Multi-Level Simulation of Non-Functional Properties by Piecewise Evaluation

Hatami, Nadereh; Baranowski, Rafal; Prinetto, Paolo; Wunderlich, Hans-Joachim

ACM Transactions on Design Automation of Electronic Systems (TODAES) Vol. 19(4)
August 2014

doi: http://dx.doi.org/10.1145/2647955

Abstract: As the technology shrinks, nonfunctional properties (NFPs) such as reliability, vulnerability, power consumption, or heat dissipation become as important as system functionality. As NFPs often influence each other, depend on the application and workload of a system, and exhibit nonlinear behavior, NFP simulation over long periods of system operation is computationally expensive, if feasible at all. This article presents a piecewise evaluation method for efficient NFP simulation. Simulation time is divided into intervals called evaluation windows, within which the NFP models are partially linearized. High-speed functional system simulation is achieved by parallel execution of models at different levels of abstraction. A trade-off between simulation speed and accuracy is met by adjusting the size of the evaluation window. As an example, the piecewise evaluation technique is applied to analyze aging caused by two mechanisms, namely Negative Bias Temperature Instability (NBTI) and Hot Carrier Injection (HCI), in order to identify reliability hotspots. Experiments show that the proposed technique yields considerable simulation speedup at a marginal loss of accuracy.

Preprint

A

Multi-Level Simulation of Non-Functional Properties by Piecewise Evaluation

NADEREH HATAMI, University of Stuttgart
RAFAL BARANOWSKI, University of Stuttgart
PAOLO PRINETTO, Politecnico di Torino
HANS-JOACHIM WUNDERLICH, University of Stuttgart

As the technology shrinks, non-functional properties (NFPs) such as reliability, vulnerability, power consumption or heat dissipation become as important as system functionality. As NFPs often influence each other, depend on the application and workload of a system and exhibit non-linear behavior, NFP simulation over long periods of system operation is computationally expensive, if feasible at all.

This paper presents a piecewise evaluation method for efficient NFP simulation. Simulation time is divided into intervals called *evaluation windows*, within which the NFP models are partially linearized. High-speed functional system simulation is achieved by parallel execution of models at different levels of abstraction. A trade-off between simulation speed and accuracy is met by adjusting the size of the evaluation window.

As an example, the piecewise evaluation technique is applied to analyze aging caused by two mechanisms: Negative Bias Temperature Instability (NBTI) and Hot Carrier Injection (HCI) in order to identify reliability hot spots. Experiments show that the proposed technique yields considerable simulation speed-up at a marginal loss of accuracy.

Categories and Subject Descriptors: I.6.4 [**Simulation and Modeling**]: Model Validation and Analysis

General Terms: Design, Verification, Reliability

Additional Key Words and Phrases: Non-functional properties, Transaction Level Modeling (TLM), multi-level simulation, parallel simulation, aging analysis, Negative Bias Temperature Instability (NBTI), Hot Carrier Injection (HCI)

ACM Reference Format:
Nadereh Hatami, Rafal Baranowski, Paolo Prinetto, and Hans-Joachim Wunderlich, 2014. Multi-Level Simulation of Non-Functional Properties by Piecewise Evaluation. *ACM Trans. Des. Autom. Electron. Syst.* V, N, Article A (January YYYY), 22 pages.
DOI:http://dx.doi.org/10.1145/0000000.0000000

1. INTRODUCTION

With the increasing complexity of embedded systems, non-functional aspects become as important as functionality. Power consumption and heat dissipation—the most prominent non-functional properties—enforced power-aware methodologies for design and verification [Man et al. 2011; Shim et al. 1999; Sunwoo et al. 2007]. As the scaling

This work has been supported by the German Research Foundation (DFG) under grant WU 245/11-1 (OASIS).
Author's addresses: N. Hatami, R. Baranowski, and H-.J. Wunderlich, University of Stuttgart, Institute of Computer Architecture and Computer Engineering, Pfaffenwaldring 47, D-70569 Stuttgart, Germany; P. Prinetto, Politecnico di Torino, Dipartimento di Automatica e Informatica, Corso Duca degli Abruzzi 24, I-10129 Torino TO, Italy.

DOI:http://dx.doi.org/10.1145/0000000.0000000

of technology nodes proceeds further, aging processes, such as Negative Bias Temperature Instability (NBTI) and Hot Carrier Injection (HCI), arise as another non-functional bottleneck that requires consideration in early design phases [Alam and Mahapatra 2005; Keane et al. 2010; Lachenal et al. 2007].

Non-Functional Properties (NFPs) relate to system operating conditions rather than system functionality. NFPs depend on the internal hardware structure and technology, as well as system's operating conditions, including the environmental conditions and the workload. NFPs are often interdependent: For instance, power consumption affects heat distribution, which in turn governs aging processes. An accurate NFP estimation approach must take all these dependencies into account.

Existing models for dynamic and static power consumption rely on transistor state, switching activity and temperature [Ghosh et al. 1992; Huang et al. 2004]. Models for robustness and vulnerability take into account electrical and logical error masking at gate- and transistor-level [Mukherjee et al. 2003; Wang and Xie 2011]. Models for aging mechanisms such as NBTI and HCI are governed by transistor-level workload [Wang et al. 2007a; Wang et al. 2007c]. High accuracy is achieved by modeling NFPs at low abstraction levels and requires continuous acquisition of fine grain observables at transistor-level [Viehl et al. 2009; Lorenz et al. 2012]. However, due to increasing design complexity, an extensive system-wide simulation with fine-grained NFP models is often infeasible. To cope with this complexity, NFP analysis can be performed for worst-case conditions [Lorenz et al. 2009], but this may be too pessimistic in case of embedded systems with well defined (fixed) applications.

In this work, we propose an accurate and efficient NFP characterization and analysis method for complex embedded systems. Our method is based on an efficient multi-level system-wide simulation that considers the target system application. High NFP evaluation speed is achieved using a novel piecewise evaluation technique which splits the simulation time into evaluation *windows* and efficiently evaluates NFP models once per window by partial linearization. To consider the mutual impact of different NFPs on each other, all NFP models are integrated into a common evaluation framework. The effect of positive or negative feedback between different NFPs is dynamically considered during simulation. This method can be used with arbitrary user-defined NFP models and can be tailored to fulfill specific accuracy requirements. Since the applications of embedded systems are usually fixed and executed in a repeated manner, the simulation results can be extrapolated to the full system lifetime. The piecewise evaluation method is a fast, yet accurate replacement for cycle-accurate NFP evaluation.

As low-level circuit models are required only for the components under NFP analysis, the proposed method can be used early in the design flow for evaluating the design alternatives w.r.t. NFPs. For the remaining parts of the system, high-level functional models can be reused from design space exploration. As a proof of concept, we show the application of the proposed method to NBTI and HCI aging prediction with an accurate reaction-diffusion model, supported by a model for heat distribution. The method can as well be used for other NFPs, such as reliability or vulnerability.

The paper is organized as follows: The next section gives a brief overview of the presented NFP simulation and evaluation method. Section 3 introduces a parallel, multi-level simulation methodology to obtain NFP observables, while section 4 introduces the piecewise NFP evaluation method. Section 5 shows an application of the proposed method on NBTI and HCI aging mechanisms. Experimental results are discussed in section 6.

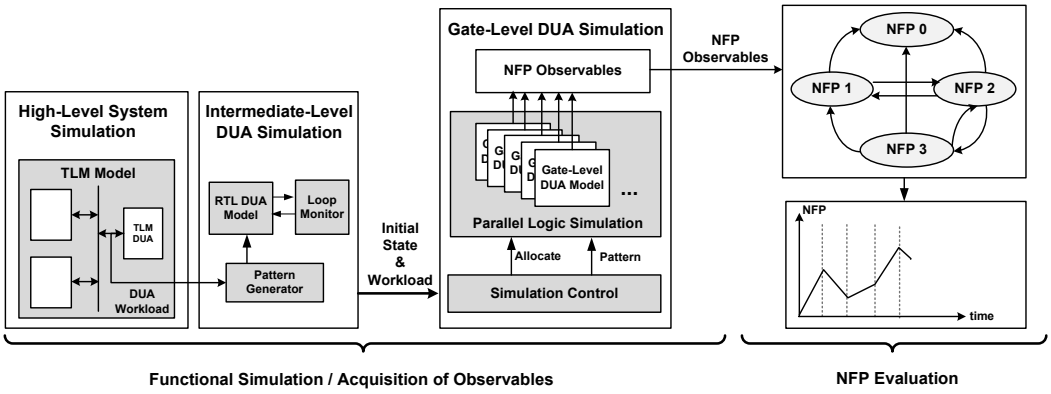

Fig. 1. Proposed NFP Evaluation Methodology

2. OVERVIEW AND RELATED WORK

This section gives an overview of the piecewise NFP evaluation methodology. We introduce the terminology used in this paper and give a short description of multi-level functional NFP simulation approach, followed by the piecewise evaluation method.

