# Relation between HCI-induced performance degradation and applications in a RISC processor

C. Bertolini, O. Heron, N. Ventroux
CEA, LIST,
PC127, F-91191 Gif-sur-Yvette, France
Email: clement.bertolini@cea.fr

F. Marc
Université Bordeaux I,
351 cours de la Libération
33405 TALENCE cedex, Bordeaux, France
Email: francois.marc@ims-bordeaux.fr

***Abstract*—Die shrinking combined with the non-ideal scaling of voltage increases the probability of MOS transistors to encounter hot carrier injections (HCI). This failure mechanism causes a performance degradation of digital ICs. The evaluation of timing degradations becomes a *must-have* to ensure the expected time-to-market and IC lifetime early in the design flow. In this paper, we present a design/verification flow at front-end from which we accurately analyze the impact of instruction-set architecture on processor timings. We show results on a RISC processor named AntX and designed in a 40 nm TSMC technology. Using typical-case scenarios can increase the maximum operating frequency by 15 % on average compared to a worst-case scenario, while considering the same lifetime. We also identify that the shift operations cause the highest timing degradations along the long processor paths.**

## I. INTRODUCTION

Die shrinking leads to faster devices and higher number of transistors per area but less reliable devices. The non-ideal scaling of voltage in VLSI significantly affects the transistor reliability. One of the major problem is the electric field over-stress that is experienced by the thin gate dielectric of MOS transistors. This negative trend increases the probability of injection of electrons from the drain pin into the gate oxide. This phenomenon, called hot carrier injection [1], results in a drift of the transistor threshold voltage that results in a loss of performance. To keep the reliability constant of the whole integrated circuit (IC), the failure rate per transistor must decrease as transistor density increases at each shrinking step [2].

Today, the design flow includes a sign-off step before design tape-out. Reliability is now part of design requirements. Existing commercial and academic simulation tools provide a solution to evaluate the performance drift due to HCI at transistor and layout levels, such as [3] and [4]. Because these State-of-the-Art tools rely on a transistor-level ageing model and simulation, they are not a sufficient answer to the verification of complex SoCs such that many-core systems: too long simulation time and too high analysis effort by designer. Other approaches propose a verification flow to analyze the effects of HCI on circuit timings at gate level (design netlist), such as [5][6][7][8].

At this abstraction level, the analysis consists in calculating the new gate timings induced by a worst case activity of all transistors. One benefit of this approach is the ability to predict the maximum frequency of the circuit while accounting on HCI effects and others. In order to prevent timing violations during the lifetime, the clock frequency of the IC is set to a value lower than the theoretical one (guard band) so as the expected processor lifetime is guaranteed. In a worst case scenario, the sequence of stimuli applied to circuit inputs often activates a non-functional state, such as test vectors obtained with an automatic test pattern generator (ATPG). However, each individual transistor is stressed depending on the IC usage by the user. Hence, their amount of degradation can be lower than the one predicted in a worst case scenario.

Relatively to RISC processor designs, our idea is to evaluate the real degradation of the processor performance at gate-level by considering the functionality of the circuit, already known at design time. It is possible to analyze the impact of the instruction set architecture (ISA) on HCI and thus identify which instruction will cause the highest degradation in the processor. A first benefit is to prevent an over-estimation of performance degradation due to HCI that leads to a too much conservative design. A second benefit is the ability to identify more accurately which part of the processor microarchitecture is the most sensitive to HCI effects and for which instruction set. By this way, the designer from integrated design manufacturer (IDM) or fabless can obtain a more accurate estimation of HCI effects on processor performance early at the front-end of the design flow. This will open the way to apply design optimization at architecture-level such as [9] and [10]. The novelty of this paper is not to build an HCI-aware verification flow at gate-level, but rather to exploit the capabilities of such flow to make the bridge between the processor functionality and structure for analyzing HCI effects. To the best of our knowledge, there is no study that links HCI-induced performance degradation and the ISA of a processor.

