Aging-aware Logic Synthesis

Mojtaba Ebrahimi Fabian Oboril Saman Kiamehr Mehdi B. Tahoori
Chair of Dependable and Nano Computing, Karlsruhe Institute of Technology, Karlsruhe, Germany
Email: {mojtaba.ebrahimi, fabian.oboril, kiamehr, mehdi.tahoori}@kit.edu

***Abstract*—As CMOS technology scales down into the nanometer regime, designers have to add pessimistic timing margins to the circuit as guardbands to avoid timing violations due to various reliability effects, in particular accelerated transistor aging. Since aging is workload-dependent, the aging rates of different paths are non-uniform, and hence, design time delay-balanced circuits become significantly unbalanced after some operational time. In this paper, an aging-aware logic synthesis approach is proposed to increase circuit lifetime with respect to a specific guardband. Our main objective is to optimize the design timing with respect to post-aging delay in a way that all paths reach the assigned guardband at the same time. In this regard, in an iterative process, after computing the post-aging delays, the lifetime is improved by putting tighter timing constraints on paths with higher aging rate and looser constraints on paths which have less post-aging delay than the desired guarband. The experimental results shows that the proposed approach improves circuit lifetime in average by more than 3X with negligible impact on area. Our approach is implemented on top of a commercial synthesis toolchain, and hence scales very well.**

I. Introduction

By aggressive technology down-scaling into the nanometer regime, design of robust systems becomes extremely challenging [1, 2]. Among various reliability threats accelerated transistor aging due to *Bias Temperature Instability* (BTI) and *Hot Carrier Injection* (HCI) can significantly affect the reliable operation of nanoscale circuits. Aging can considerably reduce the operational lifetime and even cause timing failures in the field. Moreover, it is predicted that, since supply voltage does not scale at the same pace with the device geometrics, the increased current density and temperature will further accelerate device degradation in future technology nodes [3, 4].

To avoid timing violations due to various reliability effects such as temperature, voltage droop, and accelerated transistor aging, designers add pessimistic timing margins to the circuit as guardbands. In guardbanding, the degradation that may be incurred over the expected lifetime is estimated and circuit operating frequency is reduced accordingly which is eroding gains from technology scaling. For example, a 65 nm technology requires a 20% guardband to avoid aging-induced failures within the first 10 years [3].

Since signal probabilities and switching activities, which affect BTI and HCI, respectively, alongside with temperature vary broadly for different transistors in a circuit, aging rate of different gates and paths are not uniform [3, 5, 6]. Hence, circuit timing, which is conventionally balanced according to the design time ($t = 0$) delay, becomes significantly unbalanced after some operational time ($t >> 0$). As a consequence, the conventional guardbanding technique is costly as it is based on the worst-case critical-path delay in the expected lifetime, while other paths might have enough slack to work properly for longer time. As a result, the latter ones can be designed slower to save area and energy, while the critical ones should be designed faster to increase the circuit lifetime.

Although synthesis techniques can be improved to consider the post-aging delay using pre-characterization of cells for different signal profiles and temperature conditions [7], none of existing commercial synthesis toolchains has such capabilities. Aging-aware gate-sizing is an alternative solution which can be used to balance the circuit timing according to post-aging delays [6, 8–11]. While these techniques can efficiently increase the lifetime without any change in the circuit topology, for the same reason, they have limited aging mitigation capability, as it will be shown later in our experiments.

In this paper, we propose an aging-aware synthesis approach to balance the circuit timing with respect to a specific guardband to improve lifetime. In our approach, after synthesis, post-aging delay computation and optimization are done iteratively. In each round, after computing the post-aging delays, the lifetime is improved at the expense of area by putting tighter timing constraints on paths with higher aging rate. However, the impact on circuit area is compensated by putting looser constraints on paths with less post-aging delay than the desired guarband. As a result, after several rounds of re-synthesis and aging computations, the circuit lifetime is improved with respect to the desired guardband with negligible impact on area.

Unlike previous gate-sizing-based approaches, the proposed flow is built on top of a commercial synthesis toolchain and hence completely relies on the internal mechanisms of commercial synthesis tools (such as gate-sizing, re-structuring, and logic-borrowing) to improve the circuit timing and area in an aging-aware manner. The experimental results on ITC99 benchmarks show that our proposed approach results in 3.36X lifetime improvement while gate-sizing can only gain 1.77X. Moreover, its runtime is comparable with that of conventional guardbanding techniques.

The rest of paper is organized as follows: The basics of NBTI and HCI phenomena along with existing mitigation techniques are briefly explained in Section II. Section III presents our proposed approach followed by experimental results in Section IV. Finally, Section V concludes the paper.

II. Preliminaries

Both, BTI and HCI, result in a threshold voltage shift of the impaired transistors which in turn affect the gate and path delays. In this section, we briefly explain how these phenomena degrade the circuit delay over time and discuss different mitigation approaches.

A. Transistor Aging

1) Bias Temperature Instability: BTI is a phenomenon that increases transistor threshold voltage over long periods of time,

causing the drive current to decrease. This leads to slowing down the logic gates, eventually causing the circuit to violate its timing specifications. BTI affects PMOS transistors in form of Negative BTI (NBTI) and NMOS transistors in form of Positive (PBTI). While the first one is well known to be a major reliability challenge, PBTI emerged as an issue with the introduction of high-k gate materials [12]. When the gate node of a PMOS/NMOS transistor is negatively/positively biased (logic '0/'1 is applied), the transistor experiences NBTI/PBTI stress. As a result, the magnitude of threshold voltage (V_{th}) increases. When the negative/positive bias is removed (recovery phase), the previous V_{th}-shift is partially compensated. In long term, the magnitude of BTI-induced delay degradation depends on the ratio of stress to recovery time which can be derived from the gate input signal probability (sp) (for NBTI: 1-sp, for PBTI: sp). In addition, temperature has an exponential effect on BTI [3, 4]. To estimate the threshold voltage shift due to BTI we used the model from [13].

2) Hot Carrier Injection: HCI mainly affects the threshold voltage of NMOS transistors. During transitions, energetic (hot) electrons hit the interface between channel and gate oxide. As a results, free electron-hole pairs can be created and free electrons can be trapped in the gate oxide which eventually increases the threshold voltage. The degradation rate depends on the number of transitions, and hence on clock frequency, switching activity (sw), runtime, and also temperature [14]. In this work, we use the model adapted from [15] to estimate the threshold voltage shift due to HCI.

