Static NBTI Reduction Using Internal Node Control

DAVID R. BILD and ROBERT P. DICK, University of Michigan
GREGORY E. BOK, Nico Trading

Negative Bias Temperature Instability (NBTI) is a significant reliability concern for nanoscale CMOS circuits. Its effects on circuit timing can be especially pronounced for circuits with standby-mode equipped functional units, because these units can be subjected to static NBTI stress for extended periods of time. This article describes Internal Node Control (INC), in which the inputs to some individual gates are directly manipulated to prevent this static NBTI fatigue. We prove that the INC selection problem is $\mathcal{NP}$-complete and present a linear-time heuristic that can quickly determine near-optimal placements. This near-optimality is confirmed by comparing results for small benchmarks against optimal solutions from a mixed integer linear programming formulation of our problem. We evaluate the heuristic on the ISCAS85 benchmarks and the Synopsys DesignWare Library. Our heuristic reduces static NBTI-induced delay over a ten year period by 30–60% and can reduce total path delay by an average 9.4% when NBTI degradation is severe. The INC placements and sleep signal routing require only a 1.6% increase in area.

Categories and Subject Descriptors: B.8.2 [**Hardware**]: Performance and Reliability—*Performance analysis and design aids*; B.7.2 [**Hardware**]: Integrated Circuits—*Design aids*

General Terms: Reliability, Design, Algorithms

Additional Key Words and Phrases: Internal node control, input vector control, NBTI, negative bias temperature instability

ACM Reference Format:
Bild, D. R., Dick, R. P., and Bok, G. E. 2012. Static NBTI reduction using internal node control. ACM Trans. Des. Autom. Electron. Syst. 17, 4, Article 45 (October 2012), 30 pages.
DOI = 10.1145/2348839.2348849 http://doi.acm.org/10.1145/2348849.2348849

1. INTRODUCTION

Due to the scaling trends of CMOS technology, Negative Bias Temperature Instability (NBTI) is emerging as a significant reliability concern for digital circuits. NBTI, which in current technologies only significantly affects PMOS transistors stressed with a negative bias ($V_{gs} = -V_{dd}$), manifests itself as an increase in threshold voltage that reduces switching speed [Alam and Mahapatra 2005].

At the atomic level, NBTI is caused by an electric-field-dependent disassociation of Si-H bonds at the Si/SiO_2 interface. The hydrogen diffuses into the gate oxide in

A preliminary version of this work appears in *Proceedings of the Design, Automation, and Test in Europe Conference* [Bild et al. 2009].
This work was supported in part by the SRC under award 2007-HJ-1593 and in part by the NSF under awards CCF-0702761 and CNS-0347941.
Authors' addresses: D. R. Bild (corresponding author) and R. P. Dick, Electrical Engineering and Computer Science Department, University of Michigan, Ann Arbor, MI 48109; email: drbild@umich.edu; G. E. Bok, Nico Trading, Chicago, IL 60606.

DOI 10.1145/2348839.2348849 http://doi.acm.org/10.1145/2348839.2348849

a temperature-dependent reaction, leading to the formation of interface traps, which are responsible for an increase in threshold voltage. These mechanisms lead to an interesting recovery effect; when the stress is removed ($V_{gs} = 0$), the reaction reverses, with some of the hydrogen diffusing back towards the interface and bonding with the Si [Alam and Mahapatra 2005].

Under constant stress, static NBTI effects quickly lead to performance degradation. However, thanks to the previously described recovery effect, for circuits experiencing typical switching activity, the negative impacts of dynamic NBTI degradation take longer to accumulate. For a 70 nm Berkeley Predictive Technology Model, Paul et al. predict ∼10% increase in delay after 10 years of operation for the ISCAS85 benchmarks [Brglez and Fujiwara 1985; Paul et al. 2005].