2.1. Terminology

Non-Functional Properties (NFPs) relate to the physical state of the system [Glinz 2007]. An NFP is described by a *model* which gives a prediction of the NFP for given operating conditions. *Observables* are defined as the operating conditions that the NFP models depend on, e.g., signal switching activity or duty cycle.

An *evaluation window* is the time interval over which the observables are monitored and aggregated. In the proposed piecewise evaluation method, the NFP models are evaluated once over a window with the averaged observables.

2.2. Multi-Level Functional Simulation

Typically, the workload of a circuit is modeled by input signal probabilities or by a set of "typical workload patterns" that reflect the average application [Lorenz et al. 2009; Wang et al. 2010; Wang et al. 2007c]. As the technology scales, the amplitude of NFP fluctuations under various workloads (e.g., worst vs. average case) is growing [Srinivasan et al. 2004]. This necessitates either a pessimistic worst-case analysis, or an extensive simulation of the target application. However, long-term system simulation for most of today's complex SoCs is infeasible, either because of unavailability of low-level designs at early design phases or because of the huge computational simulation effort. Multi-level simulation facilitates long-term system simulation by using the high-level system model while just the design under analysis is simulated at low-level. Therefore, the accuracy of observable acquisition is preserved while simulation performance improves.

As shown in Fig. 1, the functional simulation method (Section 3) combines three abstraction levels for fast acquisition of observables. Based on the acquired observables, NFP models are evaluated in a periodic manner, as described in Section 4.

The proposed approach analyzes the Design Under Analysis (DUA) with high accuracy within its target system and for its target application. To overcome the computational complexity, the system is simulated at multiple abstraction levels, from gate- up to transaction-level. The observables of a DUA are collected in gate-level simulation to maintain accuracy, while the rest of the system is simulated behaviorally with high

speed. Two techniques are leveraged to accelerate the multi-level simulation: The approach from [Kim et al. 2011b] is adapted to parallelize gate-level simulations by logic state prediction using RT-level models. Fast forwarding of idle loops [Hatami et al. 2012] is applied to speedup the simulation of autonomous cycles.

The concept of multi-level modeling has found widespread use for design validation [Paulin et al. 2002], fault simulation [Baranowski et al. 2011], as well as prediction of non-functional properties such as power [Brooks et al. 2000]. This work does not compete with such specialized approaches, but rather provides the basic methodology for NFP prediction using multi-level models. Alternative approaches to fast gate-level simulation include circuit partitioning (spatial distribution) and parallel event processing [Fujimoto 1990; Chamberlain 1995; Bagrodia et al. 1995]. Spatially-distributed parallel gate-level simulation can be accelerated with General Purpose Graphics Processing Units (GP-GPUs), but the performance strongly depends on the architecture of the simulated design [Chatterjee et al. 2009]. Such approaches can be combined with the proposed method to further accelerate the gate-level simulation.

2.3. Piecewise NFP Evaluation

NFP models are often non-linear w.r.t. the observables and interdependent with each other. For instance, NBTI-induced delay degradation is a non-linear function of the transistor duty cycle. It strongly depends on temperature, which is in turn a function of switching activity, among others. Thus, the NFP models must be continuously evaluated during simulation, considering the dependencies between them.

Specialized methodologies have been proposed in the past for the evaluation of particular NFPs. Viehl et al. propose a simulation-based NFP verification framework to estimate the impact of temperature variations and power consumption on system performance [Viehl et al. 2009]. They use functional system models to generate timed non-functional simulation models and assertions. The analysis is based on best and worst case execution time of the processes. Oboril et al. propose an aging framework at microarchitectural level [Oboril and Tahoori 2012] which enables performance analysis under temperature and power constraints. *New-Age* integrates RT-level, gate-level and analog simulators in conjunction with timing and power/temperature analysis tools [DeBole et al. 2009]. The NBTI analysis is performed by assuming best- and worst-case signal probabilities at device inputs.

Contrary to the existing approaches to NFP evaluation, this paper proposes a holistic simulation methodology that enables prediction of multiple, interdependent non-functional properties. The presented approach is scalable and can be easily extended to arbitrary NFPs. The evaluations are based on target application instead of corner case analysis to provide a realistic prediction of NFPs. The objective of this paper is not to suggest accurate low-level NFP models, but to provide a method for integrating arbitrary low-level models into the system analysis. The right choice of low-level models may depend on the requirements for accuracy and efficiency.

To manage the complex dependencies between different NFPs, the simulation time is divided into *evaluation windows*. Within an evaluation window, the NFP models are partially linearized w.r.t. the observables and the interdependent NFPs. At the end of an evaluation window, the observables obtained from functional simulation are subject to NFP evaluation using the piecewise evaluation technique presented in Section 4. The simulation output is the NFP progression over time.

3. WINDOW-BASED MULTI-LEVEL SIMULATION

This section presents the functional simulation method for efficient acquisition of observables. It combines models at three abstraction levels: transaction-, RT-, and gate-

level. In the following, the simulation procedure and the monitoring of low-level observables are described in detail.

3.1. High-Level System Simulation

To enable high-speed simulation, the temporal and functional behavior of system hardware and software modules is modeled at transaction-level. Transaction-level models (TLM) are often used in design space exploration, and can be reused here for the purpose of NFP evaluation.

TLM allows flexibility in temporal modeling [Ghenassia 2005], while the accuracy can be dynamically adapted during simulation [Radetzki and Salimi Khaligh 2008]. In loosely-timed transaction-level models, the temporal behavior is managed at the level of transactions. In the more accurate, approximately-timed TLM, the time is managed at sub-transaction level, i.e., it is updated for each transaction phase, such as request, transfer, and acknowledgment. While the loosely-timed TLM is faster to simulate, the approximately-timed TLM provides finer grain estimation of system timing, which is desirable for accurate NFP analysis.

During high-level system simulation, the workload of the transaction-level DUA model is continuously monitored. The payload of each transaction of the DUA is captured as *high-level DUA workload*. The high-level DUA workload is used as simulation stimuli at lower levels, as explained below.

3.2. Intermediate-Level DUA Simulation

Given the initial state of the DUA model for each high-level DUA workload (transaction payload), the gate-level simulations of consecutive workloads can potentially be executed in parallel. To facilitate parallel execution at gate-level, the initial states are efficiently calculated at an intermediate level. To this end, the DUA is simulated at RT-level, which is faster than gate-level and accurately models the logic state of the design [Kim et al. 2011a].

The RTL simulator receives the high-level DUA workload as stimuli. The transaction payload is automatically translated into pin- and cycle-accurate input patterns. After the transaction is simulated at RTL, the resulting logic state is captured. This state is later used by the gate-level simulator for the simulation of the next transaction.

Acquisition of NFP-relevant observables is required both in “mission mode” when the core is busy processing a transaction request, as well as when it is idle. During idle operation, the logic state of the DUA (content of its constituent registers) is either stable or follows a loop pattern, as shown in Fig. 2. During this time, the DUA may perform internal operations but it does not receive any incoming transaction. To accelerate the idle simulation, we adapt our recent technique for simulation fast-forwarding [Hatami et al. 2012] to RT-level simulation: As soon as an idle loop is detected, the initial and final logic state of the idle loop are captured, and the further idle simulation is skipped. Just one iteration of the idle loop is simulated later at gate-level for the acquisition of observables.

3.3. Low-Level Parallel DUA Simulation

To acquire fine-grain NFP observables, the DUA is simulated at a low abstraction level. The low-level DUA model is a post-synthesis gate-level netlist, consisting of primitive gates and registers. For the sake of brevity, the gate-level model is assumed to be cycle-accurate. When models with post-synthesis timing annotation are available, the proposed approach can be extended with accurate gate-level timing simulation.

The low-level DUA model is simulated in parallel using the stimuli and initial states provided by the RTL stimulation, as shown in Fig. 3. The gate-level simulation of idle

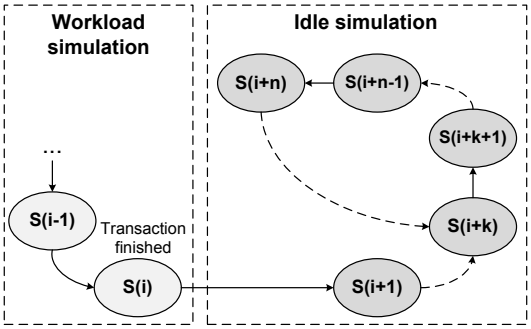

Fig. 2. Loop fast-forwarding for idle cycles

loops and transactions are conducted for the number of cycles determined by RTL simulation.