B. Related Work

Aging-aware synthesis: A synthesis approach using a BTI-aware cell library is introduced in [7]. To obtain the BTI-aware library all cells are characterized with respect to different input signal probabilities. However, this has two main shortcomings. First, signal probabilities should be calculated according to the post-synthesis simulations as these are workload-dependent and cannot be accurately calculated using analytical approaches during the synthesis flow, leading to an inaccurate aging estimation. Second, this technique cannot be integrated in the commercial synthesis flow.

Aging-aware re-timing: In order to balance the circuit timing with respect to a specific guardband, gate-sizing has been applied to gates with higher aging rates [8–11]. This will result in considerable lifetime improvement at the expense of some area overhead. In [6] a technique is presented that balances the instruction pipeline delays using gate-sizing and threshold voltage tuning at the expected lifetime. In this regards, stages with higher aging rates are designed faster and those with lower rates are designed slower. However, this technique is coarse-grained and hence cannot improve timing inside the pipeline stages/blocks. The technique presented in [16] employs time-borrowing flip-flops to balance the aged circuit timing based on the sensed delays. However, this technique might increase the clock skew as it manipulates the circuit clock tree.

Besides the existing aging-aware synthesis and re-timing categories, there are three major categories of aging mitigation techniques which are orthogonal to our proposed approach.

Signal probability balancing: One approach for BTI mitigation is to even out the wearout of pull-up and pull-down networks by balancing the signal probabilities. The techniques presented in [17, 18] employ cell-flipping in order to make the signal probability of SRAM cells close to 50%. Another technique presented in [19] periodically flips the leading bits of narrow-width values stored in an integer register-file in order to mitigate the overall degradation of the register-file. While bit-flipping techniques are very effective to mitigate the effect of BTI in memory elements, BTI mitigation of transistors in logic blocks needs more sophisticated approaches. The techniques presented in [4, 20, 21] apply pre-defined input vectors, selected according to signal probabilities of the block signals, on the input ports of blocks during idle cycles in order to mitigate aging. More specifically, as shown in [22], selection of an appropriate No-operation instruction reduces the degradation of execution units such as ALUs. Another technique proposed in [23] relies on alternating true- and complement-mode operations to equalize the utilization of devices in the data-path of processors.

Sense and adapt: An alternative aging mitigation solution is to extract the circuit delay during runtime using timing sensors [16, 24–27] or history sensors that track usage patterns [28] and scale the voltage or frequency accordingly. In order to avoid possible impact of timing sensors on the circuit timing, sensing can be based on replica paths which are demonstrative for the worst aging scenario in the circuit [5].

Power/temperature reduction techniques: Since temperature and supply voltage highly affect both BTI and HCI, a power reduction technique which reduces the overall temperature can be used to improve the lifetime. Particularly, it is shown in the literature that power gating [29, 30], dynamic voltage and frequency scaling [31, 32], and adaptive body biasing [33] can alleviate aging.

III. Proposed Approach

A. Motivation

Since aging-induced delay increase of a gate is highly dependent on the gate type, input signal profile, transition time, and temperature, a non-critical path might become critical and other way around over the operational lifetime [3, 6]. As a result, although the circuit timing is balanced at design time, due to different aging rates of transistors, it becomes significantly unbalanced after a certain period of time. Moreover, in the conventional non-aging-aware synthesis approach, the paths with large timing slacks at design time are designed slower in order to save area. However, these paths which are intentionally designed slower may have high aging rates and become critical over time. Considering these issues, an efficient approach to improve the lifetime is to balance the paths with respect to their post-aging delays based on a desired guardband instead of balancing the delays at design-time as it is done nowadays.

To illustrate this circumstance let us use the b19 benchmark circuit as a motivational example. In Figure 1, the *data arrival times*, defined as the time from the clock input through the launching flip-flop to the latching flip-flop or primary output, for nine representative flip-flops are shown for a non-aging-aware version of b19 (a) and an aging-aware one (b). Although the minimum clock period at design time is balanced according to the data arrival time of FF 1, the paths leading to FF 4 have higher aging rates and hence the data arrival time of this FF is the first one reaching the assigned guarband. Since in a

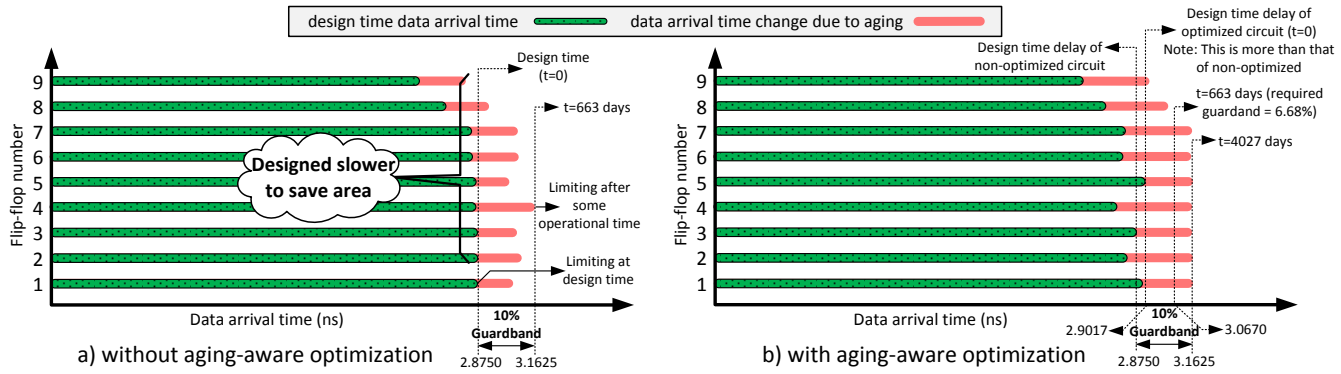

Fig. 1. Design time and post-aging data arrival times before and after aging-aware optimization for several representative flip-flops in b19 benchmark circuit

non-aging-aware synthesis the corresponding paths of FF 4 are intentionally designed slower to save area, as it can be seen in the figure, it has almost the same data arrival time as FF 1. As a consequence, these paths can be designed faster to either decrease the guardband for a specific lifetime or extend the circuit lifetime for a certain amount of guardband (see Figure 1.b).

