Aging-Aware Design of Microprocessor Instruction Pipelines

Fabian Oboril and Mehdi B. Tahoori

***Abstract*—As complementary metal–oxide–semiconductor technologies enter nanometer scales, microprocessors become more vulnerable to transistor aging, mainly due to bias temperature instability and hot carrier injection. These phenomena lead to increasing device delays during the operational lifetime, which result in growing delays of the instruction pipeline stages. However, the aging rates of different stages are different. Hence, a previously delay-balanced pipeline becomes increasingly imbalanced resulting in a non-optimized design in terms of lifetime [i.e., mean time to failure (MTTF)], frequency, area, and power consumption. In this paper, we propose an aging-aware, MTTF-balanced pipeline design, in which the pipeline stage delays are balanced at the desired lifetime rather than at design time. This can lead to significant MTTF (lifetime) improvements as well as additional performance, area, and power benefits. Our experimental results show that for two different microprocessors, MTTF can be extended by at least 2.3 times while achieving an additional 10 % energy improvement with no penalty on delay and area. If the demand for performance is higher than that for a longer MTTF, it is also possible to improve the clock frequency by 2 %.**

***Index Terms*—BTI, HCI, instruction pipeline, microprocessor, transistor aging.**

I. Introduction

NOWADAYS almost all microprocessors ranging from low-power embedded parts to high-performance processors use a pipelined architecture to increase the instruction throughput and by that means the performance [18]. To maximize the performance, designers follow the same paradigm since the dawn of the first pipelined microprocessors. They try to balance all pipeline stage delays at design-time called the *delay-balanced pipeline*. The advantage of this approach was the combination of high throughput together with efficient energy and area usage. This was due to the fact that as long as a pipeline stage is faster than the slowest one (which determines the clock frequency), it can be often made slower by using gate sizing or higher threshold voltage to save energy and die area [13], [20].

Manuscript received August 7, 2013; revised October 17, 2013; accepted December 16, 2013. Date of current version April 17, 2014. This work was supported in by the German Research Foundation (DFG) as part of the National Focal Program "Dependable Embedded Systems" under Grant SPP-1500. This paper was recommended by Associate Editor Y. Cao.

The authors are with Karlsruhe Institute of Technology, Karlsruhe 76131, Germany (e-mail: fabian.oboril@ira.uka.de; mehdi.tahoori@ira.uka.de).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCAD.2014.2298333

However, with the ongoing aggressive transistor scaling of complementary metal–oxide–semiconductor (CMOS) technology, reliability expressed in mean time to failure (MTTF) is becoming an important design constraint, together with performance, power and area [5], [7], [33]. Transistor aging due to *bias temperature instability* (BTI) [39], [49] and *hot carrier injection* (HCI) [44] leads to increasing path delays and so degrades pipeline stage delays during runtime. Hence, nowadays the clock frequency of the shipped parts can no longer be set according to the worst-case delay at design time (t_{design}). Instead, manufacturers have to add safety margins to their delay-balanced designs, to ensure that the chips will be functional for a certain lifetime (t_{target}).

As we will show in this paper, the wearout rates (i.e., delay increase due to BTI and HCI) vary widely among pipeline stages, due to different temperature and usage rates. For example, our experimental results show that the execution stage of the FabScalar microprocessor [12] has a 17 times higher delay increase than the retire stage[1] within the first three years. Hence, although the original pipeline was delay-balanced, after some operational runtime the stage delays become highly imbalanced. This also affects MTTF[2] of different pipeline stages, which varies tremendously (more than 20 times). Thus, one stage can fail due to timing violations while others are still executing correctly. Obviously such a design can be further improved. Slow-aging stages should have less slack to save area and energy, while fast-aging stages should have more slack, to improve the overall MTTF.

In this paper, we propose a radically new MTTF-balanced pipeline design scheme to replace the traditional delay-balanced paradigm. Using this paradigm the MTTF values of all pipeline stages are balanced, instead of the design time delays. As a direct consequence, the stage delays will be balanced at t_{target} (which is the targeted MTTF) rather than at t_{design}. By that means, the full optimization potential for MTTF, area, power, and performance can be exploited. We demonstrate and investigate these benefits using two complementary microprocessors. First, we use FabScalar, an out-of-order, 11-stage superscalar microprocessor. As a second case study, we apply the proposed design methodology to OpenSPARC T1 [1], which is an industrial, in-order, four-way simultaneous multithreading (SMT) processor with six pipeline stages. In

[1]In out-of-order processors the retire stage restores the original instruction order after the out-of-order execution.

[2]In this paper MTTF is equal to the time until first timing violation due to aging occurs.

summary, our MTTF-balanced approach yields a more than 2.3 times longer MTTF, while achieving the same performance (i.e., frequency) compared to the delay-balanced design. In addition, the energy consumption can be reduced by roughly 10 % and also the area can be slightly improved. If an improved MTTF is of secondary interest, the gained headroom can be used to increase the clock frequency by 2 % for a lifetime of three years.

In summary, the key contributions of this paper are as follows.

1) To avoid imbalanced pipeline stage MTTFs and improve the microprocessor design in four dimensions (performance, area, power and reliability), we propose a novel, generic aging-aware pipeline design paradigm applicable to any in-order and out-of-order processor: MTTF-balanced pipeline design.
2) To obtain such a design, we provide a detailed design methodology based on standard commercial synthesis tools that describes the design process for such a pipeline.
3) We present a comprehensive evaluation of the benefits of a MTTF-balanced pipeline design for two different microprocessors in terms of performance, lifetime, area and energy consumption.

A preliminary version of this paper was published in [37]. In this paper we extend our preliminary work by investigating OpenSPARC T1 as a second processor. Moreover, we propose various runtime enhancements for the MTTF-balanced pipeline design flow which significantly reduce the runtime by several folds. In addition, we analyze the trade-off between time consuming post-synthesis gate-level simulations to obtain accurate aging estimations and fast presynthesis high-level simulations in terms of speedup and accuracy.

The rest of this paper is organized as follows. In Section II, the BTI and HCI phenomena are introduced followed by a discussion of related work. The new design paradigm is motivated in Section III, followed by the presentation of the proposed MTTF-balanced design paradigm itself in Section IV. In Section V, the flow to extract MTTF for each stage is explained. Afterward we present in Section VI our experimental results. Finally, Section VII concludes the paper.

II. Background on Transistor Aging

Transistors degrade mainly due to BTI and HCI [5]. Both effects lead to a threshold voltage shift of the impaired transistors, which manifests in increasing gate and path delays. Hence, these effects increase the pipeline stage delay during runtime. In this section, the impact on threshold voltage is explained. Afterward, some related work is discussed.

A. BTI

BTI appears in two difference types, i.e., negative BTI (NBTI) and positive BTI (PBTI). While NBTI affects pMOS transistors, PBTI degrades nMOS transistors and emerge as a reliability issue with the introduction of high-k gate oxides [39]. In both variants, BTI consists of two different phases.

TABLE I
Impact of different parameters on BTI and HCI

vary among stages

	BTI	HCI
Temperature (T)	exponential	exponential
Frequency (f)	-	sublinear
Voltage (V_{dd})	exponential	exponential
Exec. Time (t)	sublinear	sublinear
Usage (δ, α)	sublinear	sublinear

When a logic "0" (logic "1") is applied at the gate of a pMOS (nMOS) transistor, this transistor is under (NBTI/PBTI)-stress. During that phase, traps are generated in the interface between gate oxide and channel, which increases $|V_{th}|$. In contrast, when a logic "1" (logic "0") is applied at the gate of the same transistor, some traps are filled, which leads to a decreasing $|V_{th}|$ (recovery phase). However, the initial shift cannot be entirely compensated leading to an overall V_{th} drift over time. Thereby, the shift depends on several different aspects, e.g., temperature T and the ratio between the time a transistor is under stress and total time (duty cycle δ). For estimating the V_{th} shift the model presented in [49] is used. With this analytical model it is possible to make a long term prediction of the V_{th} shift for a couple of years. Thereby, ΔV_{th} at time $t > 0$ is given by

$$\Delta V_{th}(\delta, T, t) = \left(\frac{\sqrt{K_v^2 \cdot \delta \cdot t_m}}{1 - \beta(\delta, T, t)^{1/2n}} \right)^{2n} \tag{1}$$

where n, is a technology dependent constant. The other parameters can be found in [49].

B. HCI

HCI mainly affects nMOS transistors, where accelerated electrons inside the channel collide with the gate oxide interface and thereby create electron-hole pairs. Thus, free electrons get trapped in the gate oxide layer, which leads to an increasing V_{th}. In contrast to BTI, the V_{th} shift due to HCI is irreversible [49]. The V_{th} shift has an exponential relation with temperature [8], and since "hot" energetic electrons are generated when the nMOS transistor is making a transition, the V_{th} shift is also very sensitive to the number of transitions [44], i.e., clock frequency f, runtime t and switching activity α. Putting all this together leads to the model detailed in [35], which we use in this paper for estimating the V_{th} shift

$$\Delta V_{th}(\alpha, T, t) = A_H \cdot \exp(-E_a/kT) \cdot \sqrt{\alpha \cdot f \cdot t}. \tag{2}$$

A_H and E_a are technology dependent constants, and k is the Boltzmann constant. Note that the temperature relation for technologies using feature sizes larger than 100 nm is reversed [8].

C. Related Work

1) *Aging Mitigation:* In order to alleviate the effects of BTI and HCI, the microelectronic industry including Intel [3], IBM [32], and TSMC [51] spends a great deal of effort on finding new device technologies (e.g., material compounds) that result in lower aging rates. Nevertheless, aging mitigation techniques

are still a necessity. Therefore, several schemes and design techniques are proposed of which we name just a few ones. At device- and circuit-level aging-aware gate sizing [40], [52], power gating [9], and V_{th}-tuning [47] can mitigate the effect of BTI. Based on these techniques an NBTI-resilient processor is introduced in [2]. Furthermore, body biasing techniques [21], stacking-based pin reordering [47], input vector control [50] as well as internal node control [6] and aging-aware path balancing [15], [27] can be used to compensate or slow down the V_{th}-degradation due to BTI and HCI.