During normal circuit operation, standard switching activity causes alternating stress on the PMOS transistors, and thus degradation is dominated by dynamic NBTI. However, many designs employ sleep or clock-gating techniques in order to reduce dynamic power consumption. In such schemes, idle functional units are put in standby or sleep mode by having their inputs frozen or their clock transitions gated. This prevents extraneous transitions, reducing dynamic power consumption. However, with the inputs stable for long periods of time, PMOS transistors with low inputs may degrade due to static NBTI effects. In this scenario, static NBTI optimization is relevant.

In this article, we describe and evaluate an internal node control technique to limit the effect of this static NBTI stress. Internal node controls can be inserted at the outputs of individual gates in order to force them to specific values during standby. Using this technique, static NBTI stress for a PMOS transistor can be eliminated, for example, by forcing the output of the preceding gate to V_{dd}. However, internal node control imposes a timing penalty; the additional circuitry required for node control introduces a small delay. NBTI degradation on a timing-sensitive (i.e., critical path) transistor can be eliminated by forcing noncritical path gates to circuit structure-dependent values, such that a low value is propagated to the NBTI-sensitive transistor.

We show that the problem of solving for the optimal set of insertion points leading to the minimal degradation in circuit delay after some elapsed time is $\mathcal{NP}$-complete and formulate it as a mixed integer linear program. We present a linear-time heuristic that can quickly determine near-optimal placements. This near-optimality is confirmed by comparing results for small benchmarks against optimal solutions from a mixed integer linear programming formulation of our problem. We evaluate the heuristic on benchmarks from the ISCAS85 set and the Synopsys DesignWare Library. Our heuristic reduces static NBTI-induced delay over a ten year period by 30–60% and can reduce total path delay by an average 9.4% when NBTI degradation is severe. The INC placement and sleep signal routing require only a 1.6% increase in area. PBTI, a similar mechanism affecting NMOS transistors, is becoming a significant problem as well, so we further show that our INC heuristic extends to the simultaneous reduction of NBTI and PBTI stresses, with similar results.

This article is organized as follows. In Section 2, we describe models for the NBTI degradation process and discuss related techniques for the mitigation and prevention of static NBTI effects. In Section 3, we present our model of internal node control for static NBTI control and prove that the problem is $\mathcal{NP}$-complete. In Section 4 we describe the mixed integer linear programming formulation for its optimal solution. In Section 5, we present the linear-time heuristic and give the experimental results for both the ISCAS85 and DesignWare benchmarks. In Section 7 we evaluate a multiple-IVC vector extension to INC. Finally, in Section 8, we discuss the sensitivity of the benefits of INC to both primary output slack distribution and circuit length.

2. BACKGROUND

This section first describes models for the NBTI physical degradation mechanism and explains the model used in this work. It then describes related work on techniques that compensate for or mitigate the degradation.

2.1. Overview of NBTI Models

Negative Bias Temperature Instability (NBTI) is a degradation mechanism affecting PMOS transistors, which results in increased threshold voltage and thus slower switching speeds. The increase in threshold voltage is generally thought to be caused by the generation of interface traps and oxide charges in PMOS transistors under negative bias ($V_{gs} = -V_{dd}$). These interface traps, dangling bonds, and oxide charges are attributed to an electric-field-dependent disassociation of Si–H bonds at the $Si–SiO_2$ interface. In the reaction-diffusion model, the currently accepted model for this mechanism, the interface trap generation (reaction) results in free hydrogen that diffuses into the oxide (diffusion) [Schroder 2007]. Both the reaction and diffusion regimes limit the rate of degradation. The specific diffusion species (H, H_2, or a mixture of the two) is not currently known, meaning that there is still debate about the accuracy of the various diffusion models. When the stress is removed, the hydrogen diffuses back towards the interface and can rebond with the Si. This reversal of the mechanism is known as the recovery effect. The recovery effect complicates attempts to measure NBTI degradation because some of the degradation recovers between the time that the stress is removed and the time that measurements are taken. Several recent review papers provide detailed explanations and discussions of these models [Alam and Mahapatra 2005; Huard et al. 2006; Stathis and Zafar 2006; Schroder 2007; Schroder and Babcock 2003].