Commercial synthesis tools employ a variety of techniques such as gate-sizing, register-retiming, and re-structuring to minimize the area while satisfying specified timing constraints [34, 35]. While previous techniques [9–11] have only focused on the gate-sizing technique, we will use all available techniques in a commercial synthesis tool to extend the lifetime.

In the proposed approach, our main objective is to assign aging-aware timing constraints to the circuit during the synthesis phase to improve the lifetime with respect to a particular amount of guardband. In this regard, in an iterative process, tighter timing constrains are assigned to paths which would exceed the desired guardband earlier. This will increase the overall lifetime at the expense of area. At the same time, in order to reduce the area overhead due to the first step, the timing constraints of paths that need longer time to reach the desired guardband are loosen. This optimization process is repeated until reaching a point that circuit lifetime cannot be further improved.

The data arrival times of the same b19 flip-flops after the aging-aware optimization are shown in Figure 1.b. As it can be seen, the corresponding paths of FF 4 and some others which have higher aging rates are designed faster to improve the lifetime. In contrast, those of FF 1 and FF 5 are slowed down to compensate the area penalty. Although the maximum data arrival time at design time for the optimized circuit is more than that of the original circuit, its post-aging characteristic is much better. In case of having 10% guardband, the optimized circuit works properly for 4027 days while the original circuit can only work for 663 days, meaning 6.07X lifetime improvement. If lifetime improvements are of a secondary interest, this approach can be used to reduce the amount of guardband. For instance, the optimized circuit requires just 6.68% (instead of 10% in the original circuit) for 663 days of operation which can be translated into more than 3% increase in the circuit maximum frequency and hence system performance.

Our investigation shows that considering a separated constraint for each path is infeasible for large circuits, and hence, as elaborated in Section III-B, we put constraints on the flip-flops and primary outputs. In order to assign constraints, detailed timing information of the circuit at design time and after some operation time are required. In this regard, we have adapted the static timing analysis for aged-delay computation. This is explained in Section III-C. According to the timing analysis results, some timing constraints are applied during each round of the optimization process to increase the circuit lifetime. The optimization details are given in Section III-D.

B. Optimal Targets for Timing Constraints

Previous techniques [6, 9, 10] extracted a list of critical paths and by analysing this list, they improved the lifetime of the circuit. Theoretically, in case of having a maximum aging rate of $\alpha\%$ for each transistor, all paths that have less than $\alpha\%$ slack time should be considered during the optimization. However, as the number of such paths could be intractably large, these previous techniques evaluated only a limited number of paths. For example, the b19 benchmark circuit has more than 100,000,000 paths. Accumulative distribution of these paths for different slack times is reported in Figure 2. As it can be seen, even consideration of paths with only 5% relative slack time becomes intractable in industrial-size circuits.

In our proposed approach, we employ a heuristic technique to overcome this problem. Since each path leads to either a flip-flop (FF) or a primary output (PO), by providing a timing constraints on a specific FF/PO, this constraint will be automatically applied to all corresponding paths. However, this constraint mainly affects the paths which do not meet it.

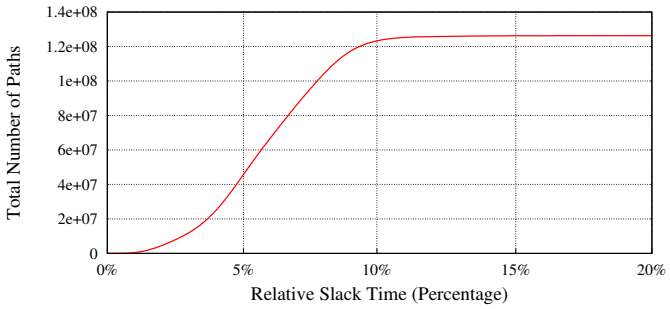

Fig. 2. Accumulative distribution of paths slack time for b19 benchmark circuit

It should be noted that due to shared gates (path segments) among different paths, it is quite possible that optimization of one path changes the timing of other paths. As in our example, b19 benchmark circuit has only 6042 FFs and 30 POs, meaning in our approach there are only 6042+30=6072 timing constraints (not 100M).

For aging-aware circuit optimization, these constraints provide almost the same result as offered by previous techniques obtained by putting constraints on paths, however, with a much shorter runtime. The only difference is that in some cases this approach might affect the timing of non-aging critical paths. As an example, consider two paths leading to the same FF and with almost the same design time delay but one is aging critical while the other one is not. Since we are putting some constraints related to all paths leading to the FF to reduce the design time delay of the aging-critical path, the other path also follows these constraints. This will result in some unnecessary area penalty. However, this area overhead is not considerable as in commercial synthesis tools parts of the circuit which are shared among several paths with unsatisfied timing constraints have higher optimization priority. In the above example, in case of existence of some shared sub-paths or gates between two paths, the synthesis tool puts more effort in these parts to avoid double overhead.

C. STA-based Post-aging Delay Computation

Static Timing Analysis (STA) is a well-known method for computing the expected timing of a digital circuit. In this section, it is demonstrated how this technique can be adapted to compute the post-aging delays for our purpose.

STA needs the timing information of all elements (i.e. gates, flip-flops, interconnects) to calculate the minimum clock period. Design time delays of all elements can be easily extracted by analysing the Standard Delay Format (SDF) file. The SDF file is provided by all commercial synthesis tools and includes timing constraints, path delays, and interconnect delays. By topological traversing a gate-level netlist extracted from a synthesis tool and taking into account maximum delay values from its corresponding SDF file, one can easily extract the maximum rise and fall data arrival time delays to all nodes including flip-flops inputs and POs. The minimum clock period is determined according the longest path in the circuit.

