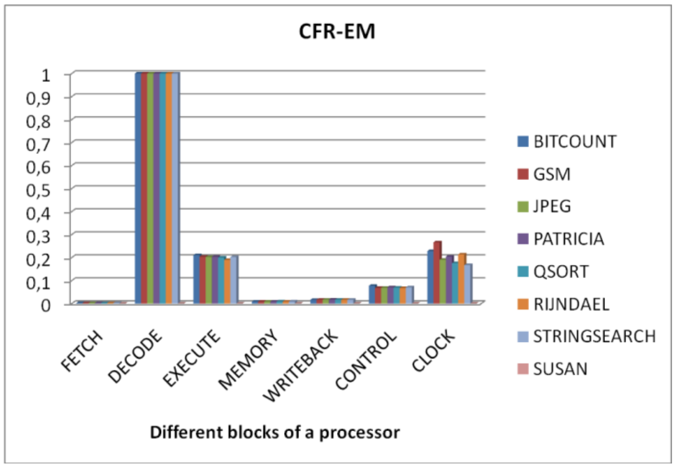

Figure 6. CFR [EM] for different blocks of a processor

5. CONCLUSION

In this paper, we have shown a methodology to predict accurate power and reliability values of a processor at functional level. In our previous work, RTME was used to estimate processor reliability at RTL abstraction, using *Wattch* as a power estimator of a generic RISC processor **[1]**. Another originality of the paper is that it incorporates reliability capabilities in the *ArchC* framework and so it can be easily incorporated in an MPSoC simulator (SystemC language) and will provide similar analysis with the execution of a parallel application, using e.g. different floorplans and scheduling policies. In this paper, we compared the effect of different power scenarios to the reliability of a MIPS processor. We showed that, power consumption and the type and number of occurrences of executed instructions have a big effect on MIPS EM and HCI. Here, temperature has not much effect due to small size of benchmarks. This point will be addressed in future works.

REFERENCES

[1] T. Gupta et al., “Effects of various applications on relative lifetime of processor cores”, IEEE IIRW'09. pp.132-135, 2009.

[2] S. Rigo et al., “Archc: a systemc based architecture description language”. IEEE SBAC-PAD’04, pages 66-73, 2004.

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

[4] J. Srinivasan et al., “The case for lifetime reliability-aware microprocessors”, ACM ISCA’04, pp. 276, 2004.

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

[6] N. Ventroux et al. “SESAM: an MPSoC Simulation Environment for Dynamic Application Processing”, IEEE ICESS’10, pp. 1880-1886, July 2010.

[7] T. Gupta et al., “High Level Power and Energy Exploration using ArchC”, IEEE SBAC-PAD’10, under press, 2010.

[8] Synopsys, Design Compiler, v Z-2007.3.

[9] Mentor Graphics, Modelsim v6.5b.

[10] D. Brooks et al., “Power, thermal, and reliability modeling in nanometer-scale microprocessors”, IEEE Micro, vol. 27, no. 3, p.49-62, May 2007.

[11] R.H. Tu et al., “Berkeley reliability tools – BERT”, IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems, vol. 12, no. 10, pp.1524-1534, 1993.

[12] L. Wu et al., “Glacier: A hot carrier gate level circuit characterization and simulation system for VLSI design”, IEEE ISQED’00, pp. 73-79, 2000.

[13] J. Shin et al., “A Framework for Architecture-Level Lifetime Reliability Modeling”, IEEE/IFIP DSN’07, p.534-543, 2007.

[14] 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.

[15] A. K. Coskun et al., “A simulation methodology for reliability analysis in multi-core SoCs”, ACM GLS-VLSI, pp 95-99, 2006.

[16] Synopsys, PrimeTime PX, v.2007.12.

[17] L. M. Po and W. C. Ma, "A novel four-step search algorithm for fast block motion estimation," IEEE Trans. Circuits Syst. Video Technol., vol. 6, no. 3, pp. 313-317, Jun. 1996.

[18] N. Pinckney et al, “A MIPS R2000 implementation”, DAC’08, pp. 102-107, 2008.

[19] JEDEC Publication, “Failure Mechanisms and Models for Semiconductor Devices”, JEP122E, March 2009.