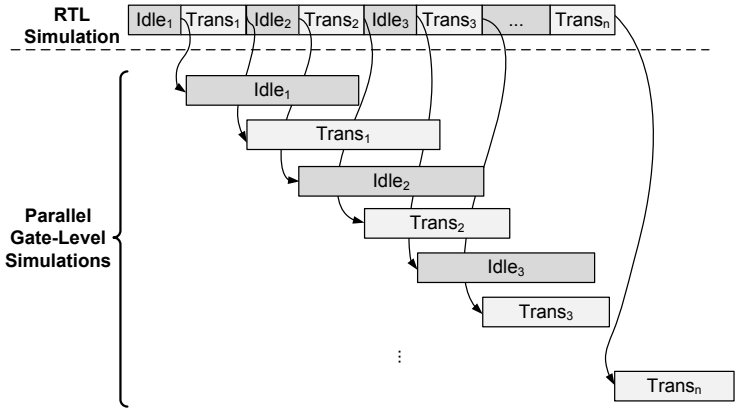

Fig. 3. Parallel gate-level simulation example. The stimuli and initial logic states are generated by RTL simulation.

The gate-level simulator is a *pattern-parallel logic* simulator [Kwong et al. 1999] which is extended with the capacity to aggregate observables such as signal switching activity or duty cycle. When a new workload is received, one of the free pattern-slots is initialized with the initial state obtained from the RTL simulation and the simulation is run. If no pattern-slots are free, the workloads are queued.

For large designs, the capacity of the gate-level simulator may become a bottleneck. To fully exploit simulation parallelism, multiple gate-level pattern-parallel simulation calls are used. The additional simulation calls extend the effective number of pattern-slots.

3.4. Window-Based Multi-Level Simulation Procedure

Concurrent simulation of models at different abstraction levels requires proper simulation control. In the following, the window-based, multi-level simulation procedure is explained.

Cycle-accurate NFP analysis is extremely computationally intensive and may require several days to simulate a second of system runtime. Therefore, to avoid cycle-accurate NFP analysis, the simulation time is split into evaluation windows. The NFP models are evaluated at the end of each window, as explained in Section 4. Fig. 4 shows

the cross-level synchronization of simulations as an example. Horizontal dashed lines represent the window limits at different abstraction levels. The window size is determined according to accuracy requirements.

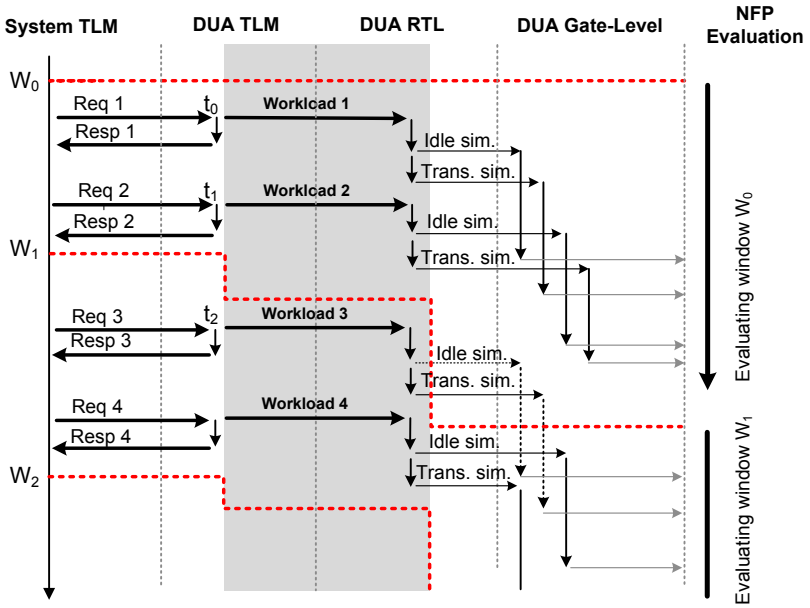

Fig. 4. Cross-level synchronization example

Initially, just the high-level system simulation is run. As long as the transaction-level DUA model is not involved in simulation, both the RTL and gate-level DUA simulations are deferred. Upon detection of a transaction, the high-level DUA workload at time t_0 is sent for RTL simulation.

The RTL simulator receives the high-level DUA workload and calculates the amount of idle cycles which passed since the last transaction. Before the transaction is processed at RTL, this idle time up to t_0 is simulated first. The idle simulation is required to synchronize the state of the RTL model with the simulation time at transaction-level and later with the gate-level. After the simulation time at RTL reaches t_0, the RTL model captures the current state of the DUA and executes the transaction. The RTL simulation proceeds as long as is required to complete the transaction. After the transaction is finished, the RTL simulator sends the captured initial state along with the calculated transaction length—i.e., the number of cycles required to finish the transaction—to the gate-level simulator, which runs concurrently. Note that the gate-level simulator receives two kinds of jobs: transactions and idle simulations. In case of idle jobs, just one idle iteration is simulated and the effect on observables is weighted by the number of skipped iterations. The RTL simulation is resumed as soon as another high-level DUA workload is available.

Processing of transactions and submitting them to gate-level simulation continues until the end of the evaluation window. After reaching the end of the window, the RTL simulation is postponed until all the pending gate-level simulations are finished. After the window is fully processed at gate-level, the collected observables are subject to NFP evaluation, as explained in the next section. The RTL simulation is resumed for the next window just after processing the observables of the current window.

3.5. Collecting NFP Observables

During gate-level simulation, the internal design nodes such as gate and register inputs are continuously monitored. For instance, for power and HCI aging estimation, signal switching activity is observed; for NBTI aging prediction, the stress factor of PMOS transistors is captured. Finally, all observables are collected and stored. The acquired observables constitute an interface to the piecewise NFP evaluation method presented in the next section.

To improve simulation performance, only a subset of observables that are relevant for NFP analysis is acquired during functional simulation. For some NFP models, it is possible to find a set of *representative* observables or NFPs that can be used for NFP prediction: For instance, to predict the delay degradation of the critical path due to NBTI and HCI aging, it may be possible to evaluate the delay degradation of only a subset of representative gates selected, for instance, according to the approach proposed in [Chen et al. 2012].

4. PIECEWISE NFP EVALUATION

The majority of NFP models are dynamic and non-linear with respect to the observables and other non-functional properties. However, they usually result from relatively slow physical processes. For instance, measurable aging effects can be observed after hours, days, or months of transistor operation. Changes in chip temperature are not instantaneous but can be observed after several hundreds of micro- or milliseconds upon a change in power consumption or cooling conditions. Although effects such as electromigration or Time-Dependent Dielectric Breakdown (TDDB) result in a sudden change of circuit properties, the damage accumulates over a long period of time [Segura and Hawkins 2004].

In the proposed method, NFPs such as temperature or accumulated damage are evaluated sparsely to improve simulation performance at a marginal loss of accuracy. The NFP models are evaluated once per evaluation window, considering average system conditions and the aggregated observables within that window. This is in contrast to traditional approaches, which evaluate NFPs based on average system conditions, e.g., [Lorenz et al. 2009; Wang et al. 2010; Wang et al. 2007c].

The window size determines how often the NFP models are evaluated. An evaluation window may span from several hundreds to several million clock cycles, depending on the NFP model response and the required evaluation accuracy. Obviously, as the evaluation window grows, the evaluation speed increases at the cost of reduced accuracy. The window size can be adjusted during simulation to tune the performance and evaluation accuracy.

The piecewise evaluation method proceeds as follows: Let W be the size of the evaluation window. At the end of the i-th simulation window, at time t_i, NFP models are fed with average observables collected in time interval $(t_i - W, t_i]$, and the values of relevant (interdependent) NFPs calculated at time $t_i - W$. In other words, the models are fed with cumulative values of observables from the current evaluation window (e.g., signal probability for NBTI), and NFP predictions from the preceding window (e.g., temperature).

The functional simulation at gate-level is synchronized with piecewise NFP evaluation as follows: When all gate-level simulation jobs reach the end of an evaluation window, the cumulative observables for that window are stored. As soon as the evaluation window is fully covered and cumulative observables are available, the observables are consolidated and the NFP evaluation of the window takes place.

To facilitate this kind of piecewise NFP evaluation, the NFP models are partially linearized w.r.t. the observables and the interdependent NFPs. In the following, we

briefly explain the piecewise NFP evaluation method for general NFP models, followed by an example for NBTI aging model.

4.1. Piecewise Evaluation of General NFP Models

An analytical NFP model, denoted by NFP$(\cdot)$, typically depends on:

— time t,
— system conditions $p_1, p_2, \ldots, p_n$ (e.g., temperature, supply voltage, etc.),
— its own previous value.