For the post-aging computation, we have employed a modified STA. The main difference is that an application profile (i.e. signal probabilities and switching activities) extracted from a post-synthesis simulation along with temperature obtained by a temperature model (e.g Hotspot tool [36]) are taken into account and based on the aging models (see Section II-A) the post-aging delay of each cell is estimated. During the post-synthesis simulation, some input vectors (e.g. a set of applications in case of a processor) based on the expected circuit functionality are simulated on the synthesized netlist and signal probabilities and switching activities for all circuit nodes are extracted. For temperature estimation, since temperature of a cell strongly depends on that of surrounding cells, the temperature model has to consider the circuit layout as well as the power profile (both leakage and dynamic) of the entire circuit. Using this information the V_{th}-Shift of each transistor is estimated and then, using an alpha-power model [37] the gate delay degradation is calculated (see Figure 3.a). Finally, during topological traverse post-aging delay of cells are used

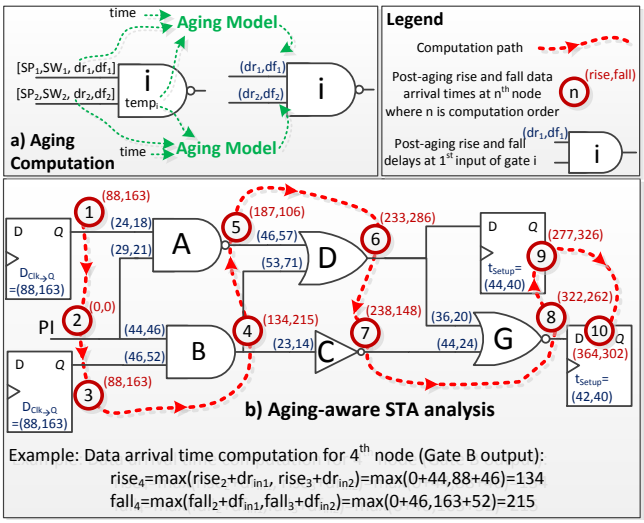

Fig. 3. An example of a) post-aging delay computation and b) aging-aware STA

to extract the data arrival time on all FFs/POs. A simple example of aging-aware STA analysis is shown in Figure 3.b. As it can be seen, it is very similar to the conventional STA analysis but here post-aging delays are taken into account during computations. It should be mentioned that the runtime of the aging-aware STA analysis is in order of $O(n)$ where n is the number of cells (i.e. gates and flip-flops) in the circuit. This is because each gate once and each flip-flop twice (once in the beginning for launch time and another at the end for capture time) are considered in the computation (see computation path in Figure 3.b)

In order to extract the circuit lifetime for a specific guardband, a binary search operation is performed over time (e.g 0-20 years). In each step, for a given time interval, the aging-aware STA analysis is done for the interval midpoint and the maximum post-aging delay of the circuit is computed and the time interval is shortened according to the relation between this delay and desired guardband. This continues until the time interval becomes small enough i.e the difference between higher and lower bound is smaller than a predefined number (e.g. few days), it is reported as the estimated lifetime for the given guardband which is denoted as t_l.

In summary, in STA-based circuit lifetime estimation with respect to the desired guardband, a binary search is performed over time and in each step STA analysis is perform (as explained from order of $O(n)$) to extract the post-aging delay. The overall runtime of this is from order of $O(n \cdot log(l/\theta))$ where n is number of cells, l in initial interval length in the binary search and θ is the predefined threshold. In case that the time interval is less than θ the search operation is terminated, so it is similar to case that a binary search operation is perform among l/θ which is from order of $O(log(l/\theta))$. Since both l and θ are constants, $O(n \cdot log(l/\theta)) \approx O(n)$.

D. Timing Constraints Optimization

Conventionally, during synthesis each path follows the timing constraint of the clock connected to its leading flip-flop. However, it is possible to assign other constraints (tighter or looser) to desired flip-flops. This will be automatically applied

to all corresponding paths. We employ this capability to modify the timing constraints of selective FFs/POs in order to improve the lifetime of the circuit with respect to a specific guardband.

Due to the shared gates among different paths, the minimum required data arrival time on one specific FF/PO is a function of timing of other FFs/POs. As a result, the optimization of constraints has to be performed iteratively with very small timing adjustments (e.g. 0.5% of overall delay) between two consecutive iterations, to monitor the effect not only on the target FF/PO but also on all others. Thus, an iterative optimization solution is employed to improve the circuit lifetime. The optimization process is explained in details in the rest of this subsection

In case that the designer adds a guardband g (as a percentage of total delay) to the current clock period T_{clk}, the minimum clock period after guardband becomes $T'_{clk} = T_{clk}(1+g)$. Let us assume that the maximum data arrival time for the i^{th} FF/PO is d_i. By computation of the post-aging delay according to the STA-based technique explained in Section III-C, the expected lifetime, i.e. t_l, is estimated. Also, using this approach, the degraded data arrival time for the i^{th} FF/PO denoted as d'_i is computed. According to these definitions, the maximum post-aging data arrival time at time t_l should be T'_{clk}.

During aging-aware synthesis, in an iterative process, the timing constraints of FFs/POs are optimized to increase circuit lifetime with little area overhead. The current timing constraint on the i^{th} FF/PO is denoted as cc_i, and in the beginning it is initialized to the corresponding clock period of which is T_{clk}. The optimized timing constraint (denoted as oc_i) can be tighter, looser, or similar to the previous iteration.

Tighter constraint: In case that the post-aging delay of a FF/PO is very close to the assigned guarband, this FF/PO determines the final lifetime. Thus, a tighter constraint is assigned to this. In other words, if $d'_i > T'_{clk}(1-\epsilon)$, where ϵ is pre-defined small number, $oc_i \leftarrow cc_i(1-\epsilon)$ (see FF A in Figure 4). This will force the synthesis tool to reduce d_i of this FF/PO by designing its corresponding paths faster with the intention that the post-aging delay of these paths (d'_i) is reduced and as a result the circuit lifetime is improved.

In case of success, the optimized circuit has less d_i than the previous one, but this does not always guarantee more lifetime for two reasons. First, circuit optimization will also affect the aging rates, due to gate sizing and path re-structuring, and in some cases, optimization may result in higher aging rates. Second, in some optimization iterations, the circuit optimization may change its structure in a way that a FF/PO with enough timing slack ($d_i << oc_i$) has a higher data arrival time than in the previous step and this specific FF/PO may become the lifetime-limiting one. However, this negative impact on lifetime does not indicate that the circuit cannot be improved further. In most cases, in the next optimization iterations and by putting some new constraints on these paths this negative effect will be solved.