Our analysis framework is similar to those proposed in [5], [10] and [6]. New capabilities are inserted in a verification flow. A performance simulation tool is coupled to a proprietary tool that is able to evaluate the amount of timing

degradation of each logic gate of a design. The simulation tools provide the bit toggling activity of each logic gate input. A static timing analyzer provides a report of the path timings. For a given program, the new flow reports the propagation delay of all processor paths and identifies the longest paths, after simulation. To reach this objective, we derive a timing degradation model of a logic gate, based on the knowledge of HCI physics and State-of-the-Art transistor-level models. The model is parameterized by the bit toggling activity of gate inputs. It is important to notice that this approach is easily adaptable to new technology nodes and compatible with any gate-level design tools. Moreover, the methodology we propose could be extended to Bias-Temperature-Instability (BTI) failure mechanism that relies on a timing degradation model similar to HCI one [5]. The difference is that BTI activation is rather related to the signal probability of logic gate inputs instead of bit toggling activity.

In this paper, we propose three main contributions:

- We show that the worst case scenario is 15 % more pessimistic on performance degradation than a typical case one and leads to a wrong identification of the *weakest* microarchitecture module.
- We show that the path timings can vary about 7 % depending on the executed instructions and data.
- We identify that the arithmetic instructions related to ALU shift unit cause the highest degradation.

Background about HCI physics and modeling is discussed in Section II. Section III presents our timing degradation models of a logic gate and path. In Section IV, a RISC embedded processor called AntX, and the augmented verification flow are described. Finally, section V presents the analysis results and discusses on the impact of applications on HCI behavior. Section VI concludes the paper.

## II. Background

The investigation of HCI at design level requires models that represent the behavior of the failure mechanism regarding the given electrical and design parameters. The models are used to evaluate the amount of degradations of some monitored circuit parameters. Drain current degradations at transistor level, or propagation delay of NAND at logic gate level, are some of these parameters. Ageing modeling and simulation are still an intensive research field. However, there is a consensus on the modeling and simulation methodology. It follows a bottom-up approach which aims at propagating upwards into the design flow hierarchy accurate information about the reliability physics [5]. An *aged* model is associated to each individual primitive of the hierarchy.

HCI is a phenomenon that occurs in MOS transistors. Due to the short channel length, the electric field of the channel becomes very high. HCI is due to the ionization caused by the electron impacts on silicon atoms at the drain, when a current flows through the transistor channel [1].

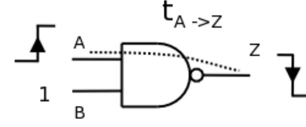



Fig. 1. Example of a timing arc in a 2-input NAND gate

Some of the hot carriers are captured in the transistor gate Si-SiO2 interface, forming a space charge and modifiying the conductive properties of the interface. Over a period of time, this causes a shift or a degradation of drain current, threshold voltage, and transconductance.

At transistor level, most of models are semi-empirical and are based on experimental characterizations of failure physics in the device under high operating levels. Takeda et al. [11] proposed an empirical model based on a worst case transistor biasing. The model represents the amount of degradation of threshold voltage $V_{th}$ with respect of the stress time $t$.

$$\Delta V_{th} = A \cdot t^n \tag{1}$$

where $A$ and $n$ are constant values derived from manufacturer technology and $t$ is the stress time of the transistor i.e. the time during which the transistor is under HCI condition. $A$ depends on drain voltage only. The models suggests that the parameter degradation is a power law with the stress time. The stress time depends on the transistor usage i.e. the total time a current flows through the channel during the experiment. In this work, we adopt this simple model to setup a timing degradation model.

## III. Timing degradation model

Each logic gate is characterized by a set of timing arcs between one input and its output, as shown in Figure 1. Each timing arc represents the possible propagation delay from a single rising/falling transition through the logic gate, e.g. $t_{A->Z}$.