The system conditions are themselves non-functional properties that may need to be evaluated concurrently and may be interdependent with the NFP model.

In the following, we describe the piecewise evaluation method for typical kinds of NFP models, expressed with recursive, differential, and memoryless equations.

Recursive Equations

An NFP model expressed with *recursive equations* is most suitable for piecewise evaluation. The NFP value at time t_i is calculated based on the previous NFP value evaluated at time $t_i - W$ and average system conditions in the time interval $(t_i - W, t_i]$. The model equation has the following form:

$$\mathrm{NFP}(t_i) = f\left(W, \mathrm{NFP}(t_i - W), p_1, p_2, \ldots, p_n\right) \tag{1}$$

An example of such a model is given in Section 5.1: The cumulative energy consumed by a CMOS device is described with a simple recursive equation that depends on the current operating conditions and the energy consumed in the past.

Differential Equations

For NFP models given as first order *differential equations*, partial linearization is performed. To this end, we approximate the NFP model as follows:

$$\int_0^T \frac{d\mathrm{NFP}(t)}{dt} \cdot dt \approx \sum_{t_i = 0, W, 2W, \ldots}^{T} \frac{d\mathrm{NFP}(t_i)}{dt} \cdot W \tag{2}$$

where T is the total simulation time.

For instance, the temperature curve follows the Newton's law of cooling or heating, and hence is often modeled with first order differential equations [Lang 1990].

Memoryless Model

A *memoryless* NFP model is expressed with an equation of the form:

$$\mathrm{NFP}(t) = f(t, p_1, p_2, \ldots, p_n). \tag{3}$$

This model provides the NFP progression in time interval $(0, t]$ assuming average system conditions $p_1, \ldots, p_n$ over this interval. The memoryless model cannot be directly used when the system conditions are not constant, i.e., when they fluctuate in the interval $(0, t]$. To enable piecewise evaluation, such models are evaluated in a recursive manner, as explained below:

Let $p_1, \ldots, p_n$ model the average system conditions over time interval $(t_i - W, t_i]$, and let $\mathrm{NFP}(t_i - W)$ denote the NFP approximation at time $t_i - W$. The NFP progression over interval $(t_i - W, t_i]$ is denoted by Δ_{NFP}:

$$\Delta_{\mathrm{NFP}} := \mathrm{NFP}(t_i) - \mathrm{NFP}(t_i - W) \tag{4}$$

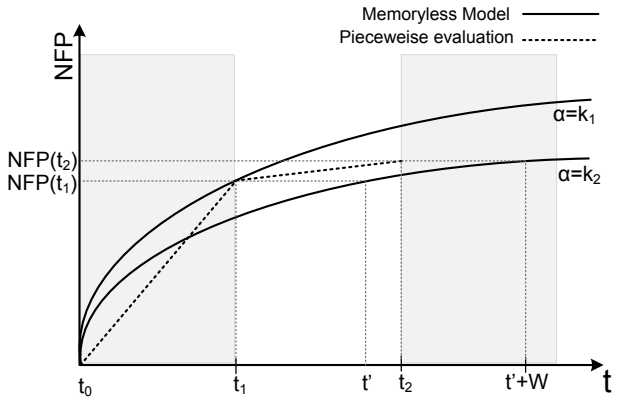

Fig. 5. Piecewise NFP evaluation example

To find Δ_{NFP}, we search for the time t' at which the NFP model under system conditions $p_1, \ldots, p_n$ reaches $\text{NFP}(t_i - W)$, i.e.:

$$\text{NFP}(t', p_1, p_2, \ldots, p_n) = \text{NFP}(t_i - W) \tag{5}$$

After deriving t', the progression of the NFP model over the time interval $(t_i - W, t_i]$ under system conditions $p_1, \ldots, p_n$ is simply calculated as:

$$\begin{aligned} \Delta_{\text{NFP}} = &\text{NFP}(t' + W, p_1, p_2, \ldots, p_n) - \\ &\text{NFP}(t', p_1, p_2, \ldots, p_n) \end{aligned} \tag{6}$$

From (4), (5), and (6) it follows that:

$$\text{NFP}(t_i) = \text{NFP}(t' + W, p_1, p_2, \ldots, p_n). \tag{7}$$

Fig. 5 visualizes the piecewise evaluation of memoryless NFP models. For the sake of simplicity, we assume that the NFP model depends on a single system condition denoted by α. Assume that during the time interval $(0, t_1]$ the average value of α is k_1, and during $(t_1, t_2]$ it is equal to k_2. To estimate the value of $\text{NFP}(t_2)$, we first find t' such that $\text{NFP}(t') = \text{NFP}(t_1)$ for $\alpha = k_2$. Then, $\text{NFP}(t_2)$ is obtained as $\text{NFP}(t' + W)$.

In the following, we present an example of the piecewise evaluation method for a memoryless NFP model describing transistor threshold voltage degradation due to NBTI aging.

4.2. Example: NBTI Aging

The delay degradation of a PMOS transistor due to the NBTI effect in time interval $(0, t]$ can be approximated with the following equation [Noda et al. 2010]:

$$\Delta d(t, \alpha) = A \cdot \alpha^n \cdot t^n \cdot D_0 \tag{8}$$

where α is the probability that $V_{GS} = -V_{DD}$ (*stress factor*), D_0 is the initial transistor delay, and A and n are technology parameters.

The memoryless NBTI equation is evaluated piecewise for window t_i as follows:

$$\Delta d(t_i, \alpha_i) = \Delta d(t' + W, \alpha_i) \tag{9}$$

where α_i is the stress factor in $(0, t_i]$ and t' satisfies:

$$\Delta d(t', \alpha_i) = \Delta d(t_i - W)$$

where $\Delta d(t_i - W)$ is the delay degradation calculated in the previous evaluation window.

According to equation (8), t' is calculated as:

$$A \cdot \alpha_i^n \cdot (t')^n \cdot T_0 = \Delta d(t_i - W)$$

hence:

$$t' = \sqrt[n]{\frac{\Delta d(t_i - W)}{A \cdot T_0}} \cdot \frac{1}{\alpha_i} \tag{10}$$

From equations (8), (9) and (10) the recursive function for delay degradation at time t_i is obtained:

$$\Delta d(t_i, \alpha_i) = A \cdot \alpha_i^n \cdot T_0 \cdot \left(\sqrt[n]{\frac{\Delta d(t_i - W)}{A \cdot T_0}} \cdot \frac{1}{\alpha_i} + W \right)^n \tag{11}$$

5. APPLICATION TO AGING PREDICTION

In this section, we apply the proposed method to two different aging mechanisms: Negative bias temperature instability, and hot carrier injection. NBTI is highly susceptible to temperature variations, which is, in turn, affected by energy consumption. HCI is highly voltage dependent and is also affected by temperature. We apply our piecewise evaluation approach to predict NFPs considering these dependencies. Our goal is to predict the progression of threshold voltage degradation for each transistor in the system. In the following, we describe the models for power dissipation, temperature, as well as HCI and NBTI aging. Note that the presented models serve only as examples and, if needed, can be replaced with more accurate models that consider chip layout, parasitics, etc.

5.1. Dynamic Power Dissipation

Dynamic power dissipation in CMOS circuits results from losses due to parasitic capacitances and short-circuits. The consumed energy is a function of time, supply voltage V_{DD}, operating frequency f, load capacitances of internal nodes, and their switching activity. The cumulative energy consumed due to parasitic capacitances of a single node is evaluated in a piecewise manner for time interval $(t_i - W, t_i]$ with the formula:

$$E(t_i) = E(t_i - W) + V_{DD}^2 \cdot f \cdot C_{load} \cdot \beta_i \cdot W \tag{12}$$

where $E(t_i - W)$ is the energy consumed in the previous evaluation window, and C_{load} and β_i are the node capacitance and switching activity over the interval $(t_i - W, t_i]$, respectively. The short-circuit component of dynamic power dissipation can be computed likewise.

5.2. Temperature Profile

Chip temperature has a significant effect on aging mechanisms and hence must be considered in simulation. A simple temperature model is formulated as [Pedram and Nazarian 2006]:

$$T_{chip} = T_a + R_\theta \cdot \frac{P_{tot}}{A} \tag{13}$$

where T_a is the ambient temperature, R_θ is the equivalent thermal resistance of the substrate, package and heat sink; P_{tot} is the total power consumption, and A is the area. This model forms the basis of more elaborate, block-based models, such as [English et al. 2008; Huang et al. 2004; Noda et al. 2010; Pedram and Nazarian 2006]. Such models can be used to reflect the fluctuations of temperature distribution across the die.