Looser constraint: In the previous case, it is explained how constraints on some FFs/POs can reduce the corresponding post-aging data arrival times to be very close to $T'_{clk}(1-\epsilon)$. In the opposite case, if there is some slack with respect to $T'_{clk}(1-\epsilon)$ on a FF/PO at t_l, by putting looser constraint on this FF/PO, the synthesis tool can use the additional slack time

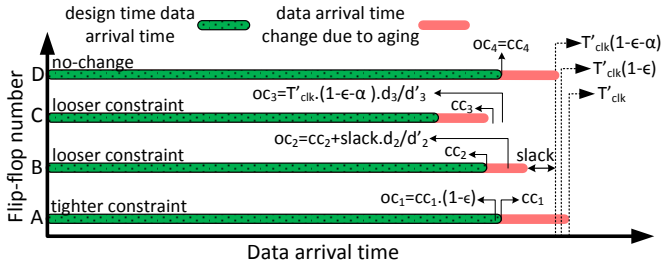

Fig. 4. Conceptual examples of different constraint optimization scenarios

to reduce the area. However, not the entire available slack time can be used for constraint optimization. Instead, the aging rate of the FF/PO data arrival time has to be considered as well. In other words, assuming the same aging rate for the FF/PO data arrival time in the optimized circuit, only d_i/d'_i fraction of the available slack can be used for constraint optimization and the rest is assumed to be used due to the extension of the post-aging delay change. Additionally, as the aging rate of the FF/PO data arrival time might change, using the entire slack time with respect to d_i/d'_i can lead to a timing violation. In this regard, $T'_{clk}(1-\epsilon-\alpha)$ is used in looser constraints instead of $T'_{clk}(1-\epsilon)$, where α is a very small constant.

Considering all these aspects together, two cases may occur when there is enough slack time ($d'_i << T'_{clk}$). First, in case that the current implementation has effectively used its slack time , i.e. $d_i \approx cc_i < d'_i$, the optimized constraint is $cc_i + slack \times d_i/d'_i$, where $slack = T'_{clk}(1-\epsilon-\alpha) - d'_i$ (see FF B in Figure 4). Second, in case that $cc_i > d'_i$, the current implementation does not use the available slack as it cannot be designed any slower to save area. In this case, the optimized constraint is set to the maximum possible value which is $T'_{clk}(1-\epsilon-\alpha)d_i/d'_i$ (see FF C in Figure 4). These two cases can be summarized to if $d'_i < T'_{clk}(1-\epsilon-\alpha)$, $oc_i \leftarrow min[cc_i + slack \times d_i/d'_i,\ \ T'_{clk}(1-\epsilon-\alpha) \times d_i/d'_i]$.

No-change constraint: When none of mentioned conditions are true, the previous constraint is preserved, i.e. $oc_i \leftarrow cc_i$.

Using these three conditions, new constraints are extracted at each iteration and then optimization is done by a synthesis tool to satisfy these constraints with little impact on area. During the optimization iterations, at some point, the synthesis tool cannot satisfy at least one of the assigned constraints. This mostly occurs on FFs/POs with tighter constraints as these cannot be designed any faster. But in some cases, it may also happen for paths with looser or no-change constraints as a result of optimization of tighter constraints on other FFs/POs. At this point, we search among different iterations and choose the circuit with maximum lifetime as target implementation. Please note that this is always in one of the last few iterations. In case that the lifetime of several implementations are very close (e.g. difference is less than 30 days) but there are significant variations on the area, one with the minimum area is preferred.

E. Overall Flow

The proposed aging-aware optimization flow is summarized in Algorithm 1. First, the circuit is synthesized using a commercial synthesis tool and its gate-level netlist, SDF file,

and maximum design time delay is extracted. Also, as we interrupt the synthesis operation to perform aging computations and extracting new constraints, a checkpoint of the current status of the synthesis process is stored (e.g in ddc format in Design Compiler) to be used during the next optimization iteration.

In order to extract the temperature profile of the circuit, a power profile of the circuit is required. The power profile is obtained by running some input vectors during the post-synthesis simulation (e.g. using Modelsim). During the simulation, the application profile is stored in the SAIF format. This is given to a power analysis tool to estimate leakage and dynamic power consumption of each cell. The power profile is forwarded into the temperature estimation tool to compute the temperature of each cell.

The constraints are set to T'_{clk} and are optimized according to the explanation provided in Section III-D (line 13-21). In case that at least one of the constraints cannot met, the optimization is terminated and by searching among all iterations the best implementation from lifetime point of view is selected.

As it is demonstrated in the experimental results, this optimization does not affect most of the cells, hence power profile will not change significantly and thus temperature will not change much. In order to avoid time-consuming temperature estimation, we only extract temperature once in advance for all cells. Since optimization rarely affects the flip-flops, we assume flip-flops have constant temperatures, and for new cells created due to the optimization, their temperature is set to the that of the closet flip-flop in the netlist. In case of change in flip-flops due to the time-borrowing technique, new flip-flops take the temperature of the original one.

Algorithm 1: Constraints Optimization

```
   input  : HDL: design Verilog/VHDL description
   input  : lib: synthesis library
   input  : app: application
   input  : g: desired guardband
   input  : T_clk: minimum clock period
 1 DDC, SDF, netlist ← synthesize (HDL, lib, T_clk)
 2 SAIF ← post-synthesis-simulation (netlist, SDF, app)
 3 pow ← power-estimation-tool (netlist, SAIF)
 4 layout ← place-and-route-tool (netlist, lib)
 5 temp ← temperature-estimation-tool(netlist, pow,layout)
 6 T'_clk ← T_clk(1 + g)
 7 oc_i ← T_clk
 8 repeat
 9     cc_i ← oc_i
10     sp, sw ← post-synthesis-simulation (netlist, sdf, app)
11     d_i, d'_i, t_l ← STA (netlist, SDF, sw, sp, temp, T'_clk)
12     foreach i in FF/PO do
13         if d'_i > T'_clk(1 − ϵ) then
14             oc_i ← cc_i(1 − ϵ)
15         end
16         else if d'_i < T'_clk(1 − ϵ − α) then
17             oc_i ← min[cc_i + (T'_clk(1 − ϵ − α) − d'_i)d_i/d'_i
               , T'_clk(1 − ϵ − α) × d_i/d'_i]
18         end
19         else
20             oc_i ← cc_i
21         end
22     end
23     DDC, SDF, netlist ← incremental-synthesis (DDC, lib, oc_i)
24     temp ← adjust-temperature (temp)
25 until constraints met
26 Search among all iterations and find the one with maximum lifetime
27 Report constraints of that step as final answer
```

F. Other Timing Constraints

The proposed approach has no limitation about different timing scenarios in the design. It should be noted that, in the proposed approach, the synthesis tool automatically takes care of setup/hold time and input/output delay constraints. Additionally, this approach can be applied to designs with more complicated timing scenarios such as multiple clock domains without additional effort.