For the reader convenience, we will consider in the following demonstration that the propagation delay of a logic gate refers to only one timing arc. The result can be easily extended to all timing arcs of a logic gate. In this work, the propagation delay of a logic gate under HCI stress condition is represented by the following analytical model, as proposed in [6]:

$$d_{aged} = d_0 + \Delta d \tag{2}$$

Where $d_0$ is the static or fresh propagation delay, $d_{aged}$ is the aged delay and $\Delta d$ represents the timing increase due to HCI or delay degradation. The last term displays the effects of HCI on the internal transistor(s) of the logic gate flowed by a current i.e. when the output switches. This study considers a power law dependency between the delay degradation and the transistor stress time. We refer to the degradation model proposed in [5], which was validated on silicon. Temperature gradients are not yet taken into account. Hence, the delay

degradation has a direct dependency with the $V_{th}$ shift of the transistor (Eq. 1), as follows:

$$\Delta d = B \cdot t^n \quad (3)$$

where $B$ and $n$ are constant values derived from manufacturer technology and $t$ is the stress time of the transistor. $B$ depends on drain voltage.

An internal transistor is under HCI stress condition when a current flows through its channel. In a CMOS technology, this condition depends on the input toggling activity and the internal transistor topology. The stress time of a timing arc can be expressed as follows:

$$t = \theta \cdot TC \quad (4)$$

Where $\theta$ is the time spent by the current to flow through the transistor i.e. when the logic gate output switches. $TC$ corresponds to the number of times the gate input switches (the stress condition is verified). For a given period of operation, the value of $TC$ is incremented whenever a current flows through a transistor. The determination of this variable will be explained in Section IV.

By combining the three previous equations, the propagation delay of a logic gate can be expressed as follows:

$$d_{aged} = d_0 + C \cdot TC^n \quad (5)$$

Where C is a constant equal to $B \cdot \theta^n$.

At circuit level, a combinational path is formed by $m$ logic gates $g_0, ..., g_i, ..., g_{m-1}$ between two registers of an integrated circuit. Whatever is the timing path, it exists an input vector that allows the propagation of a transition on this path until its output. The aged propagation delay of the timing path $dp_{aged}$ is the sum of the aged propagation delay $d_{aged}$ of the logic gates that form this timing path. It can be expressed as follows:

$$dp_{aged} = \sum_{g_i} d_{aged}(i) \quad (6)$$

Where $d_{aged}(i)$ is the aged propagation delay of the logic gate $g_i$. By combining Eq. 5 and Eq. 6, the propagation delay can be expressed as follows:

$$dp_{aged} = dp_0 + C \cdot \sum_{g_i} (TC(i))^n \quad (7)$$

Where $dp_0$ is the fresh propagation delay of the timing path. The term $\sum_{g_i} (TC(i))^n$ will refer to the path toggling count in the following sections. The path toggling count is the total number of times the stress condition of all path gates is verified. It is important to note that this term depends on the circuit usage and in particular on the executed program in a processor design.

In the next section, we present an experiment framework that enables the evaluation of timing degradations in a processor design. The timing degradation models derived in this section (Eq. 5 and Eq. 7) will be implemented in this framework.

## IV. Design and simulation framework

To evaluate HCI degradations, we use a processor core named AntX [12]. AntX is a scalar, 5-stage pipeline, and monothreaded RISC core designed at CEA LIST. It is a 32-bit architecture specifically designed to be used as a low-cost control core in an MPSoC environment. Therefore, there are no complex units such as a branch predictor, a FPU, or a multiplier. Its register file is composed of 16 32-bit registers. The AntX processor has been developed in VHDL and synthesized in 40 nm low-power TSMC technology using Design Compiler tool (Synopsys) [13] at 200 MHz ($V_{DD}$= 1.1 V and T= 25 °C). The overall core area is 10,265 $\mu$m$^2$, which is about 7.24 kilogates. AntX comes along with a dedicated GNU toolchain. To perform its evaluation, the core has been connected to a single external memory to store both instructions and data.