The NBTI models mentioned in the preceding papers are useful for understanding the physical mechanisms that lead to the degradation, but are too detailed, and thus too slow, for use in circuit-level analysis. Cycle-based, transistor-level simulation is too time-consuming for use in reliability analysis tools or reliability-aware CAD algorithms. Consequently, several researchers have developed analytical models for predicting NBTI degradation [Bhardwaj et al. 2006; Kang et al. 2007; Kumar et al. 2006; Paul et al. 2005; Saluja et al. 2008; Vattikonda et al. 2006; Wang et al. 2007b]. As mentioned in the previous paragraph, when the stress on a transistor is removed, the reaction reverses and thus some of the degradation is also reversed. This recovery effect is so pronounced for gates experiencing rapid switching that most analytical models differentiate between static and dynamic stress. Static stress, as might occur in an idle functional unit and with which we are concerned in this work, occurs when the transistor is stressed continuously for a long period of time. Dynamic stress occurs when the stress is repeatedly and alternately applied, as would be common in an operating functional unit.

Much of the analytical literature focuses on dynamic stress. Although models for static stress exist, due to the difficulties in obtaining experimental data for extended-period static stress, they are not accurate for long time periods. These models employ empirical constants that are found by fitting against experimental data. This data is only available for short timescales (e.g., days) and consequently the models fail to fully capture the limiting processes that slow the degradation rate over time, resulting in overestimation of the stress when extrapolated to periods of five to ten years. The models can be used to determine the evolution of NBTI over time (i.e., the shape of the NBTI degradation versus time relationship) by fixing the amount of degradation after some fixed period (e.g., 10% increase in V_{th} after ten years) and solving for fitting parameters that match these two boundary conditions (the other being the trivial

0% increase in V_{th} after zero years). In this work, however, we are concerned with the aggregate static degradation (i.e., total change in delay after the rated lifetime of the part) and thus employ the following model based on degradations reported in the literature.

2.2. Aggregate Static NBTI Model

Our *static NBTI* model assumes a fixed percentage increase in delay for each gate stressed in idle mode. We note three properties of our particular problem that make this simplification appropriate, but first point out its primary advantage. Our results are parametrized in this one metric, so readers with differing (possibly proprietary) NBTI models can easily estimate the impact of INC for their processes without actually running our proposed algorithm with their models. This aggregate model is appropriate, instead of one that explicitly models time, because our problem formulation has the following properties.

(1) Our technique targets and reduces static NBTI stress only, so we need not model dynamic stress. The performance of INC is independent of any dynamic stress.
(2) Our technique targets and reduces static NBTI stress in idle mode only. In our problem formulation, signal inputs in idle mode are fixed at design time to control implementation overhead. Thus, our static NBTI model, unlike one for dynamic stress, does not need to incorporate signal probabilities; all nodes have fixed idle-mode values at design time.
(3) The evolution of static NBTI stress over time is unimportant; thanks to its nondecreasing nature, the aggregate V_{th}degradation at the rated lifetime of the part is the relevant metric.

These three properties together imply that we need not explicitly model the static stress as a function of active-to-standby ratio, input signal probabilities, or time, but can instead assume a fixed degradation per stressed gate at the end of our 10-year target lifetime.

10-year degradation rates reported in the literature vary widely, in some cases due to the differing NBTI-sensitivities of various processes (e.g., high-k dielectrics or nitridization amounts affect NBTI degradation rates [Li et al. 2004]) and in others due to the difficulty of estimating the degradation rates for long time durations (e.g., the correct time exponent in the R-D model is still debated [Schroder 2007]). Thus, we evaluate our technique at two different degradation percentages, spanning the reported values. The first value, the lower bound, is a 10% increase in delay for each stressed gate [Abella et al. 2007; Paul et al. 2005], which translates to an average 3.3% increase in critical path delay for our benchmarks[1]. The second value, the upper bound, is a 50% increase in delay for each stressed gate based on the average degradation reported over a set of representative benchmarks [Wang et al. 2010], which translates to an average 16.3% increase in critical path delay for our benchmarks.