IV. Experimental Results

We have conducted several experiments on ITC99 benchmark circuits to show the effectiveness of the proposed approach and compare it with pure guardbanding and gate-sizing techniques.

A. Experimental Setup

All parts of the proposed flow excluding synthesis, optimization, power consumption estimation, and temperature computation are integrated in an in-house tool developed with C++. Synopsys Design Compiler is employed for synthesis and optimization operations. However, the proposed approach can be applied to all other commercial synthesis tools as well. The circuit is first synthesized with the minimum possible clock period using the Nangate 45 nm standard cell library. The minimum clock period can be determined by setting the clock period to zero during synthesis and extracting the maximum negative slack. Then, during optimization iterations, new constraints are assigned to the corresponding signals of each FF/PO using the *set_max_delay* command provided by Design Compiler.

In order to extract application profiles, we have used random input vector simulations with an average signal probability of 0.5 as there is no specific application for ITC99 benchmark circuits. For each benchmark circuit, a testbench containing 10^6 input vectors is used in each optimization round to calculate signal probabilities and switching activities. This will assure us that we have exactly the same profile for unchanged parts of the circuit.

In order to extract the power consumption using Design Compiler, all changes of the internal signals when applying input vectors are necessary. Therefore, SAIF files are extracted using Modelsim. It should be mentioned that as extracting SAIF files and analysing them doubles the effort (most time consuming part is access to file for both writing and reading), we just used them for extracting power as it is needed once in advance. For computing signal probabilities and switching activities during optimization iterations, we have used the logic simulator embedded in our in-house tool. The signal probabilities and switching activities comparison for all ITC99 benchmark circuits confirmed that our logic simulator extracts the exactly same results in almost 30% of the time needed for extracting and analysing SAIF files.

The temperature of all elements in the circuit are extracted by the means of Hotspot. This tool gets the power profile extracted from Design Compiler and also the physical layout of the circuit and computes the temperature with respect to physical adjacencies, switching activities, etc. The circuit layout is extracted by Cadence SOC Encounter.

B. Lifetime Improvement

The lifetime of ITC99 benchmark circuits for both gate-sizing and the proposed approach in case of 10% guardband are presented in Table I. As it can be seen, in average

TABLE I. EVALUATION OF GUARDBANDING, GATE-SIZING, AND PROPOSED APPROACHES FOR 10% GUARDBAND, AND OPTIMIZATION PARAMETERS OF $\epsilon = 0.5\%$ AND $\alpha = 0.2\%$

Circuit	Elements			Guardbanding			Gate-sizing [8–11]		Proposed			
	FFs	Gates	T_{clk} [ns]	Lifetime $[days]$	Area $[\mu m^2]$	Run-time$[s]$	Lifetime improv.	Area improv.	Lifetime (improv.)$[days]$	Area improv.$[\mu m^2]$	Opt. rounds	Runtime (overhead)$[s]$
b01	5	56	0.344	980	93.6	29.7	1.61X	−1.13%	4,404 (4.49X)	94.3 (−0.74%)	16	72.9 (2.45X)
b02	4	29	0.315	941	57.6	18.3	1.79X	−0.98%	2,318 (2.46X)	58.2 (−0.99%)	8	31.0 (1.69X)
b03	30	152	0.389	951	372.7	33.9	1.35X	−0.41%	1,992 (2.09X)	374.5 (−0.49%)	7	44.3 (1.31X)
b04	66	545	0.480	664	1,036.2	74.2	2.32X	−2.80%	2,684 (4.04X)	1,020.3 (+1.53%)	20	135.8 (1.83X)
b05	34	730	0.701	1,091	1,115.7	112.1	1.07X	−0.12%	1,574 (1.44X)	1,080.1 (+3.19%)	3	123.2 (1.10X)
b06	9	71	0.357	881	144.6	19.1	1.42X	−0.66%	1,332 (1.51X)	145.8 (−0.82%)	4	33.2 (1.74X)
b07	44	625	0.445	702	1,030.2	38.4	1.27X	−0.89%	1,536 (2.19X)	1,008.2 (+2.14%)	3	48.6 (1.27X)
b08	21	183	0.460	602	354.3	26.5	1.18X	−0.13%	1,348 (2.24X)	347.4 (+1.95%)	4	38.4 (1.45X)
b09	28	203	0.432	974	413.2	23.5	1.00X	+0.00%	974 (1.00X)	393.4 (+4.80%)	2	29.4 (1.25X)
b10	17	220	0.455	663	372.9	24.3	1.28X	−0.79%	859 (1.30X)	363.9 (+2.40%)	3	29.9 (1.23X)
b11	30	830	0.519	721	1,221.9	91.8	1.86X	−2.34%	2931(4.07X)	1,267.4 (−3.72%)	6	120.8 (1.32X)
b12	121	1,133	0.573	882	2,114.7	81.8	2.39X	−3.13%	3,526 (4.00X)	2,109.8 (+0.23%)	23	183.3 (2.26X)
b13	48	324	0.382	1,124	677.0	29.2	1.00X	+0.00%	1,401 (1.25X)	656.2 (+3.07%)	3	31.2 (1.07X)
b14	215	9,289	1.299	606	12,991.4	1,796.2	1.93X	−2.28%	1,571 (2.59X)	13,130.2 (−1.07%)	27	2189.6 (1.22X)
b15	417	7,995	1.010	884	12,501.4	967.5	2.11X	−2.04%	5,638 (6.38X)	12,644.4 (−1.14%)	76	2443.4 (2.50X)
b16	55	2,931	0.442	911	4,280.9	951.3	1.62X	−0.84%	1,917 (2.10X)	4,235.5 (+1.05%)	4	1061.9 (1.11X)
b17	1,317	24,347	1.035	826	38,434.9	777.8	3.02X	−3.26%	6,134 (7.42X)	39,005.5 (−1.48%)	125	1952.0 (2.51X)
b18	3,020	76,406	3.724	1,783	110,860.9	2,523.6	1.44X	−1.63%	7,177 (4.03X)	111,254.8 (−0.35%)	16	3756.0 (1.49X)
b19	6,042	137,897	2.875	663	205,424.6	3,427.9	2,74X	−0.89%	4,027 (6.07X)	206,406.4 (−0.47%)	24	9022.0 (2.63X)
b20	430	18,488	1.388	623	25,678.0	1,775.1	2.90X	−3.31%	4,046 (6.49X)	26,479.2 (−3.12%)	86	5909.5 (3.33X)
Average							**1.77X**	**−1.38%**	**3.36X**	**+0.30%**	**23.00**	**1.74X**

the gate-sizing technique has improved the circuit lifetime by 1.77X at the expense of 1.38% area overhead, although some of this area overhead can be compensated according the technique presented [6]. In contrast, our proposed approach has improved the lifetime by 3.36X while it reduces the circuit area by 0.30%. In fact, most of the area overhead imposed by putting tighter constraints is compensated by looser ones. An important point is that the lifetime provided by the gate-sizing technique is always considerably shorter than that of the proposed approach. Indeed, as mentioned earlier, gate-sizing is just one of the techniques that commercial synthesis tools consider during optimization. Since our proposed technique exploits all these features, the lifetime results are always superior to the pure gate-sizing ones.