To perform our analysis, a collection of typical application kernels usually found in embedded systems are used. The collection includes: *bitcount*, *basicmath*, *crc* and *fft* from MiBench embedded benchmark suite [14], a bubble sorting algorithm *sort*, a finite impulse response filter *fir*, and a motion estimation algorithm described in [15] and named *motion* in this paper. Figure 2 shows the number of instructions executed by each application kernel. To simplify our analysis, we only consider instructions executed during the shortest bench execution time. Hence, all benches have the same overall execution time, and approximately have the same number of instructions. The figure discriminates arithmetic instructions (ALU) to non-arithmetic ones (e.g. load operations). The ALU instructions are subdivided into 4 categories: addition (add), substraction (sub), Boolean operations (logic) and shift operations (shift). It points out that some of applications, like *bitcount*, use much more arithmetic instructions (especially add and shift operations), compare to the *motion* application for instance. This obervation will be used to explain some of our results in the next section.

The analysis of performance degradations is performed with the aid of the design/verification flow pointed out in Figure 3. It allows a designer to get timing reports at logic gate and path levels. First of all, the RTL description of processor design is synthesized at logic gate level using the TSMC 40 nm technology library. Then, the collection of application kernels is executed on the design netlist with the aid of Modelsim simulation environment (Mentor Graphics) [16]. Primetime tool (Synopsys) [17] is next used to generate both static timing and net activity reports of the timing paths. Here, the propagation delay along the circuit nets is not considered. We implement a proprietary tool, named *Cell Delay Updater*,

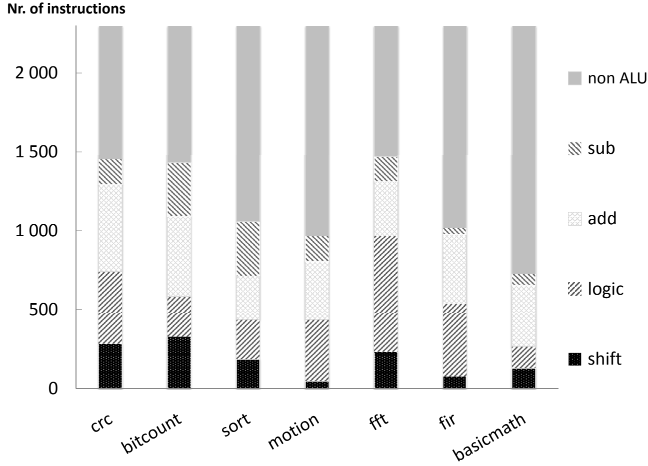



Fig. 2. Number of arithmetic instructions executed by the selected kernel applications

that computes the value of every timing arc of every logic gate at each simulation step of Primetime tool. This tool gets the toggling activity on every net. As mentioned in Section III, the timing arc affected by HCI depends on the toggling input condition and internal topology of transistors. Our tool increments the value of all timing arcs of logic gates, according to Eq. 5, whenever a logic gate input switches. Actually, the amount of degradation would depend on the gate load (factor B in Eq. . Hence, the term $TC$ accounts for the toggling count of the logic gate inputs. Finally, Primetime tool is used again to get the reports of aged circuit timings. To simplify this preliminary analysis, the internal topology is not considered. All timing arcs rely on an identical degradation that remains constant over time. Some past related works, such as [5], will aid to relax this limitation in future. The factor $C$ is equal to $232 \cdot 10^{-6}$ns so as the longest propagation delay will increase by 50% at the end of simulation of our shortest application. The factor $n$ is usually equal to 0.5.

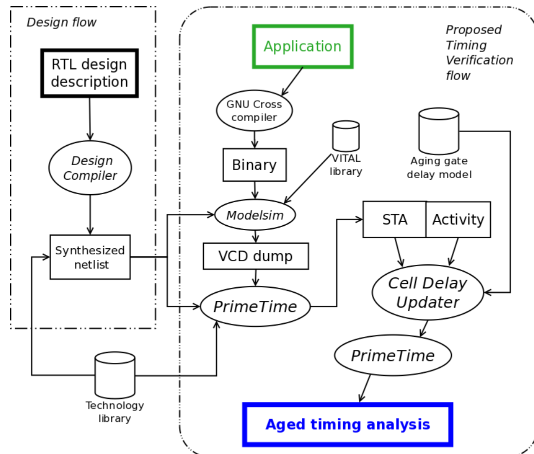



Fig. 3. Design and verification flow used to compute the circuit timings under HCI stress conditions