Furthermore, various techniques at (micro)architecture-level are proposed to mitigate the impact of transistor aging. Most of these solutions focus on the execution units of a microprocessor, since these are typically the lifetime-limiting factors [14], [37]. Various instruction scheduling techniques are evaluated in [10], [34], [42] that aim at increasing the lifetime of the functional units. Firouzi *et al*. [16] use an aging-aware no operation (NOP) instruction to alleviate the impact of NBTI on the ALU of an MIPS processor, which can be used for other execution units as well. Besides these techniques, in [17] it is proposed to periodically invert the instruction opcode to alleviate aging in the pipeline frontend. Wearout of this part of the pipeline is also addressed by an aging-aware instruction set encoding in [38]. Also, various techniques address wearout in memory elements, such as [26] and [28], that use cell-flipping in order to make the duty-cycle close to 0.5. Another approach presented in [48] is intended to mitigate BTI-induced degradation in a register-file by flipping the leading bits of narrow-width values periodically.

At higher abstraction layers, enhanced application scheduling techniques [46] and various dynamic runtime adaptation techniques such as dynamic voltage and frequency scaling (DVFS) approaches [4], [22], [31], [36] or adaptive body biasing (ABB) techniques [46] are proposed to address BTI- and HCI-induced wearout.

In summary, all of the aforementioned techniques can be classified into two categories, namely, design-time approaches and dynamic runtime techniques (e.g., power gating, scheduling, DVFS, adaptive body biasing, and cell flipping). Our methodology belongs to the first category and is hence orthogonal to all dynamic runtime techniques. As a result, all of them can be used in combination with our method. For that purpose, it is just necessary to take the applied techniques during the aging estimation step into account to avoid an overestimation of the wearout rates (i.e., underestimation of MTTF) and hence imbalanced pipeline design. In addition, the mentioned design-time techniques are complementary to our work as these focus on circuit- and also on device-level, while our approach addresses the microarchitecture-level. Hence, these approaches can also be combined with our design paradigm.

2) *Aging Estimation Flows:* In [14], a flow to estimate the delay degradation of a pipeline stage is introduced, which is similar to the flow presented in Section V. However, this flow can only extract lower and upper bounds for aging induced delay-degradation, but not the real value that is necessary to obtain a balanced pipeline design. Another aging estimation flow is presented in [37]. Although this one is very accurate, it is very time consuming (takes up to several days to extract power, temperature and wearout for all pipeline stages). Hence, it is infeasible for large designs. Instead, the flow used in this paper needs less than two minutes to perform the aging estimation.

3) *Pipeline Delay Balancing:* In the context of pipeline delay rebalancing, a famous technique is cycle-time stealing/borrowing. For example in [29] and [45], such approaches are proposed to rebalance the pipeline delay due to process variation. Cycle time is "stolen" from fast stages and given to slow stages, so that the pipeline can operate at a clock period closer to the average stage delay. Potentially, this idea can be used similarly to our MTTF-balanced design paradigm. Stages that have high aging rates, take some cycle time from stages with lower aging rates to increase their MTTF. However, using these techniques, cycle time has to be redistributed, which is a complex task and not always possible. Instead, our design paradigm does not require a redistribution of cycle time, which makes our technique suitable for almost every design.

Another rebalancing technique using cycle-time borrowing is presented in [41], which is intended to balanced the power consumption of different pipeline stages. Due to this, the problem that some pipeline stages consume much more energy than others is reduced. Potentially, this can also help to avoid hotspots, which can slow down transistor aging. However, since the purpose is to minimize the overall power consumption, the timing slack for each pipeline stage is minimized after applying cycle-time borrowing to the pipeline. This slack reduction can negatively affect MTTF. In contrast, our technique tries to increase the timing slack of some stages, to improve their MTTF and so the MTTF of the entire processor.

III. Motivation and Main Idea

As mentioned in Section II, BTI and HCI can significantly increase pipeline stage delays during runtime. To illustrate this circumstance and motivate our work, we used FabScalar, an out-of-order, 11-stage, superscalar processor [12] as well as the OpenSPARC T1 processor, which has an in-order, six-stage pipeline that features four-way simultaneous multithreading (SMT) [1]. Both processors were synthesized with Synopsys Design Compiler using the TSCM 65 nm library. For the evaluation, we used the framework detailed in Sections V and VI and a timing guardband of 10 %.

To investigate the aging rates of different pipeline stages, we extracted the delay at design time and after three years (= t_{target}) for each stage using the flow described later in Section V. The results of this analysis, illustrated in Fig. 1, clearly show that different pipeline stages have different wearout rates. While the delay of FabScalar's execution stage increases by almost 10 % within three years, the delay of the retire stage increases by less than 1 %, although their delays at design time were similar ($\approx$ 1.35 ns). Similar results are obtained for OpenSPARC as shown in Fig. 1(b). Also, the imbalance in terms of MTTF (given a timing slack of 10 %) can be huge. Between the execution stage of FabScalar, which starts to fail first, and the retire stage, there is a factor of more than 20x difference. This means that one pipeline stage already produces timing failures, while other stages are still running

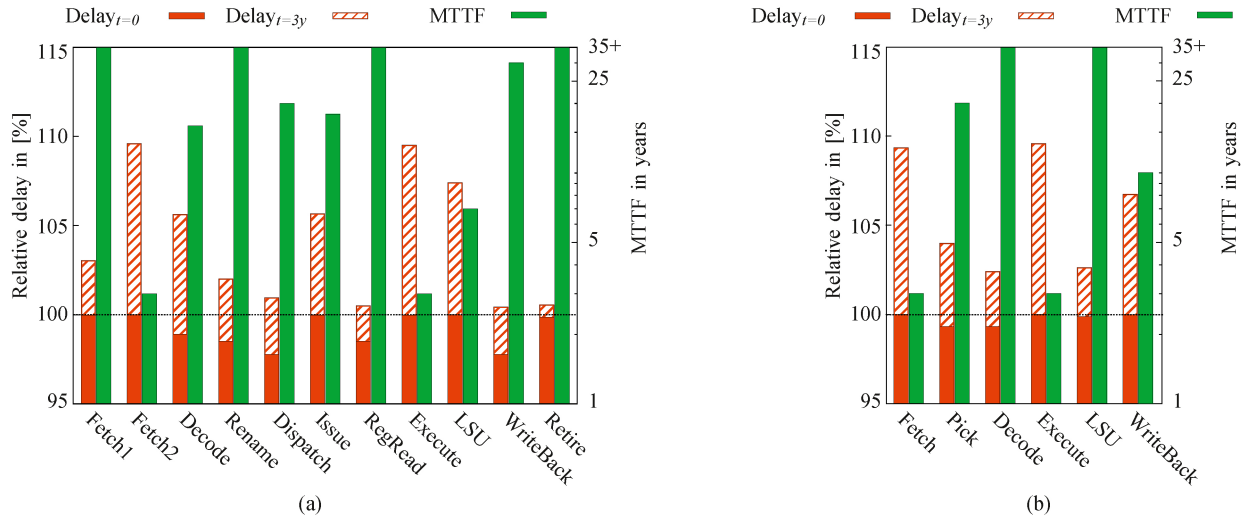

Fig. 1. Delay at design time and after three years (normalized to worst case design time delay) and MTTF for different pipeline stages for two microprocessors (dotted line is the minimum clock period). (a) FabScalar. (b) OpenSPARC T1.

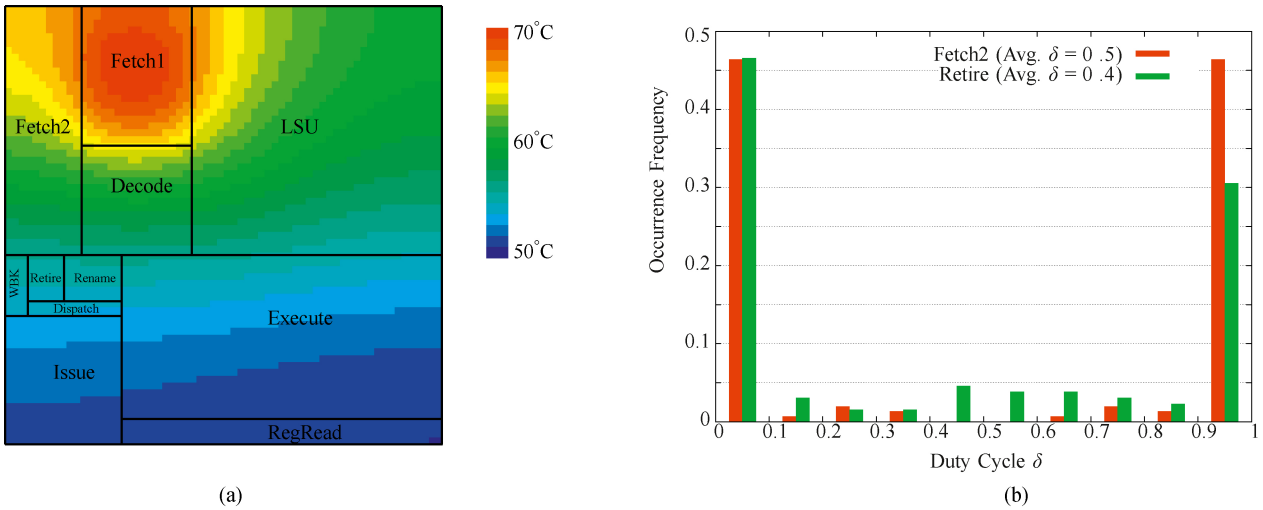

Fig. 2. Illustration of wearout affecting parameters (temperature, duty cycle) for the FabScalar microprocessor extracted with the toolset given in Section VI. (a) Simplified temperature distribution for the FabScalar microprocessor running the 181.mcf benchmark. (b) Distribution of duty cycles for all signals within the 100 most critical paths after three years for FabScalar's Fetch2 and Retire stage (higher duty cycle means faster wearout) for the 181.mcf benchmark.

correctly. Hence, the latter are overdesigned. Furthermore, we observed that the critical stage changes over runtime. For both processors it is the execution stage which is critical after three years. However, at the beginning it is the load-store-unit (LSU) and the writeback stage for FabScalar and OpenSPARC, respectively.