5.3. HCI Aging Mechanism

In MOS transistors, carriers are accelerated (become hot) due to the lateral electric field near the drain junction. A small part of these hot carriers may become energetic enough to be injected into the gate oxide and cause a damage [Bernstein et al. 2006; Oboril and Tahoori 2012]. The rate of hot carrier injections depends on the channel length, oxide thickness, operating voltage, and switching activity of the transistor. The HCI degradation over time period $(0, t]$ can be quantified with the transistor threshold voltage increase, e.g., using the following equation (after [Oboril and Tahoori 2012]):

$$\Delta V_{th}(t) = A \cdot \exp\left(\frac{-E_a}{kT}\right) \cdot \sqrt{\alpha \cdot f \cdot t}, \tag{14}$$

where A is a technology constant, E_a is the activation energy, k is the Boltzmann constant, T denotes the temperature within time period $(0, t]$, f is the operating frequency, and α is the average transistor switching activity within $(0, t]$.

This model is evaluated in a piecewise manner. To calculate the threshold voltage increase over the last window, i.e., to find $\Delta V_{th}(t_i)$ given $\Delta V_{th}(t_i - W)$, the following equation is derived from (14):

$$\Delta V_{th}(t_i) = A \cdot \exp\left(\frac{-E_a}{kT_i}\right) \cdot \sqrt{\alpha_i \cdot f \cdot (t' + W)}, \tag{15}$$

where:

$$t' = \left(\frac{\Delta V_{th}(t_i - W)}{A \cdot \exp\left(\frac{-E_a}{kT_i}\right)}\right)^2 \frac{1}{\alpha_i \cdot f}. \tag{16}$$

Here, T_i denotes the temperature within time period $(t_i - W, t_i]$, and α_i is the average transistor switching activity within $(t_i - W, t_i]$.

5.4. NBTI Aging Mechanism

The NBTI effect consists in the generation of traps at the oxide interface of PMOS transistors. The traps are generated only when the transistor is stressed, i.e., when negative voltage is applied between its gate and source. Removal of the stress condition can partially anneal the interface traps resulting in a partial recovery [Wang et al. 2007a].

NBTI effect causes a gradual increase in the threshold voltage (V_{th}). Due to V_{th} degradation, NBTI results in poor drive current and lower noise margin which shortens the device lifetime and degrades the overall circuit performance [Bhardwaj et al. 2006; Wang et al. 2007b]. As the degradation is a dynamic process characterized by stress and recovery phases, the operating point of the transistors is a crucial observable [Keane et al. 2010; Wang et al. 2007d].

A closed form expression for the upper bound of NBTI-induced V_{th} degradation over time period $(0, t_i]$ under average transistor duty cycle α_i is derived as (after [Cao 2011]):

$$\Delta V_{th}(t_i) = \frac{\sqrt{(K_0 \cdot C_i)^2 \alpha_i}}{1 - [\beta(t_i)]^{\frac{1}{2n}}}, \tag{17}$$

where K_0 and n are technology parameters, $\beta(t_i)$ is a function of time that is defined later, and C_i is the diffusion constant expressed as:

$$C_i = K_1 \exp\left(\frac{-E_a}{k \cdot T_i}\right), \tag{18}$$

where T_i is the temperature within period $(0, t_i]$. K_1 is a constant.

To calculate the threshold voltage increase over the last window, i.e., to find $\Delta V_{th}(t_i)$ given $\Delta V_{th}(t_i - W)$, the parameter $\beta(t_i)$ is calculated in a piecewise manner:

$$\beta(t_i) = 1 - \frac{K_2 + K_3\sqrt{C_i(1-\alpha_i)}}{K_4 + \sqrt{C_i \cdot (t' + W)}}, \tag{19}$$

where K_2, K_3 and K_4 are technology constants, and t' is calculated as:

$$t' = \frac{K_2 + K_3\sqrt{C_i(1-\alpha_i)}}{(1-\beta(t'))C_i^2} - \frac{K_4}{C_i^2}, \tag{20}$$

where:

$$\beta(t') = \left(1 - \frac{\sqrt{(K_0 \cdot C_i)^2 \cdot \alpha_i}}{\Delta V_{th}(t_i - W)}\right)^{2n}. \tag{21}$$

5.5. Model Interrelation

Fig. 6 shows the interrelation between the observables and the models used to predict NBTI and HCI aging mechanisms. The energy required by the DUA is calculated using the model discussed in Section 5.1 considering the switching activity in the circuit. Based on the estimated energy and the ambient temperature, the chip temperature is calculated. The temperature prediction and transistor-level stress factors (duty cycles and switching activities obtained from functional simulation) are used together for accurate evaluation of NBTI and HCI models.

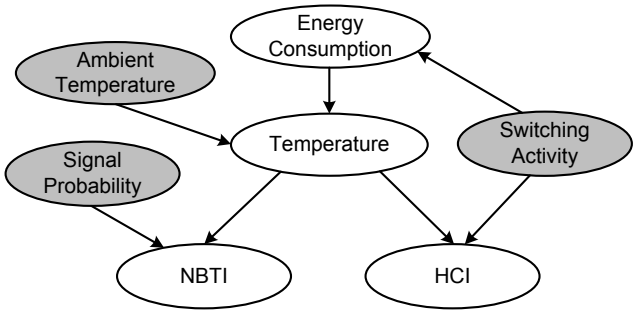

Fig. 6. Relation between NFP models for aging prediction (shown in white) and the required observables (shown in grey)

6. EXPERIMENTS

The proposed method is applied to the case study of an SoC platform based on an OpenRISC-1200 processor[1]. As depicted in Fig. 7, the system is equipped with two display controllers (VGA and LCD), an Inverse Discrete Cosine Transform (IDCT) accelerator, and a Floating Point Unit (FPU). The processor communicates with the peripheral devices through a Wishbone bus using single 32-bit read/write bus cycles.

The subject of the case study is the aging analysis of the IDCT accelerator and the FPU. We consider both the functional system workload and temperature variations of the die. For aging estimation, we assume Predictive Technology Models[2] (PTM) for 45 nm technology. The performance improvement and accuracy loss of the multi-level piecewise evaluation method is compared to the results obtained from a *reference simulator*. The reference simulator uses the same transistor aging model as the proposed method, but it performs an extensive gate-level functional simulation and cycle-accurate aging analysis.

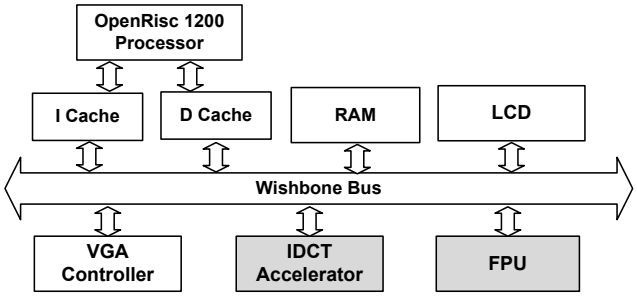

Fig. 7. System architecture

6.1. Experimental Setup

The system is modeled functionally at transaction-level using the SystemC language and the TLM-2.0 modeling library. Each functional unit is modeled as a set of concurrent, behavioral processes. The units are approximately-timed and communicate via TLM sockets.

The IDCT accelerator has been obtained from OpenCores[3]. It is a sequential implementation of the IDCT algorithm with integer precision. The core has been extended with two FIFO memories for input and output data buffers, and a slave interface to the Wishbone bus.

The floating point unit (FPU) has been extracted from the MicroSparcII processor[4] and equipped with a Wishbone slave interface. It is a multi-cycle implementation with double precision arithmetic.

To obtain structural gate-level models, the two DUAs were synthesized for the LSI10k generic library. The IDCT accelerator consists of 28,772 gates (113,686 transistors) and 3,775 registers. The FPU requires 17,113 gates (62,362 transistors) and 630 registers.

[1]OpenRISC Project, http://opencores.org/or1k
[2]http://ptm.asu.edu/
[3]http://www.opencores.org
[4]Oracle, http://www.oracle.com/us/sun

RT-level models of the FPU and IDCT cores are generated automatically from the corresponding gate-level model by *Verilator*[5]. The RTL models are described in SystemC and model the DUAs cycle-accurately.

6.2. Sample Applications

The following signal processing applications using the FPU core are considered in the case study:

— FIR: High-pass Finite Impulse Response filter of order 14
— IIR: Low-pass Butterworth filter of order 5
— FFT: Fast Fourier Transform for vectors of length 64
— IFFT: Inverse FFT (vector length 64)
— JPEG: Image decompression with a floating point IDC transformation for an 8x8 pixel image.

Floating point operations, such as addition, multiplication, and division, are offloaded to the FPU. As input to the applications, random data (noise) is used.