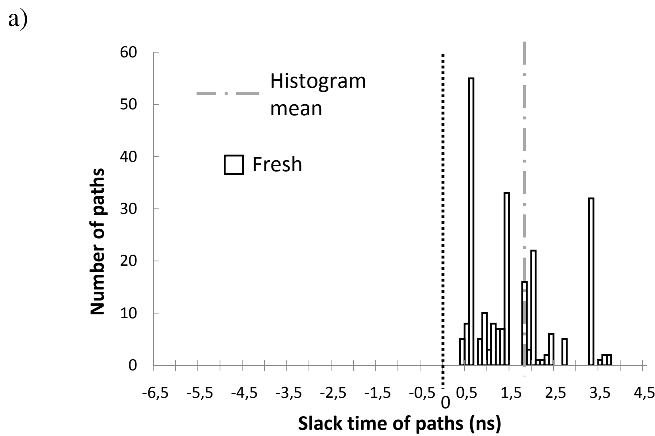
## V. Results

This section presents the result of the timing analysis of AntX processor design under HCI stress condition. A first static timing analysis (STA) is performed to get the list of paths and their *fresh* propagation delay. The total number of paths is equal to 238. The critical path under *fresh* conditions (no HCI stress) has a propagation delay equal to 4.54 ns. This path will be called *fresh longest path*. A second timing analysis is performed with the *aged* propagation delays of logic gates, using our proprietary tool. As mentioned in the previous section, the amount of degradation depends on the bit toggling activity on the logic gate inputs ($TC$) and simulation time ($T_s$). The simulation time is a constant in all experiments ($T_s$ =17133 ns) and corresponds to the shortest execution time among executed benches. Two scenarios of bit toggling activity are considered in this paper. The first one, named *worst case* scenario, assumes that every logic gate input toggles at each processor clock cycle. Here, the bit toggling is generated by our proprietary tool. The second one, called *typical case* scenario, considers that bit toggling activity is controlled by executed application kernels (Fig. 2). These seven typical case scenarios have been presented in the previous section. In all scenarios, the clock frequency remains constant at 200 MHz.

Figure 4.a shows the distribution of fresh path slack times of AntX processor as histogram bars. The x-axis is subdivided into 100 ranges of values. The mean path slack time is equal to 1.57 ns. Figure 4.b compares the path slack times between fresh and stress conditions in an XY-Graph. Stress conditions are performed by running the worst case scenario. The mean value is now equal to -1.1 ns, hence a shift of -170 % compared to fresh conditions. This scenario results in 157 violations of setup time (slack time $< 0$). According to Eq. 7, the aged propagation delay of these paths exceeds the clock period minus the latch setup time.

Figures 5.a and 5.b show the same type of comparison of path slack times. Stress conditions are obtained by running respectively *bitcount* and *motion*. In both cases, 83 paths violate the setup time. In *bitcount* case, the mean value is equal to -0.28 ns, i.e. a shift of -117 % compared to fresh conditions. In *motion* case, the mean value is equal to 0.08 ns, i.e. a shift of -94 % compared to fresh conditions. It is interesting to notice that both applications generate an identical number of timing violations. But *bitcount* causes a higher degradation of slack time. As shown in Figure 2, *bitcount* is the application that executes the highest number of ALU Shift operations while *motion* executes the lowest ones. As detailed later, the pipeline stage *Execute* of AntX processor contains the longest paths, i.e. the highest value $dp_0$. The application kernel that executes the highest number of ALU Shift operations will cause the highest path toggling count (Eq. 7).

a)




b)