The lifetime improvement of both gate-sizing and the proposed approach mostly depends on how much one can tighten the timing constraints of the paths with higher aging rates. As a consequence, it can happen, that the lifetime cannot be improved, if the design-time critical path (cannot be designed faster) has also the highest aging rate. However, in such a case, area can be significantly reduced by slowing down all other paths. For example, the lifetime of b09 benchmark cannot be improved as the design time most-critical path has the highest aging rate. However, as shown in the table, the proposed approach can reduce its area by 4.80%. In contrast, for most of the benchmarks circuits (e.g. b15) a considerable lifetime improvement is gained at the expense of a negligible area overhead.

Figure 5 shows how lifetime is improved over optimization iterations and how area is affected. As it can be seen, although in some points lifetime decreases, its overall trend is increasing. In the first few iterations, there is a considerable reduction in the area as we can put lots of looser constraints, however, in the later iterations, the lifetime is improved due to tighter constraints at the expense of area penalty.

C. Runtime

In order to evaluate the scalability of the proposed approach to optimize the circuit timing in aging-aware manner, the runtimes of this approach and a pure guardbanding approach

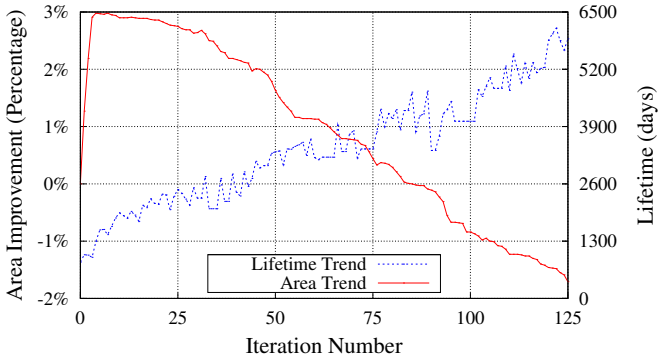

Fig. 5. a) Lifetime and b) area improvement for different optimization iterations of b17 benchmark

are reported in Table I. All experiments are conducted on a workstation with Intel Xeon E5540 2.53 GHz and 16 GB RAM.

For both guardbanding and the proposed approach, in the beginning, we need to do a complete synthesis operation to map the HDL description according to the synthesis library. Extraction of SAIF files, power profile and temperature is also done for both techniques once in advance. In guardbanding, by running the STA analysis the circuit lifetime is extracted. Our analysis shows that on average only 2.8% of the reported runtime is due to the STA analysis and the remaining 97.2% is used for initial computations. In the proposed approach, after the initial computations, in each iteration application profiles are extracted using our in-house tool, a STA analysis is performed, new constraints are assigned, circuit optimizations are done and at the end temperature is assigned accordingly. Among these, the circuit optimization is the most time consuming one. It should be noted that the runtime of the proposed approach is highly dependant on the number of iterations. However, for a specific circuit, the time needed for some iterations might be several orders of magnitude lower depending on how much effort Design Compiler should put to satisfy the desired constraints.

As reported in Table I, the proposed approach has 1.74X

more runtime than the pure guardbanding. This means that the complete synthesis and temperature extraction are still the dominant parts in the aging computation. This is indicative of the scalability of the proposed approach.

V. Conclusions

In this paper, an aging-aware synthesis approach is presented to address the design-challenges imposed by accelerated transistor aging due to BTI and HCI. It iteratively optimizes circuit timing with respect to post-aging delays in a way that all paths reach the assigned guardband at the same time. Experimental results on ITC99 benchmark circuits show that the proposed approach can improved the lifetime by 3.36X which is two times better than the state of the art gate-sizing techniques. This approach has negligible impact on area and its runtime is comparable with that of a pure-gurdbanding technique, and hence, scales very well.

Acknowledgment

This work was partly supported by the German Research Foundation (DFG) as part of the national focal program "Dependable Embedded Systems" (SPP-1500, http://spp1500.ira.uka.de).

References

[1] J. Fang, S. Gupta, S.V. Kumar, S.K. Marella, V. Mishra, P. Zhou, and S.S. Sapatnekar. Circuit reliability: from physics to architectures. In *International Conference on Computer-Aided Design*, pages 243–246, 2012.

[2] J. Henkel, L. Bauer, N. Dutt, P. Gupta, S. Nassif, M. Shafique, M. Tahoori, and N. Wehn. Reliable on-chip systems in the nano-era: lessons learnt and future trends. In *Design Automation Conference*, pages 1–10, 2013.

[3] W. Wang, S. Yang, S. Bhardwaj, R. Vattikonda, S. Vrudhula, F. Liu, and Y. Cao. The impact of NBTI on the performance of combinational and sequential circuits. In *Design Automation Conference*, pages 364–369, 2007.

[4] Y. Wang, H. Luo, K. He, R. Luo, H. Yang, and Y. Xie. Temperature-aware NBTI modeling and the impact of input vector control on performance degradation. In *Design, Automation and Test in Europe Conference*, pages 1–6, 2007.

[5] S. Wang, J. Chen, and M. Tehranipoor. Representative critical reliability paths for low-cost and accurate on-chip aging evaluation. In *International Conference on Computer-Aided Design*, pages 736–741, 2012.

[6] F. Oboril and M.B. Tahoori. MTTF-Balanced pipeline design. In *Design, Automation and Test in Europe Conference*, pages –, 2013.

[7] S.V. Kumar, C.H. Kim, and S.S. Sapatnekar. NBTI-aware synthesis of digital circuits. In *Design Automation Conference*, pages 370–375, 2007.