The performance of our technique, measured as percent reduction in *NBTI-induced* delay, improves as the NBTI degradation percentage increases. Thus, the results for the 10% stressed-gate degradation assumption are conservative; larger degradation percentages result in greater degradation reduction. However, the impact of NBTI on total path delay is small (3.3% increase) so even full mitigation has limited practical value. Under the 50% stressed-gate degradation assumption, total path delays are significantly impacted (16.3% increase), so those results illustrate the practical value

[1]Not all gates on the critical path are stressed; only the stressed subset incur the 10% increase in delay resulting in a substantially smaller increase in total path delay.

of the INC technique for process technologies and temperatures for which NBTI is a significant problem. Section 6 presents additional explanation and full experimental results for both degradation values.

2.3. Related Work

Several techniques have been proposed for dealing with the impacts of NBTI. These methods generally fall into two groups: those that simply compensate for the NBTI-induced timing degradation and those that actively attempt to decrease and mitigate the degradation. In the following two sections, we summarize and discuss existing work of both types.

2.3.1. Compensating Techniques. Methods of this class, which includes guard banding, gate sizing, V_{dd} tuning, and V_{th} tuning, have been used in industry to compensate for timing degradation. Such techniques compensate for the effects of NBTI at the expense of timing, area, or power because they do not reduce the NBTI-induced degradation.

In guard banding, the maximum clock frequency of a circuit is artificially limited, often by as much as 10%, to compensate for possible future NBTI-induced delay [Abella et al. 2007]. This ensures that the processor will not fail due to NBTI degradation by sacrificing a significant percentage of the initially-available performance. In gate sizing, the sizes of the transistors are increased, thus increasing the initial speed of the circuit, so that the NBTI-degraded circuit will still meet the timing requirements. However, this technique imposes an 8%–12% area overhead and increases power consumption [Vattikonda et al. 2006]. Similarly, in V_{dd} and V_{th} tuning, the voltage of the circuit is adjusted to increase the initial operating speed [Vattikonda et al. 2006]. This technique has two problems. First, increasing the operating voltage increases the rate of NBTI degradation, requiring a further increase of V_{dd}. Second, increasing operating voltage increases the power consumption of the circuit. Techniques that minimize the NBTI degradation are needed.

2.3.2. Mitigating Techniques. Power-gating and clock-gating methods have been used to reduce the power consumption of idle functional units [Tsai et al. 2004]. In power gating, a *sleep transistor* that can be turned off to prevent any static or dynamic power consumption is added between the power supply and the functional unit. In clock gating, the clock input to the idle functional unit is disabled to prevent dynamic power consumption. This is often combined with Input Vector Control (IVC) to reduce the leakage power consumption. Leakage power consumption is dependent on the state of the inputs to a gate, and thus in IVC, the functional unit inputs are chosen to minimize the total leakage.

Both techniques could be used for NBTI degradation reduction. Power gating both reduces leakage power and eliminates NBTI degradation, but compared to clock gating and IVC, suffers from increased wake-up latency and must be activated for a minimum number of cycles in order to save energy. Thus, clock-gating methods are more suitable for short sleep durations.

Traditional power-gating architectures had a wake-up latency that was orders of magnitude longer than for clock-gated designs [Tsai et al. 2004]. Recently, Calimera et al. proposed a method for sequentially activating optimally-sized sleep transistors to limit the current draw during wake-up. They show that single-cycle reactivation is possible while limiting the wake-up current of a functional unit to its maximum active current and claim that this avoids ground bounce problems [Calimera et al. 2009]. This certainly prevents dangerous IR-drops that could disrupt neighboring units, but does not necessarily prevent dI/dt drops, which are dependent not on the magnitude of the current, but on its time derivative. Activating an idle functional unit can result