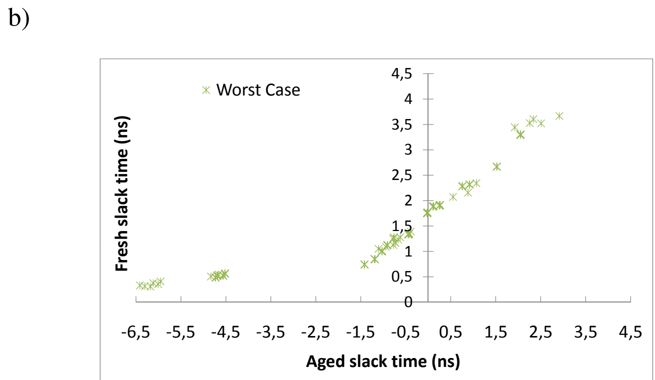


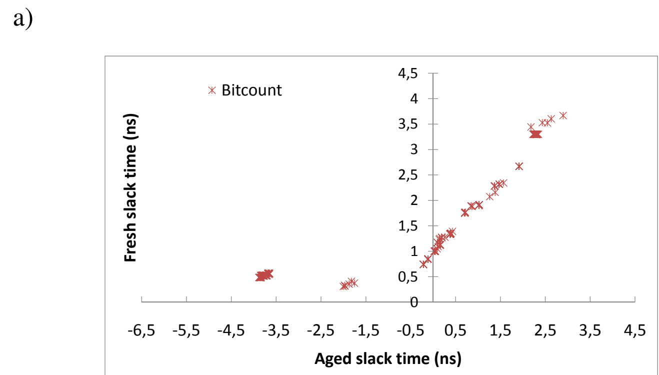

Fig. 4. a) Histogram bars of path slack time of AntX processor under fresh conditions and (b) XY-Graph of path slack time between fresh conditions and worst case scenario

Table I reports the *aged* propagation delays and the relative degradation of the maximum frequency of the processor for both typical and worst case scenarios. The first column shows the simulated scenario. The second column shows the propagation delay of the longest path after running the scenario. The longest path is identified by running an STA. The third column reports the propagation delay of the fresh longest path. A first remark is that the longest path after HCI stress conditions is not the fresh longest path in all scenarios even if the propagation delay of this path is also degraded. Hence, the fresh longest path is not a good monitor of circuit ageing under HCI stress condition.

Let's now discuss on the *worst case* approach impact on the prediction of maximum operating frequency ($F_{MAX}$) of AntX processor. The relative variation of the maximum operating frequency is shown in the third column of Table I. The reference corresponds to the maximum operating frequency of the *fresh longest path* ($1/4.54\,ns$). In the typical case scenarios, the maximum operating frequency decreases from -41 % to -48 % depending on *the executed application kernel.* In the worst case scenario, the maximum operating frequency decreases of -60 %. Therefore, an STA based on worst case scenario would lead to a conservative design and hence, the maximum operating frequency of AntX processor would be under-estimated of 15 % in average.

Figure 6 shows the variation of the slack time of the

a)




b)

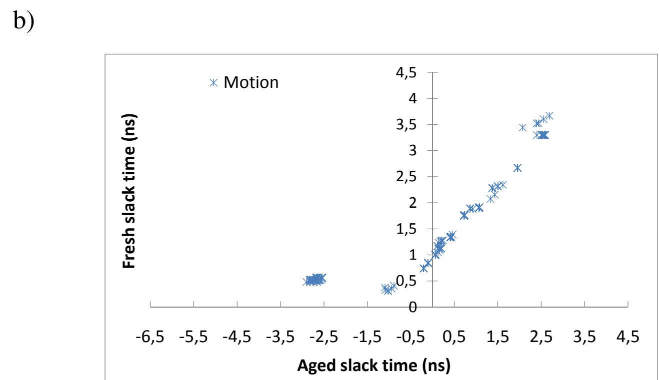



Fig. 5. XY-Graph of path slack time of AntX processor after running (a) *bitcount* and (b) *motion* kernel applications

90 longest paths of AntX processor for both typical case and worst case scenarios. We observed that the variation of their slack time is superior to any other processor paths. These paths belong to the *execute* pipeline stage of the AntX processor. The path with the lowest rank value (x-axis) is the path with the lowest slack time, i.e. the longest path. We also plot the fresh slack time of these paths. For the reader convenience, the paths are grouped into 8 groups denoted as A, B, ..., H. Groups A to F refer to a single path. They are located in the module that manages the control bits of the pipeline (overflow, carry, etc.). Groups G and H refer to a set of paths. The paths belonging to group G are the result bus of the ALU. The paths located in group H are located in the module that computes the memory address.