The IDCT core is used to periodically decode and display color JPEG images of three different sizes: 8x8, 16x16 and 64x64 pixels. The applications are run for about 200 million cycles. The progress of the evaluated aging mechanisms is extrapolated from the results obtained for this interval.

6.3. Framework Implementation

The NFP evaluation framework consists of three components that communicate through TCP/IP protocol:

— System simulator (SystemC) with state prediction at RT-level (SystemC)
— Parallel, gate-level DUA simulator for the acquisition of observables (Java)
— NFP evaluator (C++).

The system simulator executes the system model at transaction-level. During simulation, the high-level DUA stimuli in form of time-annotated transaction payloads are captured (cf. Section 3.1). The high-level workload is used as stimuli for DUA simulation at RT-level. The RTL simulator runs in parallel to system (high-level) simulator and fast-forwarding is used to improve its performance (cf. Section 3.2). The RTL simulator generates the logic states of the DUA and the information about idle cycles (i.e., the length, the depth and the number of idle loops). The stimuli generated with the RTL simulator (initial states and information about the required simulation length) is passed over to the gate-level DUA simulator for the acquisition of observables (cf. Section 3.2 and 3.3).

The gate-level DUA simulator is implemented in Java. It contains a gate-level model of the DUA and performs parallel, gate-level logic simulation. An instance of the gate-level simulator is a 64-slot pattern-parallel simulator. Each simulation job received from the RTL simulator (initial state, simulation length) is allocated to one of the free slots. If all slots of the gate-level simulator are occupied, the simulation jobs are queued, or a new instance of the gate-level simulator is started on the same or on a different machine.

During gate-level DUA simulation, the stress factors for NBTI and HCI, i.e., transistor-level switching activity and signal probability, are monitored cycle-accurately (cf. Section 3.5). At the end of each evaluation window, the simulator collects the stress factors from all simulation slots and sends them to the NFP evaluator.

[5]http://www.veripool.org/wiki/verilator

Fig. 8 shows the structure of the NFP evaluator. It comprises models for energy consumption, temperature distribution, as well as NBTI and HCI aging.

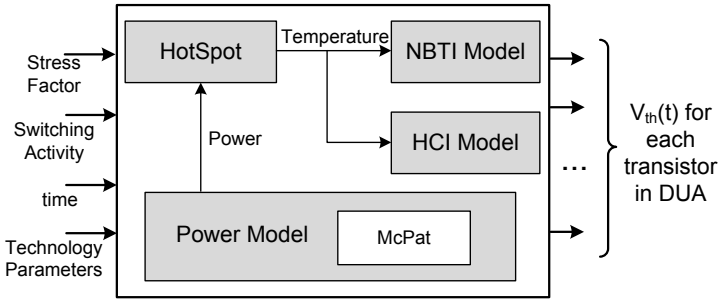

Fig. 8. Block diagram of the NFP evaluator

The energy dissipated by the DUA is calculated using the fine-grain model from Section 5.1. The energy dissipation of the microprocessor is estimated at the architectural level using McPat [Li et al. 2009]—an open source, architectural-level power estimation tool. Temperature profiles are generated with HotSpot, an open source tool for die temperature estimation [Huang et al. 2006].

For aging estimation, the NFP evaluator implements HCI and NBTI aging models presented in Section 5.4 and 5.3. At the end of each evaluation window, the average die temperature is calculated together with the average observables (transistor switching activities for HCI and signal probabilities for NBTI) obtained from the gate-level simulator. Based on this data, for each transistor in the DUA, the NFP evaluator calculates the increase in threshold voltage over each evaluation window.

The framework supports concurrent NFP evaluation of multiple DUAs: The RT- and gate-level simulators for separate DUAs run in parallel and can be distributed across different machines. To accelerate evaluation of the NFP models, the NFP evaluator can also be split into multiple jobs handling different NFPs and/or different DUAs.

6.4. Validation Experiments

The aim of the validation experiments is to evaluate the accuracy loss and speedup of the piecewise evaluation technique for the chosen set of NFP models. In each validation experiment, the system is simulated for hundreds of millions of cycles. We evaluate the aging-induced threshold voltage degradation for each transistor of the DUA. The accuracy and performance of the proposed method are compared to the results from accurate, reference simulation. At the end of each evaluation window t_i, the threshold voltage degradation of each transistor $c \in$ DUA obtained with the proposed method, denoted by $\Delta\hat{V}_{th}^{c}(t_i)$, is compared to the value obtained in the reference simulation, denoted by $\Delta V_{th}^{c}(t_i)$. The absolute estimation error for transistor c is defined as:

$$e^{c}(t_i) = |\Delta\hat{V}_{th}^{c}(t_i) - \Delta V_{th}^{c}(t_i)| \tag{22}$$

The accuracy of the proposed method is measured as a relative mean error over the threshold voltage degradation for all constituent transistors in the DUA:

$$Error(t_i) = \sum_{c \in \mathrm{DUA}} \frac{e^{c}(t_i)}{\Delta V_{th}^{c}(t_i)} \cdot \frac{1}{|\mathrm{DUA}|} \tag{23}$$

To compare the accuracy of our method to state-of-the-art approaches based on random stimuli, as in [Noda et al. 2010], we conduct additional *comparative* experiments. In these experiments, we apply random stimuli to the DUAs and evaluate average observables. These approximated observables are fed to the aging models to calculate the threshold voltage degradation of each transistor in the DUA. This output is compared with the reference simulation which considers the actual workload (application) of the system.

As a thorough reference simulation with cycle-accurate NFP evaluation is very time consuming and requires huge amounts of storage, we use the proposed method with a window size of 10 clock cycles instead of cycle-accurate NFP evaluation. Our experiments show that the error resulting from a window size of 10 cycles is below 10^{-5} mV/day (0.5×10^{-5} %/day) w.r.t cycle-accurate evaluation.

6.5. Experimental Results

The efficiency and accuracy of the piecewise evaluation method is presented in Table I for the JPEG application running on the FPU core. Table II presents the results for the same application running on the IDCT core. The first column shows the window size while the next three columns show the required time for RTL simulation, gate-level (GL) simulation and piecewise aging evaluation, respectively. RTL simulation results in a small performance overhead but enables a huge simulation speedup due to parallelization of gate-level simulations. NFP evaluation time dominates the simulation time for small window sizes, but it drops significantly as the window size grows. The presented speedup considers the gate-level simulation and piecewise evaluation time. Columns 6 and 7 show the relative error growth for NBTI and HCI mechanisms, respectively.

Win. size	RTL sim.[s]	GL sim.[s]	Evaluation [s]	Speedup [x]	Error [%/day] NBTI	HCI
10	160.9	11430.8	75286.1	1	≈ 0	≈ 0
10^2	46.7	4471.1	2745.7	12.0	0.001	0.0001
10^3	35.5	1810.6	318.8	40.7	0.003	0.0003
10^4	35.1	1581.0	35.5	53.6	0.004	0.0008
10^5	34.8	1476.6	5.9	58.5	0.005	0.0017
10^6	34.3	1379.1	1.6	62.8	0.007	0.0017

Table I: FPU-based 8x8 pixel image JPEG decoding

As the window size grows, the simulation time significantly drops while the resulting loss of accuracy is marginal. NFP evaluation time dominates the simulation time for small window sizes, but it reduces significantly as the window size grows.

Win. size	RTL sim.[s]	GL sim.[s]	Evaluation [s]	Speedup [x]	Error [%/day] NBTI	HCI
10	22.4	5545.9	14714.5	1	≈ 0	≈ 0
10^2	2.8	2711.8	1572.8	4.7	0.0003	0.0002
10^3	0.5	264.1	159.0	47.9	0.0011	0.0002
10^4	0.3	46.5	138.9	109.3	0.0011	0.0004
10^5	0.3	16.5	107.8	162.9	0.0012	0.0013
10^6	0.2	16.3	92.9	185.6	0.0065	0.0017

Table II: IDCT-based 8x8 pixel image JPEG decoding

Also for IDCT-based JPEG application, as shown in Table II, the speedup increases significantly with the window size. Piecewise evaluation with window size of 100 cycles is 4.7x faster while window size of 10^6 cycles is 185.6x faster than the reference simulation. Moreover, the error grows only slightly as the window size increases.

In both experiments, the prediction error for HCI as well as NBTI increases very slowly with growing window size while the evaluation speedup is significant. The error analysis shows that the evaluation error does not exceed 0.007 %/day for NBTI and 0.0017 %/day for HCI evaluation with 200 million simulation cycles in the worst case. The IDCT based JPEG application gains more speedup compared to the FPU based JPEG application: FPU based applications use the DUA more intensively, as the majority of CPU operations are offloaded to the FPU. For this reason, the performance gain due to simulation fast forwarding is less compared to the IDCT core.