Note that for other microprocessor designs or other technology libraries, Fig. 1 might look different, i.e., other stages age faster, have different MTTF values, and so on. However, the overall observation of delay and MTTF imbalance after a certain runtime remains valid (e.g., in [14] similar results are reported for the IVM microprocessor), as also shown by the two complementary processors chosen for this paper.

The tremendous differences in terms of MTTF and delay degradation are due to the fact that the parameters influencing aging, i.e., temperature and usage (duty cycle, switching activity) are different for different pipeline stages as shown in Table I. This circumstance is also illustrated in Fig. 2 for the FabScalar microprocessor and is also reported by various papers such as [14] and [43]. As shown in Fig. 2(b), the reason for the faster degradation of the Fetch2 stage is not only its higher temperature compared to many other stages [Fig. 2(a)], e.g., the Retire stage, but also the high duty cycle for many signals in the most critical paths. Considering the 100 most critical paths after three years, the average duty cycle in these paths is roughly 0.5, while it is around 0.4 for the Retire stage. The difference in duty cycle of the critical paths and temperature is caused by three major factors: 1) the gate-level implementation; 2) the microarchitecture design; and 3) the workload (i.e., input patterns) that is currently executed by each stage [30]. Moreover, the degradation rate of a pipeline stage strongly depends on the amount of stress on the timing critical paths, while the behavior of all other paths is almost negligible. However, since the aging rate depends on so many

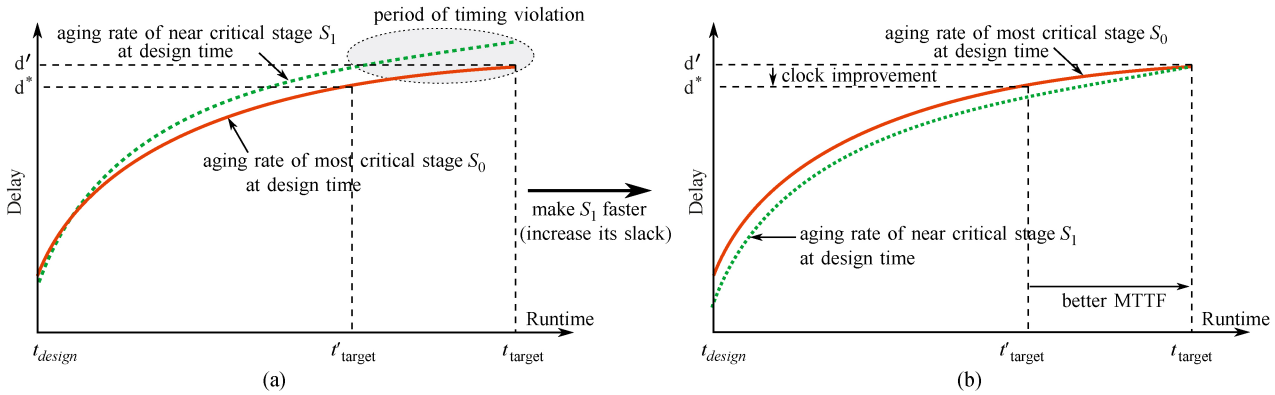

Fig. 3. Abstracted graphic to illustrate the effect of using an MTTF-balanced design on MTTF and performance. Left (Delay-Balanced): at t_{design} stage S_1 has less delay than stage S_0, but ages faster ⇒ For clock period = d' the lifetime = t'_{target}. Right (MTTF-Balanced): S_1 is accelerated ⇒ After t_{target} stages S_0 and S_1 have same delay (d') ⇒ longer MTTF possible (t_{target} instead of t'_{target}) Alternatively: For same MTTF (t'_{target}) a smaller clock period is possible (d^* instead of d').

interrelated factors, it is a necessity to run detailed circuit-level simulations to obtain accurate results.

A. Main Idea

Since MTTF of the microprocessor is determined by the smallest MTTF of all pipeline stages, and the clock frequency is mandated by the slowest stage after the target lifetime t_{target}, it is obvious that the described imbalance leads to a sub-optimal design. Looking at the criticality of different pipeline stages at t_{target}, there are two possible optimization strategies.

1) First, stages that are faster (i.e., have more slack) than the critical stage after t_{target} can be designed slower (i.e., with less slack), by applying appropriate gate-sizing techniques, using a higher threshold voltage, and so on, leading to extra energy and area savings [13], [20].
2) Second, if a stage S_1 degrades faster than the design-time-critical stage S_0, it can be designed faster (i.e., with more slack). This can be used in two different ways, both shown in Fig. 3, where the dotted line represents the slow-aging, design-time-critical stage and the solid line, the fast-aging stage. First, the clock frequency (i.e., clock period) can be kept constant, so that a higher MTTF can be achieved (which means that t_{target} can be increased). In Fig. 3, this means that the clock period remains at d' and the target lifetime increases to t_{target}. In the second case, MTTF is kept constant (i.e., equal to t'_{target}), so that the clock period can . be reduced (i.e., less guardband and hence higher frequency, meaning higher performance). Using the annotations from Fig. 3, it means that the new clock period is $d^* < d'$.
   Depending on whether the first or second case is chosen as the optimization target, the design should be balanced either at t_{target} or t'_{target}, respectively. In this paper, we have chosen the first case as explained in Section VI.

Hence, in summary, slow-aging stages should be designed with less slack (i.e., slower) to save area and energy, while the timing slack for fast-aging stages should be increased (i.e., speed-up), to improve their MTTF and in turn the MTTF of the entire microprocessor or the overall performance (i.e., clock frequency). Thereby. the key design aspect is that the pipeline stage delays should be balanced after the target lifetime and not at design time. Since this is not achievable using the traditional delay-balanced design approach, we propose a new MTTF-balanced pipeline design. We will explain this paradigm in detail in the following section.

IV. Aging-Aware Pipeline Design

The key idea of the MTTF-balanced pipeline design is that the pipeline stage delays are balanced after the desired lifetime t_{target} (t'_{target}) rather than at design time t_{design}. Hence, also MTTFs of all stages are equal to t_{target} (t'_{target}). In the following we will explain the flow to generate an MTTF-balanced pipeline. Thereby, for the matter of simplicity we will only refer to the targeted lifetime t_{target}.

A. Generation of MTTF-Balanced Pipeline Design

The transformation flow, detailed in Fig. 4, is a multipurpose flow that can be used for various optimization targets such as getting the best MTTF while maintaining a given clock frequency or extracting a design that is as fast as possible for a given target lifetime t_{target}. The last case will be explained in detail now.

The starting point of the transformation process is a delay-balanced design, as it is used nowadays (Step 1). Next, the delay, d^*, after the given target lifetime t_{target} of the critical stage at design time is extracted using the flow presented later in Section V (Step 2). Since this stage cannot be designed any faster (otherwise it would not be critical at design time), the clock period of the final MTTF-balanced pipeline cannot be smaller than this delay. Since the final design should be as fast as possible, d^* will work as a reference for the clock period.

The next step (Step 3) is to extract the delay d_i of each pipeline stage after t_{target} and to compare it with d^*. If the delay is smaller than d^* (i.e. the stage is faster than necessary), a new, slower version of this pipeline stage will be generated. Therefore, we adjust the timing constraints for this pipeline stage and resynthesize it. As the synthesis tool supports gate-sizing, path reorganization and time borrowing all these techniques will be applied in parallel to optimize for delay,

```
1. Generate a delay-balanced design
2. d* = delay at t_target of stage, that is critical at t_design
   d' = delay at t_target of stage, that is critical at t_target
   d~ = given clock delay in case there is a clock target and no lifetime target
/* ⇒ clock period of delay-balanced design d_clk = d' > d* */
3. Forall stages i = 0 , . . . , n do
      Extract d_i = delay at t_target of stage i
      stage^i_old = current version of stage i
      /* if stage is faster than necessary */
      If (d_i < d*) then
         stage^i_new = new version of stage^i_old with more delay at t_design
      /* if stage is slower than necessary */
      Elseif (d_i < d*) then
         stage^i_new = new version of stage^i_old with less delay at t_design
         /* if no (further) speedup is possible adjust d* */
         If (stage^i_new == stage^i_old ) then
            d* = d_i
            Goto 3. /*restart with new d* */
         End if
      Else
         stage^i_new = stage^i_old
   End if
Forall
   End
/* Extract new delay information at t_target */
4. Forall stages i = 0 , . . . , n do
      Extract d_i,new = delay at t_target of stage^i_new
      /* if old version was faster and new version is slower take old one */
      If (d_i,new > d*) and (d_i < d*) then
         stage^i_new = stage^i_old
      End if
   End
/* If no more modifications are possible, transformation is done */
5. For all stages i = 0 , . . . , n do
      If (stage^i_old ≠ stage^i_new ) then
         Goto 3. /* restart */
      Endif
   End
6. Done. /* generated MTTF-balanced design with d_clk = d* */
```

Fig. 4. Algorithm to transform a delay-balanced design into an MTTF-balanced design.

power and area efficiency. In addition also a higher threshold voltage can be used [13], [20]. However, as a result it is possible that different pipeline stage designs result in the same MTTF. In Section IV-B, we will explain how such scenarios are handled. In case resynthesis is not feasible, it is also possible to modify only small sub-circuits or gates [11].