in dI/dt-related voltage emergencies as the current draw switches from pure leakage to both leakage and dynamic [Joseph et al. 2003]; activating a unit which has been power-gated to reduce leakage leads to an even greater change in current, even with the proposed limit on the maximum current. On-chip capacitance can eliminate this problem for short events, for example, single-cycle current bursts, but for the sustained current draw seen in a freshly activated functional unit, the dI/dt effects can still lead to voltage emergencies [Joseph et al. 2003]. Thus, even if it is possible to charge the virtual ground in a single cycle, additional idle cycles may be necessary to prevent dI/dt emergencies. Waking-up a clock-gated functional unit does not require discharging the virtual ground capacitance, implying that it can be accomplished quickly using a smaller change in current than power gating and thus may be safer for more temporally fine-grained control. In summary, despite recent advances in power-gating technology, it appears that clock gating continues to have advantages in the presence of dI/dt effects.

Although power gating reduces leakage current, enabling the gating consumes additional energy as the virtual ground voltage rises to V_{dd}. Thus, in order for power gating to save energy, it must be enabled for enough cycles that the savings due to reduced leakage are greater than the losses from charging the virtual ground. Usami et al. state that for typical arithmetic functional units, the break-even point occurs at about 40–100 clock cycles, depending on temperature [Usami et al. 2009]. Clock gating does not suffer from the virtual ground losses and thus is useful for shorter sleep durations.

Power gating is a useful technique for both reducing power consumption (static and dynamic) and preventing NBTI degradation when the function unit durations are long. However, clock gating with IVC (and our proposed internal node control extension) is useful for much shorter duration sleeping (down to one clock cycle). For functional units which are unused for a majority of clock cycles, but are still used frequently (e.g., on average every twentieth instruction uses a particular unit), clock gating with IVC will outperform power gating.

Wang et al. investigated the use of IVC to reduce NBTI degradation [Wang et al. 2007b, 2009a]. In practice, this control can be implemented either by placing MUXes on the inputs or by using a scan-chain. They find that IVC can reduce the degradation by an average 30%, but note that for many circuits, the input vector may only be able to control a few levels of the circuit's internal gates. Wang et al. predict that for future technologies, smaller gate sizings and higher temperatures may increase the benefit of such techniques.

In contrast with IVC, internal node control (our proposed technique) permits much greater control of all levels of the circuit, thereby allowing greater reduction in the NBTI-induced delay. Wang et al. independently and simultaneously published a related technique, gate replacement, at the same conference where our initial work on internal node control was presented [Bild et al. 2009; Wang et al. 2009b]. Our proposed technique, Internal Node Control (INC), is more general than gate replacement. With INC, any gate output can be forced either high or low by a sleep signal. With gate replacement, certain combinations cannot be represented. For example, a two-input NOR gate with a *low* sleep-mode output value can be represented by a three-input NOR gate with the third input driven by an active-low sleep signal. However, a two-input NOR gate with a *high* sleep-mode output value cannot be represented by a standard three-input gate. Wang et al. presented results for the co-optimization of leakage power and NBTI degradation. Our work is distinguished by the optimal MILP formulation, the subsequent comparison of the INC-placement heuristic to the optimal results, the analysis of multiple-vector INC, and the analysis of the sensitivity of INC to circuit size and primary output slacks.

3. INTERNAL NODE CONTROL

Internal Node Control (INC) refers to setting the states of individual nodes or gate outputs at any layer of the circuit to specific values. With this extension to IVC, further control and thus NBTI mitigation is possible. INC can be implemented by the addition of node control circuitry at the output of each controlled gate.

There are several important observations about INC insertion for NBTI minimization in CMOS. We first describe two specific implementations of INC, one for static CMOS logic originally developed for static power consumption minimization and one for pass-transistor CMOS logic. We then discuss the difficulty of removing NBTI stress from all PMOS transistors in a circuit and note a property of NOR gates that lessens the associated cost. Next, we explain the structural properties of transistors requiring NBTI stress removal and give our problem definition. We show that the problem is $\mathcal{NP}$-complete via a reduction from circuit-SAT.