Fig. 9 and 10 show the overall speedup vs. NBTI evaluation error of the proposed method w.r.t. reference simulation for the FPU and the IDCT core, respectively. The left y axis shows the speedup while the right y axis represents the error growth in percent per day. The error lines are distinguished from speedup by point symbols. As expected, the speedup increases as the window size grows, but it saturates for large window sizes: As the window size gets larger, all slots of the gate-level simulator become occupied. For instance, for the window size of 100 cycles, the overall speedup ranges from 5x to 11x for the FPU, and from 5x to 8x for the IDCT core. For 10^4 cycles, the speedup is from 63x to 487x for the FPU and from 109x to 312x for the IDCT core, depending on the application. Although the speedup for large window sizes is significant, the error rate is only slightly increasing.

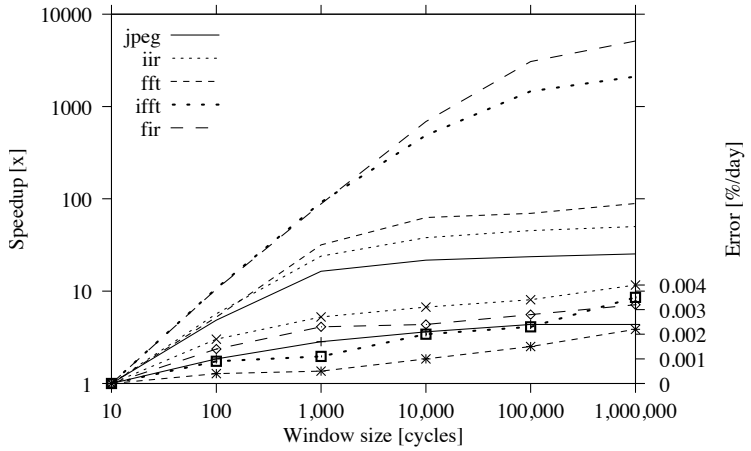

Fig. 9. Overall speedup vs. window size for FPU core

Table III summarizes the speedup vs. NBTI and HCI estimation error for two other applications running on the FPU.

In Fig. 11, the accuracy of the *comparative* experiments that neglect the application is evaluated for the 8x8 JPEG application running on the FPU and IDCT cores and compared against the proposed approach. The error of the piecewise evaluation of NBTI degradation with the window size of 10^6 cycles for FPU is 0.007 %/day while the comparative experiment results in 0.037 %/day. The error of the proposed approach for the IDCT core ranges from 0.0003 %/day for window size of 10^2 cycles to 0.0065 %/day

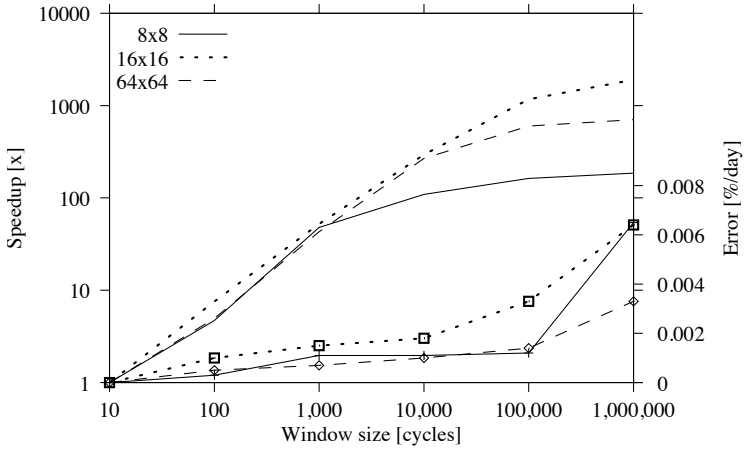

Fig. 10. Overall speedup vs. window size for IDCT core

Win.	Sim. speedup [x]		Eval. speedup [x]		Overall Speedup [x]		NBTI [%/day]		HCI [%/day]	
size	FFT	FIR	FFT	FIR	FFT	FIR	FFT	FIR	FFT	FIR
10	1	1	1	1	1	1	≈ 0	≈ 0	≈ 0	≈ 0
10^2	3.5	5.8	7	10	5	10	0.0004	0.001	0.0009	0.0008
10^3	15.4	10.4	77	90	32	88	0.0005	0.002	0.0012	0.0008
10^4	24.1	15.5	575	775	63	686	0.0010	0.002	0.0013	0.0012
10^5	25.1	17.3	3381	5720	70	3068	0.0015	0.003	0.0013	0.0013
10^6	31.9	17.8	11301	21333	89	5125	0.0022	0.003	0.0014	0.0013

Table III: speedup vs. error for the FPU core

for 10^6 cycles. The comparative experiments result in 0.0435 %/day error which is significantly higher.

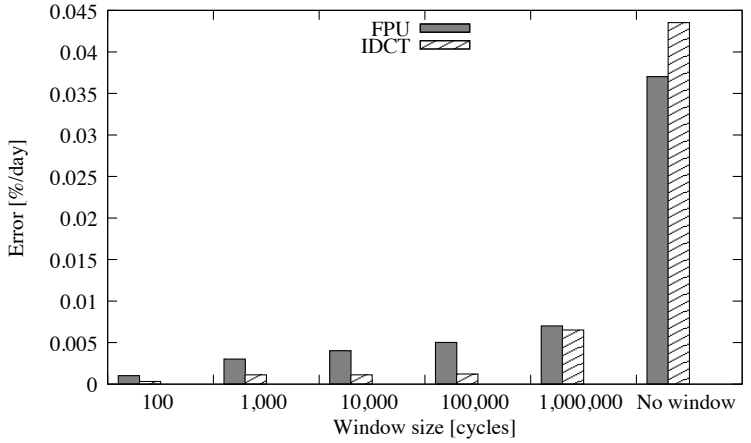

Fig. 11. Relative error with random input patterns for NBTI degradation

7. CONCLUSIONS

This paper proposes a multi-level, simulative, piecewise evaluation method for long term analysis of non-functional parameters. The simulation time is split into evaluation windows, during which the average system operating conditions are observed. NFP models are partially linearized w.r.t. their parameters and evaluated once per window. In the proposed method, the accuracy of NFP evaluation can be adjusted with the window size. Experimental results for NBTI and HCI aging prediction show that with a window size of 1 million clock cycles, the achieved speedup ranges from 25x to 35893x, while the average error for transistor threshold voltage estimation grows from 0.0022%/day to 0.065%/day for NBTI and 0.0013%/day to 0.0018%/day for HCI aging mechanism.

REFERENCES

M.A. Alam and S. Mahapatra. 2005. A Comprehensive Model of PMOS NBTI Degradation. *Microelectronics Reliability* 45, 1 (August 2005), 71–81.

R. Bagrodia, Yu an Chen, V. Jha, and N. Sonpar. 1995. Parallel Gate-Level Circuit Simulation on Shared Memory Architectures. In *Proc. Workshop on Parallel and Distributed Simulation (PADS)*. 170–174.

R. Baranowski, S. Di Carlo, N. Hatami, M.E. Imhof, M.A. Kochte, P. Prinetto, H.J. Wunderlich, and C.G. Zoellin. 2011. Efficient Multi-Level Fault Simulation of HW/SW Systems for Structural Faults. *SCIENCE CHINA Information Sciences* 54, 9 (September 2011), 1784–1796.

J.B. Bernstein, M. Gurfinkel, X. Li, J. Walters, Y. Shapira, and M. Talmor. 2006. Electronic Circuit Reliability Modeling. *Microelectronics Reliability* 46, 12 (December 2006), 1957–1979.

S. Bhardwaj, W. Wang, R. Vattikonda, Y. Cao, and S. Vrudhula. 2006. Predictive Modeling of the NBTI Effect for Reliable Design. In *Proc. Custom Integrated Circuits Conference (CICC)*. 189–192.

D. Brooks, V. Tiwari, and M. Martonosi. 2000. Wattch: A Framework for Architectural-Level Power Analysis and Optimizations. *SIGARCH Comput. Archit. News* 28 (May 2000), 83–94. Issue 2.

Y. Cao. 2011. *Predictive Technology Model for Robust Nanoelectronic Design*. Springer US.

R.D. Chamberlain. 1995. Parallel Logic Simulation of VLSI Systems. In *Proc. Design Automation Conference (DAC)*. 139–143.

D. Chatterjee, A. DeOrio, and V. Bertacco. 2009. Event-Driven Gate-Level Simulation with GP-GPUs. In *Proc. Design Automation Conference (DAC)*. 557–562.

J. Chen, S. Wang, and M. Tehranipoor. 2012. Efficient Selection and Analysis of Critical-reliability Paths and Gates. In *Proc. ACM Great Lakes Symposium on VLSI (GLSVLSI)*. 45–50.