If the delay d_i is greater than d^* (i.e., stage is slower than necessary), a new, faster version (using gate sizing, and so on) will be generated. If this is not possible, the final design has to use a clock frequency of at least d_i. Hence, d^* will be increased and set to d_i. In that case Step 3 has to be restarted.

After all pipeline stages are analyzed and eventually modified, their new delay information is extracted (Step 4). Here it is extremely important to investigate all stages in one step and not only those that have been changed in Step 3. This is due to the fact that as long as one stage is modified, the power consumption and hence the temperature distribution will change, which can affect also the wearout and hence the delay of other stages. If it is detected in Step 4 that a stage, which was previously faster than necessary, is now slower than necessary, the changes leading to this situation will be reverted and the previous implementation will be used. Since these situations are undesired, the delay differences between the new and the old implementation should be very small (see Section IV-C for more details).

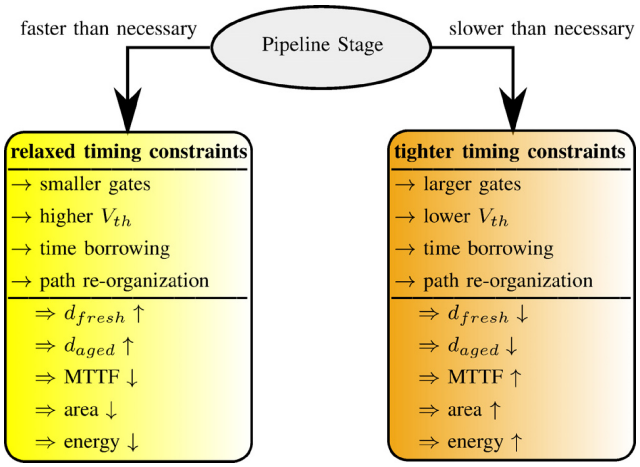

Fig. 5. Pipeline stage modification (faster/slower) required to generate an MTTF-balanced design.

If there is at least one modified stage remaining after Step 4, again Step 3 followed by Step 4 will be executed until no pipeline stage is modified anymore, i.e., until no stage can be tuned further. When this saturation state is reached, the transformation to the MTTF-balanced design is finished.

Note that in some application areas it might be more important to minimize the die area or energy consumption, instead of performance (clock frequency). In that case, the transformation procedure is very similar to the one explained before. The only difference is that d' is used as reference delay in place of d^*. Hence, no stage will be accelerated. Instead all stages, beside the one that is critical at t_{target}, will be designed slower, hence with less energy and area consumption.

The flow can also accept a given clock target instead of a lifetime target to find the MTTF-balanced design with the best MTTF. In this case d^* is replaced with $\tilde{d}$ and t_{target} is set according to the lifetime of the design-time critical stage given the delay target $\tilde{d}$. If during the optimization phase a pipeline stage cannot satisfy the given clock target (slower than necessary), t_{target} will be reduced to the lifetime of this stage (instead of adjusting d^* as shown in Step 3) and the transformation process is restarted (Step 3).

B. Modification (Faster/Slower) of Pipeline Stage

A crucial part of the previously presented transformation flow is the modification of a pipeline stage, i.e., the step to generate a faster or slower version of a pipeline stage. As already mentioned, we use the synthesis tool for this purpose. Therefore, the timing constraints are tightened or relaxed (e.g., by 1 %) and then the pipeline stage is resynthesized using the new timing constraints, while all other constraints are kept the same. To match the new timing constraints the synthesis tool applies gate-sizing (smaller gates for relaxed constraints, larger gates for tighter constraints), path reorganization as well as time borrowing techniques, and also the transistor threshold voltage can be tuned (lower V_{th} for relaxed constraints, higher V_{th} for tighter constraints) as illustrated in Fig. 5. Hence, there are many different ways to obtain an optimized design, e.g., with and without V_{th}-tuning. As a consequence, it is possible

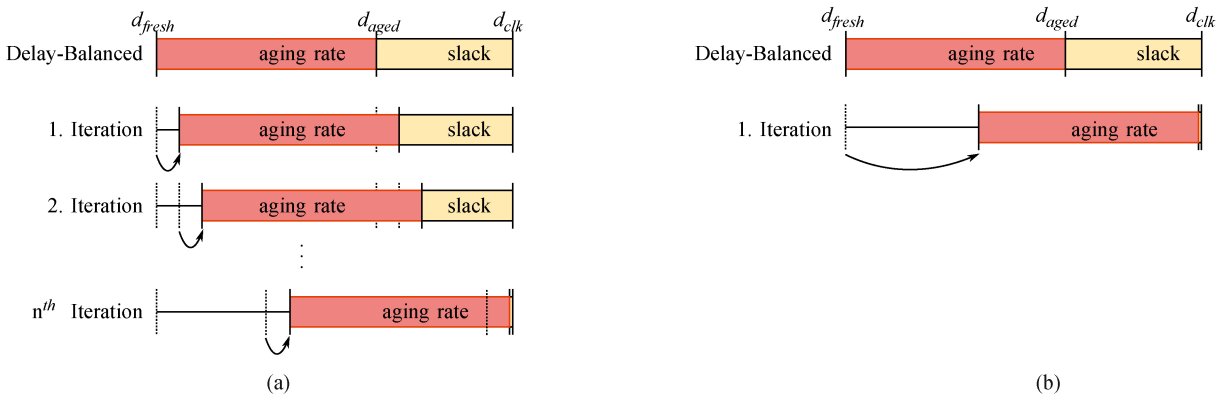

Fig. 6. Simple and enhanced transformation flow for a single pipeline stage (that can have weaker timing constraints) from delay-balanced to MTTF-balanced. (a) Simple transformation using a fixed step width. (b) Enhanced transformation using non-uniform steps.

that there are several different designs for the same pipeline stage that have different delays at t_{design} but the same MTTF. For example the Fetch1 stage of FabScalar can achieve a lifetime of seven years in two different ways: 1) using the nominal V_{th} and a fresh delay of 1.38 ns, and 2) with a higher V_{th} and a fresh delay of 1.40 ns (see Section VI for more details). In such scenarios, it is of course the question, which design should be chosen for the final microprocessor. Therefore, to select one final design, the other design parameters such as area and energy consumption are used and for example the design with the best energy consumption can be chosen. This way, lifetime, energy consumption and area can be co-optimized. However, generating more versions of the same pipeline stage increases the transformation time. Hence, the number of generated versions also depends on the time budget of the manufacturer or designer.

C. Runtime Improvements

The runtime for the transformation process from a delay-balanced to an MTTF-balanced pipeline is proportional to the number of necessary iterations, i.e. the number of synthesis steps and delay/MTTF estimation steps. Hence, to reduce runtime, the number of iterations has to be reduced.

In our preliminary work [37], we used a fixed resolution of 0.01 ns for each iteration with which the delay constraints were tightened or relaxed, as shown in Fig. 6(a). However, such a uniform step width can lead to a huge number of iterations until the final MTTF-balanced implementation is found. For example, the delay constraint for FabScalar's retire stage could be relaxed by a total of 0.1 ns, which corresponds to at least ten iterations considering all pipeline stages.

To improve the runtime of the transformation process, we propose to use a non-uniform resolution as shown in Fig. 6. We observed that tighter or weaker timing constraints have only a weak effect on the delay degradation itself. For example, if the delay degradation of a pipeline stage is roughly 9 % after three years, the gate-level modification to have a faster/slower version of this stage do not affect this value very much. This is reasonable, since gate-sizing or reorganization of just a few paths does not affect the majority of the internal signals and hence the aging rate is not significantly affected [15]. Hence, the aging rate of the original, delay-balanced version of a pipeline stage can be used to estimate its design time delay (i.e., timing constraints) for the MTTF-balanced version, according to the following equation:

$$d_{fresh}^{new} = d_{clk} \cdot \left(1 + \frac{d_{aged}^{orig}}{d_{fresh}^{orig}}\right)^{-1} = \frac{d_{clk}}{\text{wearout rate}} \tag{3}$$

where d_{clk} is the targeted clock period, d_{fresh}^{new} and d_{fresh}^{orig} are the design time delays of the modified and delay-balanced pipeline stage, respectively, and d_{aged}^{orig} is the aged delay (here, after three years) of the delay-balanced version. Using this estimation, the timing constraints for resynthesis are set and the design is optimized accordingly. Then, it is evaluated whether the new design matches the MTTF-balanced criteria or not. In the latter case, the design is tuned further using a fixed step size. By this means, it is possible to significantly reduce the number of iterations. For example, in case of FabScalar the number of iterations is reduced from ten to three, which corresponds to a three times more runtime improvement.

Besides the number of iterations also the runtime of a single iteration step is crucial for the overall runtime. To keep this as low as possible, the pipeline stages are not fully synthesized every time an optimization step is performed. Instead, an intermediate representation of the last version is stored in form of a `ddc`-file (Synopsys database format), which is then used for further optimizations. As the ddc-file contains already the gate-level design with all optimizations to match the current timing constraints, the initial synthesis from a behavioral to a gate-level description as well as basic optimization are avoided, which improves runtime furthermore. Overall these optimizations can reduce the runtime of a single iteration step to less than 1 min for a single pipeline stage, if the runtime for post-synthesis simulations is not considered. Hence, the runtime for one iteration is less than 10 min for the entire FabScalar processor considering all 11 pipeline stages and even less than 5 min for OpenSPARC, as OpenSPARC has just six pipeline stages. As a result, the overall runtime for the transformation flow is dominated by the post-synthesis simulations, which are part of the aging estimation step.