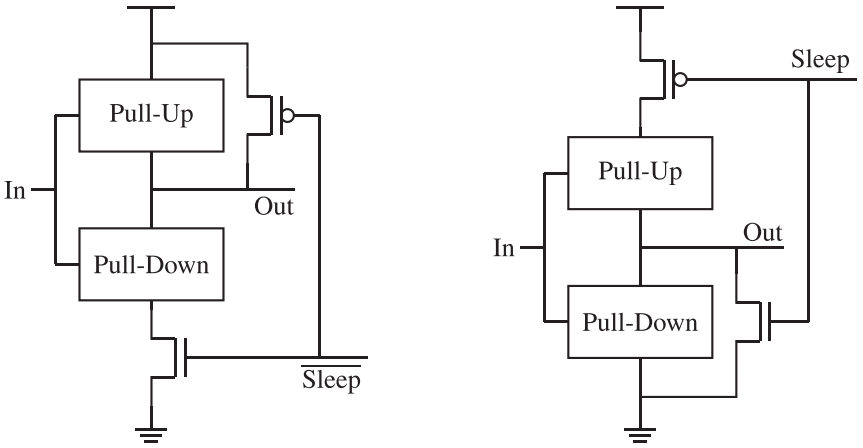
3.1. INC Implementation

Selectively forcing a node to a specific value requires modifying the driving gate. We describe two techniques for doing this, one for static CMOS logic and one for pass-transistor CMOS logic. For a typical standard cell flow, at least three versions of each cell must be included in the library: the unmodified cell, one with INC to force the output low, and one with INC to force the output high. Our INC methods are general to any static or pass-transistor logic cell and can be applied automatically by a CAD tool to an existing library. Furthermore, this additional design and characterization work is done only once, when the standard cell library is developed. Individual circuit designers are not impacted. Many libraries already support similar special-purpose versions of their cells for low-power designs and multithreshold processes.

For static CMOS logic gates (i.e., negative unate functions), we use a technique from Abdollahi, Fallah, and Pedram's work on leakage minimization [Abdollahi et al. 2004]. To force the output high, the output of the gate is connected to V_{dd} via a PMOS transistor in parallel with the existing pull-up network, controlled by an active-low sleep signal and the pull-down network is gated by a series NMOS. Note that this transistor is structurally equivalent to the sleep transistor used in some leakage power reduction schemes (e.g., MTCMOS) [Roy et al. 2003]. To force an output low, a similar modification is made, as illustrated in Figure 1. As mentioned in Section 2.3.2, for some n-input cells, an $n + 1$-input cell is equivalent to one of the INC cells, for example, a 2-input NAND with INC to force the output high is structurally equivalent to a 3-input NAND.

Pass-transistor logic is often used in standard cell libraries for nonunate primitives like XOR. Figure 2 shows an INC implementation for such cells. The output is forced high by the additional PMOS transistor and the pass-logic is gated from the output by an additional transmission gate. To force an output low, replace the PMOS transistor with an NMOS pull-down transistor. This technique requires three additional transistors, but is general.

More creative approaches can also be used for pass-transistor gates, at greater cost during library development. Note that with our INC technique, the gate inputs in the sleep state are fully determined at design time. Thus, an INC cell must only implement one of the rows of the cell's truth table with the sleep signal active (obviously, all rows with sleep inactive must be implemented). This allows a typical XOR gate to be modified to include INC with only two additional transistors and no additional capacitive output load, as shown in Figure 3. Similar designs could be manually constructed for other primitives.



(a) Node control to force output high. (b) Node control to force output low.

Fig. 1. Static CMOS gates modified to include node control [Abdollahi et al. 2004]. Two additional transistors required.