| Scenarios | $dp_{aged}$ ( ns ) longest path | $dp_{aged}$ ( ns ) fresh longest path | $\%\Delta F_{MAX}$ |
|---|---|---|---|
| basicmath | 8.30 | 6.24 | -45 |
| bitcount | 8.72 | 6.86 | -48 |
| crc | 8.62 | 6.91 | -47 |
| fft | 8.45 | 6.35 | -46 |
| fir | 7.95 | 6.24 | -43 |
| motion | 7.74 | 5.88 | -41 |
| sort | 8.59 | 5.93 | -47 |
| worst case | 11.31 | 11.03 | -60 |

TABLE I
VARIATIONS OF PROPAGATION DELAYS AND MAXIMUM FREQUENCY OF ANTX PROCESSOR FOR TYPICAL AND WORST CASE SCENARIOS

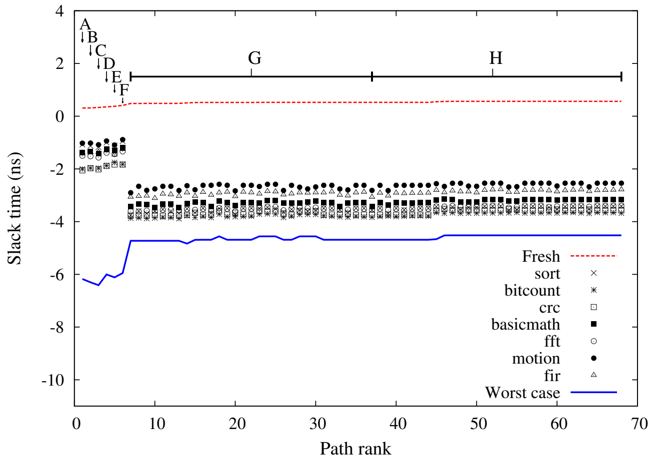



Fig. 6. Variation of slack time of the 68 longest paths of AntX processor for both scenarios

As pointed out by Figure 6, the timing of groups A to F are the most degraded in the worst case scenario. This is due to the fact that the paths with the highest number of lagic gates lead to the highest path toggling count. On the contrary, the timing of groups G to H are the most degraded when considering typical case scenarios. Hence, the *worst case* approach cannot help in identifying the *weak* processor design part, i.e. modules that are the most sensitive to HCI stress. In both scenarios, the related weak modules are different and hence will require different mitigation techniques for HCI.

Let's now focus on the typical case scenarios. The groups G and H contain the *weakest* paths *among the seven applications kernels*. As already mentioned, the application *bitcount* causes the highest variation of slack time while *motion* causes the lowest variation. By comparing this result with Figure 2, it is interesting to notice that the application which executes the highest number of *shift* operations causes the largest variation in the slack time and vice-versa. It means that these instructions lead to the highest path toggling count along the long paths. In our AntX processor, these paths are formed by the logic gates located in the ALU used to control the result bus, and those used for memory address computation.

Let's now discuss about this last observation. The result points at which part of the processor is the most prone to HCI and for which instruction. The user could next build a program that executes various *shift* operations with different operands selected randomly. By executing this program during a certain time, the designer can obtain the highest variation of $\Delta FMAX$. As an example; a drift of 32% after a stress time equal to $1ms$ and a model configuration $(C = 232, n = 0.5)$. According to the target technology, the real model configuration $(C,n)$ could be obtained and the estimated drift could be extrapolated to the mission duration, e.g. a drift of 9% after a stress time of 2 years. Finally, the designer can decide to optimize or not the design if the drift exceeds the threshold expected after 2 years.