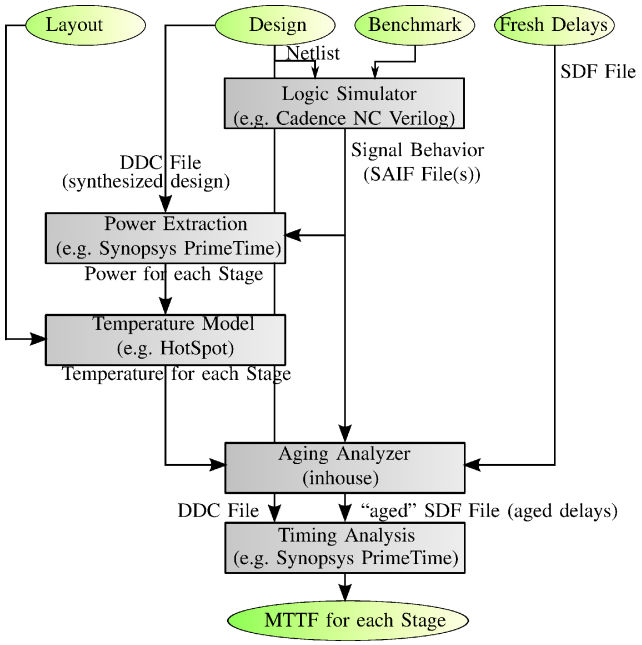

Fig. 7. Flow for extracting MTTF for each pipeline stage.

V. MTTF-Estimation Flow

To accurately evaluate the aging rates (delay changes) and MTTF values for each pipeline stage, a suitable analysis flow is necessary. In this section a generic flow is described, which is based on standard industrial design tools (see Table II).

As shown in the Section II, the aging rate of a transistor strongly depends on its duty cycle, switching activity and temperature. Hence, it is very important to accurately calculate these values. Therefore, the first step of the estimation flow, depicted in Fig. 7, is to generate a gate-level description (netlist) of the pipeline stage under investigation. Afterward, gate-level simulations are performed to obtain the properties of all internal signals for the evaluated stage. This data is used to extract first the energy consumption and then the temperature behavior of this stage. Since two neighboring stages affect the temperature behavior of each other, the entire processor has to be considered during the last step. The next step is to extract the delay degradation for the evaluated pipeline stage. Therefore, the temperature information and the signal behaviors (duty cycles, switching activities) are given to an in-house aging estimation tool. This tool accurately calculates the delay degradation for each gate based on gate-level models similar to the ones presented in [23]. Therefore, for each gate the duty cycle and switching activity for each transistor inside this gate is obtained considering the stacking effect [47]. Later, the threshold voltage shift for each transistor using the transistor-level aging models described in Section II is estimated. By that means the delay degradation for each gate can be obtained. With the help of a standard delay format (SDF) file containing the design time delay information for all gates, an "aged" SDF file is generated that is based on the degraded gate delays. Finally, this "aged" SDF is read by the synthesis tool, the design is annotated accordingly and a new timing report is extracted which contains the most critical paths of the aged design. Together with the information about the most critical path at design time and the given clock period the aging rate as well as MTTF for the entire stage can be extracted.

TABLE II
Tools used for Result Extraction

Synthesis + Timing Estimation	Synopsys Design Compiler D-2010.03-SP4
Simulation + SAIF-Generation	Cadence NC Sim 12.10-s005
Power Extraction	Synopsys PrimeTime D-2010.03-SP4
Temperature Extraction	HotSpot 5.02 [19]
Aging Analysis	Inhouse C++-Tool

A. Runtime Improvements

Compared to our preliminary flow presented in [37], this new flow is enhanced to provide a better accuracy (all paths are considered instead of only the top most critical ones), a shorter runtime (minutes instead of days) and to require less computing resources.

Due to the low runtime for the aging estimation step itself, the overall runtime for the aging estimation flow is dominated by the time required to perform post-synthesis simulations for a sufficiently large number of clock cycles, which can take up to several hours for very complex pipeline stages. In this section, we will discuss two possibilities to eliminate these costly simulations and how the accuracy of the aging estimation is impacted using these techniques.

Since the post-synthesis simulations are performed to extract the (average) signal properties over a long period of time, the first approach is to use a default annotation (e.g., duty cycle = 0.5, switching activity = 0.01) for all primary inputs of the pipeline stage under investigation and to propagate these information through the remaining design to get the signal properties for all internal signals. In contrast, the second technique uses the real signal properties for all primary inputs, outputs and flipflops (extracted during higher-level simulation steps) and propagates these information through the remaining design. In both ways, the costly post-synthesis simulations can be avoided and the synthesis tool can be used to perform the signal property propagation to extract the signal behaviors for the entire design. However, the cost for the speedup (several orders of magnitude: seconds versus minutes or hours) are inaccurate signal properties compared to post-synthesis simulations as the signal property propagation is never 100 % accurate [25]. As a result, the aging estimation using these two techniques will be inaccurate. In fact, as shown in Fig. 8 the inaccuracy for the first approach can reach almost 6 % in case of the FabScalar microprocessor which corresponds to an inaccuracy of more than ten times in terms of MTTF. In contrast, the second approach is much more accurate (less than 1 % deviation), which is due to the fact, that real data is used to annotate the design. Therefore, this approach can be chosen whenever some small inaccuracy is acceptable or post-synthesis simulations are not feasible.

Note that the high-level simulations that are necessary for the second technique are always performed during the typical design-flow and hence do not need to be conducted additionally. For example during the design of the microarchitecture

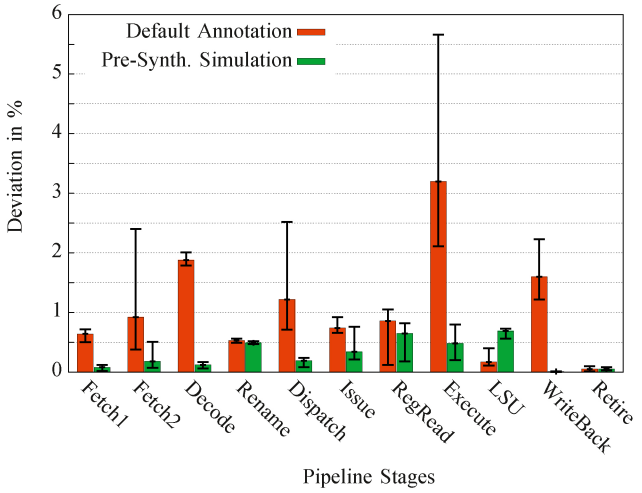

Fig. 8. Inaccuracy (min, avg, max for six SPEC2000 benchmarks) in terms of aging rate of the default annotation ($\delta = 0.5$, $\alpha = 0.01$) and a presynthesis simulation compared to comprehensive post-synthesis simulations for the FabScalar microprocessor.

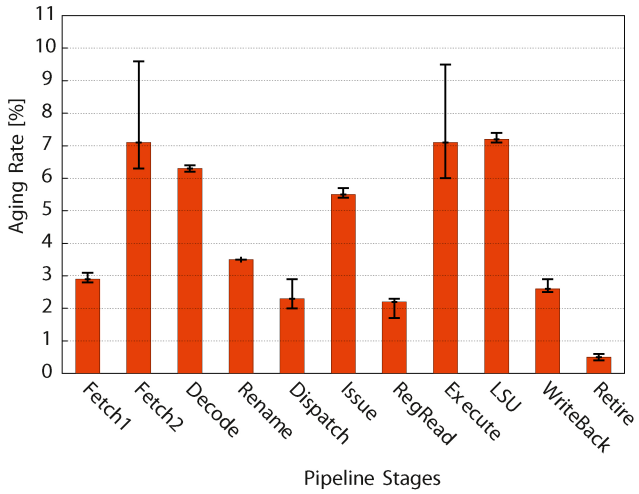

Fig. 9. Aging rates (i.e., delay after three years over design time delay) of different pipeline stages of the FabScalar microprocessor for six SPEC2000 benchmarks (min, max, avg).

TABLE III
ARCHITECTURE COMPARISON OF FABSCALAR AND OPENSPARC T1

	FabScalar [12]	OpenSPARC T1 [1]
Frequency	740 Mhz	1140 Mhz
Architecture	out-of-order	in-order
Pipeline Stages	11	6
Simultaneaous Multithreading	no	4-way
Frontend-Width (per Thread)	4 insts/cycle	1 inst/cycle
Exec. Units (ALU/MUL/AGEN)	1/1/1	1/1/1

or the verification such simulations are performed. Hence, no additional runtime is required, only the necessary data needs to be stored for the future.

VI. EXPERIMENTAL RESULTS

In this section a comparison using the FabScalar microprocessor [12] and OpenSPARC T1 [1] between the proposed MTTF-balanced designs paradigm and the classical delay-balanced one is presented. Using these two microprocessors, we can confirm that the proposed approach is applicable to a wide range of microprocessor designs, as these are representatives of complementary microprocessor families as shown in Table III. The MTTF-balanced designs were generated using the flow presented in Section IV and the TSMC 65 nm library. To have a fair comparison in terms of energy and area, we used the optimization target to get the best MTTF for a given clock frequency, which is used by both the MTTF-balanced and the delay-balanced design. Thereby, the clock period is given by the longest of all pipeline stage delays plus an additional safety margin of 10 % to avoid timing failures due to aging. The ambient processor temperature was set to 40 °C resulting in processor temperatures between 50 °C and 75 °C, which is reasonable for modern processors. Since the application choice for the aging estimation step is crucial as shown in Fig. 9, we evaluated in all our experiments for FabScalar six different SPEC2000 benchmarks (bzip, gap, gzip, mcf, parser, and vortex) provided with FabScalar and simulated the processor behavior for 10^6 cycles after a warmup. For OpenSPARC T1 we used the regression test suite that comes with the simulation environment. Table IV summarizes the main results for FabScalar and Table V the ones for OpenSPARC. In these tables, as well as in the rest of this section, for the sake of simplicity, we present just the worst-case delays and MTTFs for the different application tests as well as the average energy consumption over the used benchmarks.