M. DeBole, R. Krishnan, V. Balakrishnan, W. Wang, H. Luo, Y. Wang, Y. Xie, Y. Cao, and N. Vijaykrishnan. 2009. New-Age: A Negative Bias Temperature Instability-Estimation Framework for Microarchitectural Components. *International Journal of Parallel Programming* 37, 4 (August 2009), 417–431.

T. English, K.L. Man, E. Popovici, and M.P. Schellekens. 2008. HotSpot : Visualizing Dynamic Power Consumption in RTL Designs. In *Proc. East-West Design and Test Symposium (EWDTS)*. 45–48.

R. M. Fujimoto. 1990. Parallel Discrete Event Simulation. *Commun. ACM* 33, 10 (October 1990), 30–53.

F. Ghenassia (Ed.). 2005. *Transaction-Level Modeling with SystemC - TLM Concepts and Applications for Embedded Systems*. Springer.

A. Ghosh, S. Devadas, K. Keutzer, and J. White. 1992. Estimation of Average Switching Activity in Combinational and Sequential Circuits, In Proc. Design Automation Conference (DAC). *ACM/IEEE Design Automation Conference (DAC)* (1992), 253–259.

M. Glinz. 2007. On Non-Functional Requirements. In *Proc. International Requirements Engineering Conference (RE)*. 21–26.

N. Hatami, R. Baranowski, P. Prinetto, and H. Wunderlich. 2012. Efficient System-Level Aging Prediction. In *Proc. European Test Symposium (ETS)*. 1–6.

W. Huang, S. Ghosh, S. Velusamy, K. Sankaranarayanan, K. Skadron, and M.R. Stan. 2006. HotSpot: a Compact Thermal Modeling Methodology for Early-Stage VLSI Design. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 14, 5 (May 2006), 501–513.

W. Huang, M. R. Stan, K. Skadron, K. Sankaranarayanan, S. Ghosh, and S. Velusam. 2004. Compact Thermal Modeling for Temperature-Aware Design. In *Proc. Design Automation Conference (DAC)*. 878–883.

J. Keane, Tae-Hyoung Kim, and C.H. Kim. 2010. An On-Chip NBTI Sensor for Measuring pMOS Threshold Voltage Degradation. *IEEE Transaction on VLSI Systems* 18, 6 (October 2010), 947–956.

D. Kim, M. Ciesielski, K. Shim, and S. Yang. 2011b. Temporal Parallel Simulation: A Fast Gate-Level HDL Simulation Using Higher Level Models. In *Proc. Design, Automation and Test in Europe Conference (DATE)*. 1–6.

D. Kim, M. Ciesielski, and S. Yang. 2011a. A New Distributed Event-Driven Gate-Level HDL Simulation by Accurate Prediction. In *Proc. Design, Automation and Test in Europe Conference (DATE)*. 1–4.

A.L.-C. Kwong, B.F. Cockburn, and D.G. Elliott. 1999. Dynamic Combined Pattern-Parallel and Fault-Parallel Fault Simulation on Computational RAM. In *Proc. Canadian Conference on Electrical and Computer Engineering*. 438–445.

D. Lachenal, F. Monsieur, Y. Rey-Tauriac, and A. Bravaix. 2007. HCI Degradation Model Based on the Diffusion Equation Including the MVHR Model. *Microelectronic Engineering* 84, 9-10 (September 2007), 1921–1924.

W. Lang. 1990. Heat Transport from a Chip. *IEEE Trans. on Electron Devices,* 37, 4 (1990), 958–963.

S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi. 2009. McPAT: An Integrated Power, Area, and Timing Modeling Framework for Multicore and Manycore Architectures. In *Proc. International Symposium on Microarchitecture (MICRO)*. 469–480.

D. Lorenz, M. Barke, and U. Schlichtmann. 2012. Efficiently Analyzing the Impact of Aging Effects on Large Integrated Circuits. *Microelectronics Reliability* 52, 8 (2012), 1546–1552.

D. Lorenz, G. Georgakos, and U. Schlichtmann. 2009. Aging Analysis of Circuit Timing Considering NBTI and HCI. In *Proc. International On-Line Testing Symposium (IOLTS)*. 3–8.

K. L. Man, J. Ma, T. T. Jeong, Y. Lei C.and Wu, Sheng-Uei Guan, J. K. Seon, and Y. Lee. 2011. Design, Analysis, Tools and Applications for Programmable High-Speed and Power-Aware 4G Processors. In *Proc. International SoC Design Conference (ISOCC)*. 321–324.

S.S. Mukherjee, C. Weaver, J. Emer, S.K. Reinhardt, and T. Austin. 2003. A Systematic Methodology to Compute the Architectural Vulnerability Factors for a High-Performance Microprocessor. In *Proc. International Symposium on Microarchitecture (MICRO)*. 29–40.

M. Noda, S. Kajihara, Y. Sato, K. Miyase, X. Wen, and Y. Miura. 2010. On Estimation of NBTI-Induced Delay Degradation. In *Proc. European Test Symposium (ETS)*. 107–111.

F. Oboril and M. B. Tahoori. 2012. ExtraTime: Modeling and Analysis of Wearout Due to Transistor Aging at Microarchitecture-Level. In *Proc. International Conference on Dependable Systems and Networks (DSN)*. 1–12.

P.G. Paulin, C. Pilkington, and E. Bensoudane. 2002. StepNP: A System-Level Exploration Platform for Network Processors. *IEEE Trans. Design & Test of Computers* 19, 6 (December 2002), 17–26.

M. Pedram and S. Nazarian. 2006. Thermal Modeling, Analysis, and Management in VLSI Circuits: Principles and Methods. *Proc. IEEE* 94, 8 (August 2006), 1487–1501.

M. Radetzki and R. Salimi Khaligh. 2008. Accuracy-Adaptive Simulation of Transaction Level Models. In *Proc. Design, Automation and Test in Europe Conference (DATE)*. 788–791.

J. Segura and F. C. Hawkins. 2004. *CMOS Electronics:How It Works, How It Fails*. Wiley-IEEE Press.

K. Shim, I. Kyun Oh, S. Min Hong, B. Seon Ryu, K. Young Lee, and T. Won Cho. 1999. A Multi-Level Approach to Low Power MAC Design. In *Proc. Workshop on Signal Processing Systems (SiPS)*. 723–731.

J. Srinivasan, S.V. Adve, P. Bose, and J.A. Rivers. 2004. The Impact of Technology Scaling on Lifetime Reliability. In *Proc. International Conference on Dependable Systems and Networks (DSN)*. 177–186.

D. Sunwoo, H. Al-Sukhni, J. Holt, and D. Chiou. 2007. Early Models for System-Level Power Estimation. In *Proc. International Workshop on Microprocessor Test and Verification (MTV)*. 8–14.

A. Viehl, B. Sander, O. Bringmann, and W. Rosenstiel. 2009. Analysis of Non-functional Properties of MPSoC Designs. In *Languages for Embedded Systems and their Applications*, M. Radetzki (Ed.). Springer.

F. Wang and Y. Xie. 2011. Soft Error Rate Analysis for Combinational Logic Using an Accurate Electrical Masking Model. *IEEE Transaction on Dependable and Secure Computing* 8, 1 (January–February 2011), 137–146.

W. Wang, V. Reddy, A.T. Krishnan, R. Vattikonda, S. Krishnan, and Cao. Y. 2007b. Compact Modeling and Simulation of Circuit Reliability for 65-nm CMOS Technology. *IEEE Transactions on Device and Materials Reliability* 7, 4 (December 2007), 509–517.

W. Wang, Z. Wei, S. Yang, and Y. Cao. 2007c. An Efficient Method to Identify Critical Gates Under Circuit Aging. In *Proc. International Conference on Computer-Aided Design (ICCAD)*. 735–740.

W. Wang, S. Yang, S. Bhardwaj, R. Vattikonda, S. Vrudhula, F. Liu, and Y. Cao. 2007d. The Impact of NBTI on the Performance of Combinational and Sequential Circuits. In *Proc. Design Automation Conference (DAC)*. 364–369.

W. Wang, S. Yang, S. Bhardwaj, S. Vrudhula, F. Liu, and Y. Cao. 2010. The Impact of NBTI Effect on Combinational Circuit: Modeling, Simulation, and Analysis. *IEEE Transaction on VLSI Systems* 18, 2 (January 2010), 173–183.

Y. Wang, H. Luo, K. He, R. Luo, H. Yang, and Y. Xie. 2007a. Temperature-Aware NBTI Modeling and the Impact of Input Vector Control on Performance Degradation. In *Proc. Design, Automation and Test in Europe Conference (DATE)*. 546–551.