## VI. Conclusion

This paper presented the first study that correlates the HCI-induced performance degradations in a RISC processor with ISA, at front-end design flow. We augmented a design/verification flow so as the new flow is able to evaluate the path timings of the design netlist according to the bit toggling activity and HCI effects. We derived a timing degradation model of a logic gate due to HCI from the literature. This flow can be easily adapted to other timing degradation models of a logic gate, such as BTI and technology libraries. We showed that a worst case approach leads to over-estimate the performance degradation by 15 % compared to a typical scenario. We finally identified that the shift operation causes the highest timing degradation along the related AntX paths. This observation will aid the designer to write a reference application that would lead to evaluate more accurately the $\Delta FMAX$ shift.

## References


[1] JEDEC, *Failure Mechanisms and Models for Semiconductor Devices*, JEDEC Solid State Technology Association, November 2010.

[2] ITRS, *Process Integration, Devices, and Structures*, International Technology Roadmap for Semiconductors, 2010.

[3] R. Tu, E. Rosenbaum, W. Chan, C. Li, E. Minami, K. Quader, P. Ko, and C. Hu, "Berkeley reliability tools-bert," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 12, no. 10, pp. 1524 –1534, oct 1993.

[4] P. Lee, M. Kuo, K. Seki, P. Lo, and C. Hu, "Circuit aging simulator (cas)," in *Electron Devices Meeting, 1988. IEDM '88. Technical Digest., International*, 1988, pp. 134 –137.

[5] V. Huard, N. Ruiz, F. Cacho, and E. Pion, "A bottom-up approach for system-on-chip reliability," *Microelectronics Reliability*, vol. 51, pp. 1425 – 1439, 2011.

[6] D. Lorenz, M. Barke, and U. Schlichtmann, "Aging analysis at gate and macro cell level," in *Computer-Aided Design (ICCAD), 2010 IEEE/ACM International Conference on*, nov. 2010, pp. 77 –84.

[7] Y. Kawakami, J. Fang, H. Yonezawa, N. Iwanishi, L. Wu, A. I-Hsien Chen, N. Koike, P. Chen, C.-S. Yeh, and Z. Liu, "Gate-level aged timing simulation methodology for hot-carrier reliability assurance," in *Design Automation Conference, 2000. Proceedings of the ASP-DAC 2000. Asia and South Pacific*, june 2000, pp. 289 –294.

[8] L. Wu, J. Fang, H. Yonezawa, Y. Kawakami, N. Iwanishi, H. Yan, P. Chen, A. I.-H. Chen, N. Koike, Y. Okamoto, C.-S. Yeh, and Z. Liu, "Glacier: a hot carrier gate level circuit characterization and simulation system for vlsi design," in *Quality Electronic Design, 2000. ISQED 2000. Proceedings. IEEE 2000 First International Symposium on*, 2000, pp. 73 –79.

[9] J. Srinivasan, S. V. Adve, P. Bose, and J. A. Rivers, "Lifetime reliability: Toward an architectural solution," *IEEE Micro*, vol. May-Jun, pp. 70–80, 2005.

[10] F. Firouzi, S. Kiamehr, and M. Tahoori, "Nbti mitigation by optimized nop assignment and insertion," *DATE'12*, 2012.

[11] E. Takeda and N. Suzuki, "An empirical model for device degradation due to hot-carrier injection," *Electron Device Letters, IEEE*, vol. 4, no. 4, pp. 111 – 113, apr 1983.

[12] C. Bechara, A. Berhault, N. Ventroux, S. Chevobbe, Y. Lhuillier, R. David, and D. Etiemble, "A small footprint interleaved multithreaded processor for embedded systems," Beirut, Lebanon, dec. 2011.

[13] Synopsys, "Design compiler 2009.06-sp1."

[14] M. Guthaus, J. Ringenberg, D. Ernst, T. Austin, T. Mudge, and R. Brown, "Mibench: A free, commercially representative embedded benchmark suite," in *Workload Characterization, 2001. WWC-4. 2001 IEEE International Workshop on*, dec. 2001, pp. 3 – 14.

[15] L.-M. Po and W.-C. Ma, "A novel four-step search algorithm for fast block motion estimation," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 6, no. 3, pp. 313 –317, jun 1996.

[16] http://www.model.com/, "Modelsim 6.5b."

[17] Synopsys, "Primetime px 2008.12."