A. Optimization for FabScalar

The minimum clock period for FabScalar was 1.35 ns (equal to a maximum clock frequency of 740 MHz), limited by the LSU, which is the most complex unit of this microprocessor. Hence, given a margin of 10 %, the clock target was 1.48 ns. For these settings the standard delay-balanced design will fail after three years as depicted in Fig. 10. After seven years, the overall degradation reaches already 12.5 %, and after ten years the delay increase is around 14 %. In contrast, our proposed MTTF-balanced approach is able to achieve a MTTF of seven years (2.3 times improvement). Therefore, the Fetch2 and Execute stage were designed faster (with less design time delay) using the synthesis optimization detailed in Section IV-B. All other stages were designed with less slack (i.e., slower). Therefore, a higher threshold voltage (20 % increase in V_{th}) in addition to the synthesis optimization techniques could be applied to all stages apart from the Issue stage, which could not match the timing constraints if high-V_{th} transistors were used in the (near-)critical paths. By this means, the average energy consumption over all benchmarks (extracted with Synopsys PrimeTime) of the MTTF-balanced design is 10 % lower than that of the traditional delay-balanced pipeline for a clock period of 1.48 ns. Furthermore, a higher V_{th} also reduces the wearout rates, which can be used to achieve even higher energy savings. In addition, the area is reduced by 2 % if the MTTF-balanced version is employed.

If the threshold voltage is not increased, the savings are much smaller, i.e., 1 % and 2 % for energy and area, respec-

TABLE IV

COMPARISON OF A DELAY-BALANCED AND MTTF-BALANCED DESIGN FOR THE FABSCALAR MICROPROCESSOR IN TERMS OF DESIGN TIME DELAY, WORST CASE MTTF, AVG. ENERGY CONSUMPTION (WITHOUT SRAM) AND AREA (WITHOUT SRAM) CONSIDERING ALL BENCHMARKS (GS = GATE SIZING, HVT = HIGHER THRESHOLD VOLTAGE IN THE CRITICAL PATH)

Stage	Delay-Balanced					MTTF-Balanced										Changes
						Nominal V_{th}					V_{th}-Tuning					
	delay		MTTF	Energy	Area	delay		MTTF	Energy	Area	delay		MTTF	Energy	Area	
	@t_{design} [ns]	@t_{3y} [ns]	(1.48 ns) [years]	[μJ]	[μm^2]	@t_{design} [ns]	@t_{3y} [ns]	(1.48 ns) [years]	[μJ]	[μm^2]	@t_{design} [ns]	@t_{3y} [ns]	(1.48 ns) [years]	[μJ]	[μm^2]	
Fetch1	1.35	1.39	45	12.2	23262	1.38	1.46	7	12.0	25338	1.40	1.46	7	7.3	22459	GS, HVT
Fetch2	1.35	1.48	3	16.6	35030	1.33	1.44	7	16.8	36482	1.33	1.44	7	16.8	36482	GS
Decode	1.34	1.43	16	6.1	24568	1.36	1.45	7	6.1	16697	1.42	1.46	7	2.8	17000	GS, HVT
Rename	1.33	1.38	50+	1.3	4049	1.33	1.38	50+	1.3	4049	1.43	1.46	7	0.7	3475	GS, HVT
Dispatch	1.32	1.37	20	0.1	1867	1.32	1.37	20	0.1	1867	1.41	1.46	7	0.1	1861	GS, HVT
Issue	1.35	1.43	18	14.3	30719	1.38	1.44	7	13.4	29157	1.38	1.44	7	13.4	29157	GS
RegRead	1.33	1.36	50+	2.1	12061	1.33	1.36	50+	2.1	12061	1.43	1.46	7	1.5	12074	GS, HVT
Execute	1.35	1.48	3	6.4	27529	1.33	1.45	7	6.5	28959	1.33	1.45	7	6.5	28959	GS
LSU	1.35	1.45	7	50.6	107664	1.35	1.45	7	50.6	107664	1.35	1.45	7	50.6	107664	none
WriteBack	1.32	1.36	30	4.0	3193	1.32	1.36	30	4.0	3193	1.43	1.46	7	2.5	3174	GS, HVT
Retire	1.35	1.36	50+	1.6	3201	1.36	1.37	50+	1.5	2579	1.45	1.46	7	1.0	2562	GS, HVT
Overall	1.35	1.48	3	115.4	273133	1.38	1.46	7	114.5	268406	1.45	1.46	7	103.4	268746	
						+2 %	-2 %	+233 %	-1 %	-2 %	+7 %	-2 %	+233 %	-10 %	-2 %	

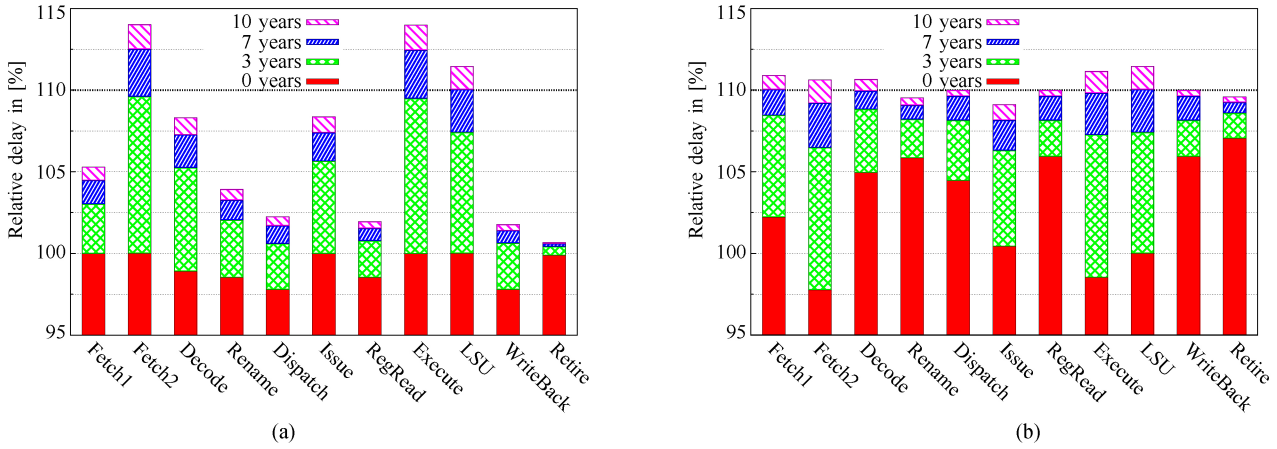

Fig. 10. Delay degradation of the delay-balanced design and MTTF-balanced design for the FabScalar microprocessor. (a) Delay-balanced design. (b) MTTF-balanced design.

tively. This is mainly due to the fact that some of the pipeline stages cannot be designed any slower without using a higher V_{th}. The reason is the academic nature of FabScalar due to which the original design is not efficiently balanced.

Note that the energy and area savings are not just positive side-effects, but are due to the optimization process. As most pipeline stages are not aging-critical, energy consumption and area usage can be reduced for the majority of the pipeline stages resulting in overall energy and area savings.

If an increased MTTF is of secondary interest, it is possible to reduce the clock period of the MTTF-balanced design from 1.48 ns to 1.46 ns. Still the MTTF-balanced design has a lifetime of three years, but the performance compared to the delay-balanced design will increase by more 2 %. In addition, energy and area consumption will also be lower as compared to the delay-balanced version.

An interesting observation of our results is that the Fetch2 (predecode) and the Execute stage have a very similar aging behavior, although their (microarchitecture-level) functionalities are totally different. The reason is that the delay degradation of both stages is very sensitive to number of stall cycles that appear during the application execution (the higher the stall ratio, the faster these stages wear out), while other stages are less affected by these cycles. This can be explained in the following way. During stall cycles the pipeline stage inputs remain constant, which means that also all internal signals are constant for a longer period of time. If many transistors inside the critical paths are under stress during these cycles, the delay degradation is accelerated. The worst results (degradation of almost 10 % in three years) were observed for the mcf benchmark, which had a stall ratio of 70 % for the pipeline front- and backend, while others, such as the parser benchmark, just had a stall ration of 10 % and caused a much reduced delay degradation (less than 7.5 %). In contrast, a correlation between wearout rates and instructions per cycle (IPC) could not be observed.

B. Optimization for OpenSPARC T1

OpenSPARC T1 is the open source clone of the industrial UltraSPARC T1 (*Niagara*) processor developed by Sun and

TABLE V

COMPARISON OF A DELAY-BALANCED AND MTTF-BALANCED DESIGN FOR THE OPENSPARC T1 MICROPROCESSOR IN TERMS OF DESIGN TIME DELAY, WORST CASE MTTF, AVG. ENERGY CONSUMPTION (WITHOUT SRAM) AND AREA (WITHOUT SRAM) CONSIDERING ALL BENCHMARKS (GS = GATE SIZING, HVT = HIGHER THRESHOLD VOLTAGE IN THE CRITICAL PATH)

Stage	Delay-Balanced					MTTF-Balanced										Changes
						Nominal V_{th}					V_{th}-Tuning					
	delay		MTTF	Energy	Area	delay		MTTF	Energy	Area	delay		MTTF	Energy	Area	
	@t_{design} [ns]	@t_{3y} [ns]	(0.96 ns) [years]	[μJ]	[μm^2]	@t_{design} [ns]	@t_{3y} [ns]	(0.96 ns) [years]	[μJ]	[μm^2]	@t_{design} [ns]	@t_{3y} [ns]	(0.96 ns) [years]	[μJ]	[μm^2]	
Fetch	0.88	0.96	3	398.8	68547	0.85	0.92	10	399.0	68541	0.85	0.92	10	399.0	68541	GS
Pick	0.87	0.92	20	22.2	4139	0.89	0.94	10	22.0	4138	0.92	0.94	10	9.9	4204	GS, HVT
Decode	0.87	0.90	50+	7.4	1611	0.91	0.94	10	7.3	1610	0.91	0.94	10	3.3	1655	GS, HVT
Execute	0.88	0.96	3	289.0	56142	0.85	0.92	10	291.1	56353	0.85	0.92	10	291.1	56353	GS
LSU	0.88	0.90	50+	116.4	18307	0.91	0.93	10	116.3	18300	0.91	0.94	10	34.8	16381	GS, HVT
WriteBack	0.88	0.94	10	246.3	58987	0.88	0.94	10	246.3	58987	0.88	0.94	10	246.3	58987	none
Overall	0.88	0.96	3	1080.1	207733	0.91 +5 %	0.94 -2 %	10 +333 %	1081.9 +0 %	207929 +0 %	0.92 +5 %	0.94 -2 %	10 +333 %	984.4 -9 %	206121 -1 %	