[8] B.C. Paul, K. Kang, H. Kufluoglu, M. A Alam, and K. Roy. Negative bias temperature instability: Estimation and design for improved reliability of nanoscale circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 26(4):743–751, 2007.

[9] W. Wang, Z. Wei, S. Yang, and Y. Cao. An efficient method to identify critical gates under circuit aging. In *International Conference on Computer-Aided Design*, pages 735–740, 2007.

[10] J. Chen, S. Wang, and M. Tehranipoor. Efficient selection and analysis of critical-reliability paths and gates. In *Great Lakes Symposium on VLSI*, pages 45–50, 2012.

[11] X. Yang and K. Saluja. Combating NBTI degradation via gate sizing. In *International Symposium on Quality Electronic Design*, pages 47–52, 2007.

[12] W. Yang, H.and Hwang and C. Chuang. Impacts of nbti/pbti and contact resistance on power-gated sram with high-k metal-gate devices. *IEEE Transactions on Very Large Scale Integration Systems*, 19(7):1192–1204, 2011.

[13] Y. Wang, X. Chen, W. Wang, V. Balakrishnan, Y. Cao, Y. Xie, and H. Yang. On the efficacy of input vector control to mitigate nbti effects and leakage power. In *International Symposium on Quality of Electronic Design*, pages 19–26. IEEE, 2009.

[14] A. Bravaix, C. Guerin, V. Huard, D. Roy, J.M. Roux, and E. Vincent. Hot-carrier acceleration factors for low power management in dc-ac stressed 40nm nmos node at high temperature. In *International Reliability Physics Symposium*, pages 531–548, 2009.

[15] F. Oboril and M.B. Tahoori. Extratime: Modeling and analysis of wearout due to transistor aging at microarchitecture-level. In *Dependable Systems and Networks*, pages 1–12, 2012.

[16] H. Dadgour and K. Banerjee. Aging-resilient design of pipelined architectures using novel detection and correction circuits. In *Design, Automation and Test in Europe*, pages 244–249, 2010.

[17] Y. Kunitake, T. Sato, and H. Yasuura. Signal probability control for relieving NBTI in SRAM cells. In *International Symposium on Quality Electronic Design*, pages 660–666, 2010.

[18] S.V. Kumar, K.H. Kim, and S.S. Sapatnekar. Impact of NBTI on SRAM read stability and design for reliability. In *International Symposium on Quality Electronic Design*, pages 1–6, 2006.

[19] S. Wang, T. Jin, C. Zheng, and G. Duan. Low power aging-aware register file design by duty cycle balancing. In *Design, Automation and Test in Europe Conference*, pages 546–549, 2012.

[20] J. Abella, X. Vera, and A. Gonzalez. Penelope: The NBTI-aware processor. In *International Symposium on Microarchitecture*, pages 85–96, 2007.

[21] D.R. Bild, R.P. Dick, and G.E. Bok. static NBTI reduction using internal node control. *ACM Transactions on Design Automation of Electronic Systems*, 17(4):45, 2012.

[22] F. Firouzi, S. Kiamehr, and M.B. Tahoori. NBTI mitigation by optimized NOP assignment and insertion. In *Design, Automation and Test in Europe Conference*, pages 218–223, 2012.

[23] E. Gunadi, A.A. Sinkar, N.S. Kim, and M.H. Lipasti. Combating aging with the colt duty cycle equalizer. In *International Symposium on Microarchitecture*, pages 103–114, 2010.

[24] S. V. Kumar, C. H. Kim, and S. S Sapatnekar. Adaptive techniques for overcoming performance degradation due to aging in digital circuits. In *Asia and South Pacific Design Automation Conference*, pages 284–289, 2009.

[25] L. Lai, V. Chandra, R. Aitken, and P. Gupta. SlackProbe: A Low Overhead In Situ On-line Timing Slack Monitoring Methodology. In *Design, Automation and Test in Europe Conference*, 2013.

[26] A. Rahimi, L. Benini, and R.K. Gupta. Hierarchically focused guardbanding: an adaptive approach to mitigate PVT variations and aging. In *Design, Automation and Test in Europe Conference*, pages 1695–1700, 2013.

[27] A.C. Cabe, Z. Qi, S.N. Wooters, T.N. Blalock, and M.R. Stan. Small embeddable NBTI sensors (SENS) for tracking on-chip performance decay. In *International Symposium on Quality Electronic Design*, pages 1–6, 2009.

[28] E. Mintarno, J. Skaf, R. Zheng, J. Velamala, Y. Cao, S. Boyd, R.W. Dutton, and S. Mitra. Optimized self-tuning for circuit aging. In *Design, Automation and Test in Europe Conference*, pages 586–591, 2010.

[29] K.K. Kim, H. Nan, and K. Choi. Adaptive hci-aware power gating structure. In *International Symposium on Quality Electronic Design*, pages 219–224, 2010.

[30] A. Calimera, E. Macii, and M. Poncino. Nbti-aware power gating for concurrent leakage and aging optimization. In *international symposium on Low power electronics and design*, pages 127–132, 2009.

[31] L. Zhang and R.P. Dick. Scheduled voltage scaling for increasing lifetime in the presence of nbti. In *Asia and South Pacific Design Automation Conference*, pages 492–497, 2009.

[32] M. Basoglu, M. Orshansky, and M. Erez. Nbti-aware dvfs: a new approach to saving energy and increasing processor lifetime. In *International symposium on Low power electronics and design*, pages 253–258, 2010.

[33] Z. Qi and M.R. Stan. Nbti resilient circuits using adaptive body biasing. In *Great Lakes symposium on VLSI*, pages 285–290, 2008.

[34] *Synopsys Timing Constraints and Optimization User Guide*, 2010.

[35] N. Maheshwari and S. Sapatnekar. *Timing analysis and optimization of sequential circuits*. Springer, 1998.

[36] W. Huang, S. Ghosh, S. Velusamy, K. Sankaranarayanan, K. Skadron, and M.R. Stan. Hotspot: A compact thermal modeling methodology for early-stage vlsi design. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 14(5):501–513, 2006.

[37] K.A. Bowman, B.L. Austin, J.C. Eble, X. Tang, and J.D. Meindl. A physical alpha-power law MOSFET model. In *International Symposium on Low Power Electronics and Design*, pages 218–222, 1999.