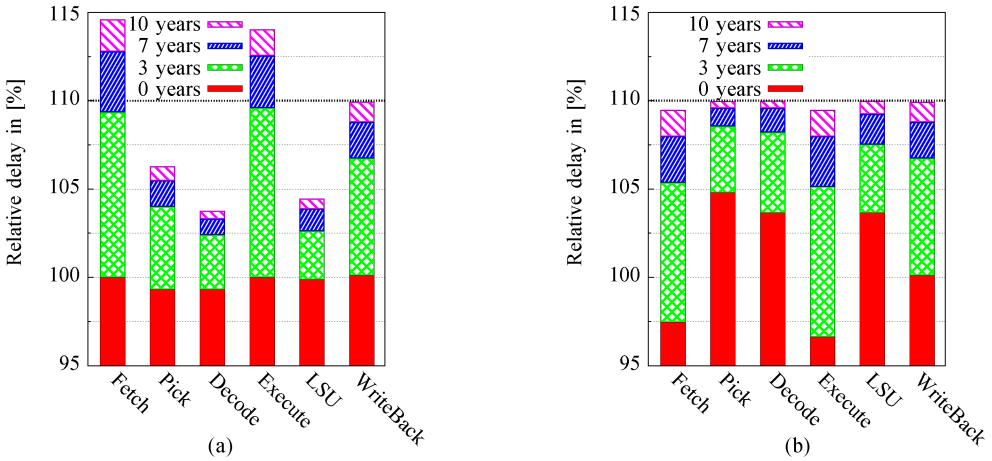

Fig. 11. Delay degradation of the delay-balanced design and MTTF-balanced design for the OpenSPARC T1 microprocessor. (a) Delay-balanced design. (b) MTTF-balanced design.

released in 2005 [24]. Hence, the maximum clock frequency is much higher than for the academic FabScalar, i.e., we could operate OpenSPARC at 1140 MHz or a minimum clock period of 0.88 ns using the TSMC 65 nm library limited by the WriteBack stage. Furthermore, the original design is much more balanced in terms of delay than the one for FabScalar.

Using the standard delay-balanced design OpenSPARC can achieve a lifetime of three years for a timing margin of 10 %, while for a lifetime of ten years already a guardband of 15 % is necessary as illustrated in Fig. 11. In contrast, using our proposed MTTF-balanced design paradigm the MTTF can be extended from 3 years to 10 years, i.e., MTTF is improved by more than three times (for 10 % timing margin). Therefore, the Fetch and Execute stages had to be designed with more design time slack, while all other stages were designed slower to save area and energy. However, without tuning the threshold voltage the savings for the remaining stages only compensate the energy and area costs due to the faster version of the Fetch and Execute stage. With a higher threshold voltage the energy consumption can be reduced by 10 % and area by 1 %. This is especially obvious for the LSU, where energy can be reduced by almost four times by using a higher threshold voltage.

Similar to FabScalar, the clock frequency can be increased by 2 %, if MTTF is kept the same as for the delay-balanced design. Hence, also here the gained headroom can be used to boost the performance.

C. Comparison of FabScalar and OpenSPARC T1

As explained in the previous sections, the lifetime for both FabScalar and OpenSPARC can be significantly extended using our proposed MTTF-balanced design paradigm, with better results for OpenSPARC. However, in general, it cannot be concluded that our technique is more efficient for simple, lightweight cores, since the two investigated processors as well as the used workloads are very different. This is also the reason why a direct comparison of the aging rates between FabScalar and OpenSPARC cannot be performed. Nevertheless, a few conclusions can be drawn by the numbers presented in this paper. For both architectures, it is the execution stage which is aging-critical. Moreover, for both designs the delay degradation for the execution stage is very similar. This is due to the fact that similar ALUs were used, and that the temperature as well as signal probabilities in the critical paths were in a similar range. Furthermore, we observed for both designs that the aging rate of a pipeline stage is not very sensitive to its

design time delay. In other words, the aggressiveness with which a pipeline stage is designed seems to have only a small effect on its aging rate. In fact, the functionality, the workload and the temperature are far more important in terms of aging.

VII. Conclusion

Microprocessors at nano-scale are exposed to various reliability issues, which include a more rapid aging of all components. This leads to increasing pipeline stage delays during the operational lifetime, resulting in imbalanced designs in terms of delay and MTTF, if the delays are balanced at design time. In this paper, we have shown that this imbalance hides a lot of optimization potential for higher clock frequencies, longer lifetimes (i.e., higher MTTF) as well as reduced power and area consumption.

Therefore, we proposed a radically new, aging-aware MTTF-balanced pipeline design scheme to replace the traditional delay-balanced paradigm. Using the new approach, the imbalance during runtime is minimized, allowing better designs. Our experimental results show that for the FabScalar microprocessor, the MTTF-balanced design yields a more than 2.3 times longer MTTF, while the same performance (i.e., frequency) as for the delay-balanced design can be maintained. For OpenSPARC T1 the lifetime can even be increased by more than three times with no negative impacts on performance and area. Moreover, for both deigns the average energy consumption can be reduced by almost 10 %.

References

[1] Oracle. (2013, Jul.). *Opencores: OpenSPARC Overview* [Online]. Available: http://www.oracle.com/technetwork/systems/opensparc/index.html

[2] J. Abella, X. Vera, and A. Gonzalez, "Penelope: The NBTI-aware processor," in *Proc.40th Annu. IEEE/ACM Int. Symp. Microarchitec.*, 2007, pp. 85–96.

[3] C. Auth *et al.*, "A 22 nm high performance and low-power CMOS technology featuring fully-depleted tri-gate transistors, self-aligned contacts and high density MIM capacitors," in *Proc. Symp. VLSI Technol.*, 2012, pp. 131–132.

[4] M. Basoglu, M. Orshansky, and M. Erez, "NBTI-aware DVFS: A new approach to saving energy and increasing processor lifetime," in *Proc. 16th ACM/IEEE Int. Symp. Low Power Electron. Design*, 2010, pp. 253–258.

[5] K. Bernstein, D. J. Frank, A. E. Gattiker, W. Haensch, B. L. Ji, *et al.*, "High-performance CMOS variability in the 65-nm regime and beyond," *IBM J. Res. Develop.—Adv. Silicon Technol.*, vol. 50, pp. 433–449, Jul. 2006.

[6] D. R. Bild, G. E. Bok, and R. P. Dick, "Minimization of NBTI performance degradation using internal node control," in *Proc. Conf. DATE*, 2009, pp. 148–153.

[7] S. Borkar, "Designing reliable systems from unreliable components: The challenges of transistor variability and degradation," *IEEE Micro*, vol. 25, no. 6, pp. 10–16, Nov./Dec. 2005.

[8] A. Bravaix, C. Guerin, V. Huard, D. Roy, J. Roux, and E. Vincent, "Hot-carrier acceleration factors for low power management in DC-AC stressed 40 nm NMOS node at high temperature," in *Proc. IEEE IRPS.*, 2009, pp. 531–548.

[9] A. Calimera, E. Macii, and M. Poncino, "NBTI-aware power gating for concurrent leakage and aging optimization," in *Proc. 14th ACM/IEEE ISPLED*, Aug. 2009, pp. 127–132.

[10] T. Chan, J. Sartori, P. Gupta, and R. Kumar, "On the efficacy of NBTI mitigation techniques," in *Proc. Conf. DATE*, 2011, pp. 1–6.

[11] J. Chen, S. Wang, and M. Tehranipoor, "Efficient selection and analysis of critical-reliability paths and gates," in *Proc. Great Lakes Symp. VLSI*, 2012, pp. 45–50.

[12] N. Choudhary, S. Wadhavkar, T. Shah, H. Mayukh, J. Gandhi, B. Dwiel, *et al.*, "FabScalar: Automating superscalar core design," *IEEE Micro*, vol. 32, no. 3, pp. 48–59, May 2012.

[13] O. Coudert, "Gate sizing for constrained delay/power/area optimization," *IEEE Trans. Very Large Scale (VLSI) Syst.*, vol. 5, no. 4, pp. 465–472, Dec. 1997.

[14] M. DeBole, R. Krishnan, V. Balakrishnan, W. Wang, H. Luo, Y. Wang, *et al.*, "New-Age: A negative bias temperature instability-estimation framework for microarchitectural components," *Int. J. Parallel Program.*, vol. 37, no. 4, pp. 417–431, Aug. 2009.

[15] M. Ebrahimi, F. Oboril, and M. B. Tahoori, "Aging-aware logic synthesis," in *Proc. 2013 IEEE/ACM Int. Conf. Comput.-Aided Design*, 2013, pp. 1–8.

[16] F. Firouzi, S. Kiamehr, and M. B. Tahoori, "NBTI mitigation by NOP assignment and insertion," in *Proc. Conf. DATE*, 2012, pp. 218–223.

[17] E. Gunadi, A. A. Sinkar, N. S. Kim, and M. H. Lipasti, "Combating aging with the colt duty cycle equalizer," in *Proc. 43rd Annu. IEEE/ACM Int. Symp. Microarchitec.*, 2010, pp. 103–114.

[18] J. Hennessy and D. Patterson, *Computer Architecture: A Quantitative Approach*. San Mateo, CA, USA: Morgan Kaufmann, 2011.

[19] W. Huang *et al.*, "HotSpot: A compact thermal modeling methodology for early-stage VLSI design," *IEEE Trans. Very Large Scale (VLSI) Syst.*, vol. 14, no. 5, pp. 501–513, May 2006.

[20] T. Karnik, Y. Ye, J. Tschanz, L. Wei, S. Burns, V. Govindarajulu, *et al.*, "Total power optimization by simultaneous dual-vt allocation and device sizing in high performance microprocessors," in *Proc. 39th Annu. DAC*, 2002, pp. 486–491.

[21] JJ. Keane, T. Kim, X. Wang, and C. H. Kim, "On-chip reliability monitors for measuring circuit degradation," *Microelectron. Reliab.*, vol. 50, no. 8, pp. 1039–1053, 2010.

[22] O. Khan and S. Kundu, "A self-adaptive system architecture to address transistor aging," in *Proc. DATE*, 2009, pp. 81–86.

[23] S. Kiamehr, F. Firouzi, and M. B. Tahoori, "Input and transistor reordering for NBTI and HCI reduction in complex CMOS gates," in *Proc. GLSVLSI*, 2012, pp. 201–206.

[24] P. Kongetira, K. Aingaran, and K. Olukotun, "Niagara: A 32-way multithreaded Sparc processor," *IEEE Micro*, vol. 25, no. 2, pp. 21–29, Mar./Apr. 2005.

[25] B. Krishnamurthy and I. G. Tollis, "Improved techniques for estimating signal probabilities," *IEEE Trans. Comput.*, vol. 38, no. 7, pp. 1041–1045, Jul. 1989.

[26] S. V. Kumar, C. H. Kim, and S. S. Sapatnekar, "Impact of NBTI on SRAM read stability and design for reliability," in *Proc. 7th ISQED*, 2006, pp. 210–218.

[27] S. V. Kumar, C. H. Kim, and S. S. Sapatnekar, "NBTI-aware synthesis of digital circuits," in *Proc. 44th Annu. DAC*, 2007, pp. 370–375.

[28] Y. Kunitake, T. Sato, and H. Yasuura, "Signal probability control for relieving NBTI in SRAM cells," in *Proc. 11th ISQED*, 2010, pp. 660–666.

[29] X. Liang, G. Wei, and D. Brooks, "ReVIVal: A variation-tolerant architecture using voltage interpolation and variable latency," *IEEE Micro*, vol. 29, no. 1, pp. 127–138, Jan./Feb. 2009.

[30] E. Mintarno, V. Chandra, D. Pietromonaco, R. Aitken, and R. Dutton, "Workload dependent NBTI and PBTI analysis for a sub-45nm commercial microprocessor," in *Proc. IEEE IRPS*, 2013, pp. 3A.1.1–3A.1.6.

[31] E. Mintarno, J. Skaf, R. Zheng, J. Velamala, Y. Cao, S. Boyd, *et al.*, "Self-tuning for maximized lifetime energy-efficiency in the presence of circuit aging," *IEEE Trans. Computer-Aided Design Int. Circuits Syst.*, vol. 30, no. 5, pp. 760–773, May 2011.

[32] S. Mittl, A. Swift, E. Wu, D. Ioannou, F. Chen, G. Massey, *et al.*, "Reliability characterization of 32 nm high-K metal gate SOI technology with embedded DRAM," in *Proc. IEEE IRPS*, 2012, pp. 6A.5.1–6A.5.7.

[33] V. Narayanan and Y. Xie, "Reliability concerns in embedded system designs," *Computer*, vol. 39, no. 1, pp. 118–120, 2006.

[34] F. Oboril, F. Firouzi, S. Kiamehr, and M. B. Tahoori, "Negative bias temperature instability-aware instruction scheduling: A cross-layer approach," *J. Low Power Electron.*, vol. 9, no. 4, pp. 389–402, 2013.

[35] F. Oboril and M. B. Tahoori, "ExtraTime: Modeling and analysis of wearout due to transistor aging at microarchitecture-level," in *Proc. 42nd Annu. IEEE/IFIP Int. Conf. DSN*, 2012, pp. 1–12.

[36] F. Oboril and M. B. Tahoori, "Reducing wearout in embedded processors using proactive fine-grained dynamic runtime adaptation," in *Proc. 17th IEEE Eur. Test Symp.*, 2012, pp. 68–73.

[37] F. Oboril and M. B. Tahoori, "MTTF-balanced pipeline design," in *Proc. Conf. DATE*, 2013, pp. 270–275.

[38] F. Oboril and M. B. Tahoori, "ArISE: Aging-aware instruction set encoding for lifetime improvement," in *Proc. ASPDAC*, 2014, pp. 1–6.

[39] S. Pae, M. Agostinelli, M. Brazier, R. Chau, G. Dewey, T. Ghani, *et al*., "BTI reliability of 45 nm high-k + metal-gate process technology," in *Proc. IEEE IRPS*, 2008, pp. 352–357.
[40] B. C. Paul, K. Kang, H. Kufluoglu, M. A. Alam, and K. Roy, "Temporal performance degradation under NBTI: Estimation and design for improved reliability of nanoscale circuits," in *Proc. Conf. DATE*, 2006, pp. 780–785.
[41] J. Sartori, B. Ahrens, and R. Kumar, "Power balanced pipelines," in *Proc. IEEE 18th Int. Symp. HPCA*, 2012, pp. 1–12.
[42] T. Siddiqua and S. Gurumurthi, "A Multi-level approach to reduce the impact of NBTI on processor functional units," in *Proc. GLSVLSI*, 2010, pp. 67–72.
[43] K. Skadron, M. R. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan, "Temperature-aware microarchitecture," *SIGARCH Comput. Architec. News*, vol. 31, no. 2, pp. 2–13, May 2003.
[44] E. Takeda, Y. Nakagome, H. Kume, and S. Asai, "New hot-carrier injection and device degradation in submicron MOSFETs," *IEEE Proc. I, Solid-State Electron. Devices*, vol. 130, no. 3, pp. 144–150, Jun. 1983.
[45] A. Tiwari, S. R. Sarangi, and J. Torrellas, "ReCycle: Pipeline adaptation to tolerate process variation," *SIGARCH Comput. Architec. News*, vol. 35, no. 2, pp. 323–334, 2007.
[46] A. Tiwari and J. Torrellas, "Facelift: Hiding and slowing down aging in multicores," in *Proc. 41st Annu. IEEE/ACM Int. Symp. Microarchitec.*, 2008, pp. 129–140.
[47] R. Vattikonda, W. Wang, and Y. Cao, "Modeling and minimization of PMOS NBTI effect for robust nanometer design," in *Proc. 43rd Annu. DAC*, 2006, pp. 1047–1052.
[48] S. Wang, T. Jin, C. Zheng, and G. Duan, "Low power aging-aware register file design by duty cycle balancing," in *Proc. DATE*, 2012, pp. 546–549.
[49] W. Wang, V. Reddy, A. T. Krishnan, R. Vattikonda, S. Krishnan, and Y. Cao, "Compact modeling and simulation of circuit reliability for 65-nm CMOS technology," *IEEE Trans. Device Mater. Reliab.*, vol. 7, no. 4, pp. 509–517, Dec. 2007.
[50] Y. Wang, X. Chen, W. Wang, V. Balakrishnan, Y. Cao, Y. Xie, *et al*., "On the efficiancy of Input Vector Control to mitigate NBTI effects and leakage power," in *Proc. Int. Symp. Qual. Electron. Design*, 2009, pp. 19–26.
[51] C. C. Wu, D. W. Lin, A. Keshavarzi, C. H. Huang, C. T. Chan, C. H. Tseng, *et al*., "High performance 22/20nm FinFET CMOS devices with advanced high-K/metal gate scheme," in *Proc. IEEE Int. Electron. Devices Meeting*, 2010, pp. 27.1.1–27.1.4.
[52] X. Yang and K. Saluja, "Combating NBTI degradation via gate sizing," in *Proc. 8th ISQED*, 2007, pp. 47–52.

Fabian Oboril received the Diploma degree in mathematics technology from the Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany, in 2010. He is currently pursuing the Ph.D. degree with the Chair of Dependable Nano-Computing (CDNC) Group, KIT.

Since 2010, he has been a Research Assistant at the CDNC Group of Prof. Tahoori at KIT. His current research interests include the reliability issues of systems build in the nano era including transistor aging, fault tolerant computing, and low-power high-performance microprocessor designs.

Mehdi B. Tahoori received the B.S. degree in computer engineering from Sharif University of Technology, Tehran, Iran in 2000, and the M.S. and Ph.D. degrees in electrical engineering from Stanford University, Stanford, CA, USA, in 2002 and 2003, respectively.

He is currently a Full Professor and Chair of Dependable Nano-Computing at the Department of Computer Science, Institute of Computer Science and Engineering, Karlsruhe Institute of Technology, Karlsruhe, Germany. In 2003, he joined the Electrical and Computer Engineering Department at the Northeastern University, Boston, MA, USA, as an Assistant Professor, and was promoted to the rank of an Associate Professor with tenure in 2009. During 2002–2003, he was a Research Scientist at Fujitsu Laboratories of America, Sunnyvale, CA, in advanced CAD research, focusing on reliability issues in deep-submicron mixed-signal VLSI designs. In addition to five pending and granted U.S. and international patents for his work, he has over 140 publications in major journals and conference proceedings on wide-ranging topics from dependable computing and emerging nanotechnologies to system biology. His current research interests include nano computing, reliable computing, VLSI testing, reconfigurable computing, emerging nanotechnologies, and system biology.

Dr. Tahoori is a recipient of the National Science Foundation Early Faculty Development (CAREER) Award. He has served as the Program Committee member as well as workshop, panel and special session organizer of various conferences and symposia in the areas of VLSI test, reliability, and emerging nanotechnologies, such as ITC, ICCAD, DATE, ETS, ICCD, ASP-DAC, GLSVLSI, and VLSI Design. He is also an Associate Editor of *ACM Journal of Emerging Technologies for Computing* and Chair of ACM SIGDA Technical Committee on Test and Reliability.