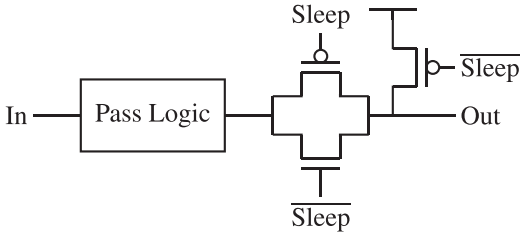
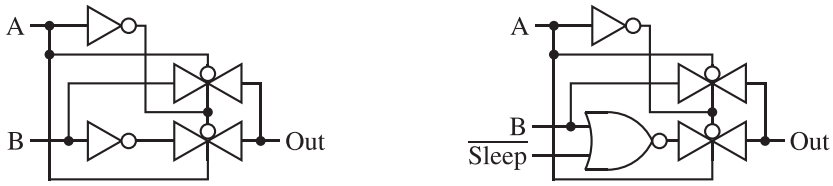


Fig. 2. General method to support INC for pass-transistor CMOS logic gates. Three additional transistors are used.

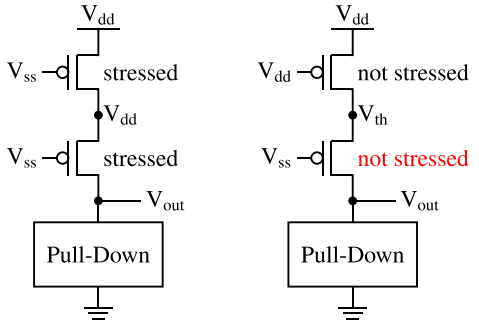


(a) XOR without INC. (b) XOR with INC.

Fig. 3. Pass-transistor XOR gate modified to force the output high (given that, in the sleep state, A=1). Only two additional transistors are used.

Unfortunately, the addition of this extra circuitry required for INC increases circuit delay. For a 65 nm Berkeley Predictive Technology Model [Cao et al. 2000; PTM 2010], this technique results in an ∼12.5% increase in delay for a simple inverter. The absolute delay increase is independent of gate type, so the percentage decreases for larger gates.

The delay overhead is smaller for more complex gates (e.g., a four-input versus a two-input gate), suggesting that INC might be most effective on circuits mapped with preference for such gates. This potential further optimization would require changes to the technology mapping stage (e.g., adjusting the cost functions employed by synthesis tools like Synopsys Design Compiler), so we do not further pursue it here. INC is beneficial even without special mapping considerations. Of course, a combined



(a) all inputs low and all transistors stressed

(b) only one input low, but all transistors not stressed

Fig. 4. For a NOR gate, destressing the top PMOS destresses all subsequent transistors in the stack.

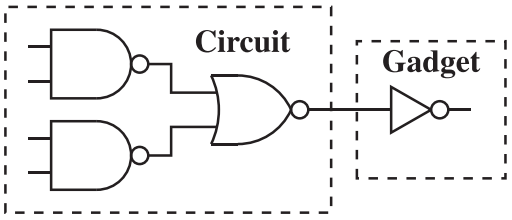


Fig. 5. Inverter *gadget* appended to output of circuit, for proof of $\mathcal{NP}$-completeness.

mapping/INC insertion methodology has the potential for even greater benefit and could be pursued in the future.

3.2. Potential of INC for CMOS

For an inverting logic implementation technology such as static CMOS, if all of the inputs to a gate are high, then the output will be low. Thus, it seems that in order to place nonstressing (high) values on all PMOS inputs, internal node control must be implemented at the output of *every* gate. Recall, however, that NBTI stress is due to negative bias between the source node and the transistor input ($V_{gs} = -V_{dd}$); it is not just due to the low gate voltage. For gates with parallel pull-up networks (e.g., inverters and NAND gates), the source node for each PMOS transistor is always at V_{dd} and each transistor is stressed whenever the input is low. For gates with series pull-up networks (e.g., NOR gates), the source node voltage, except for the top transistor in the PMOS stack, is dependent on the state of the transistors higher in the stack [Kumar et al. 2007]. Specifically, the source node voltage for any transistor below an "off" transistor will be close to ground and thus, even for a low input, V_{gs} will not approach $-V_{dd}$. This is illustrated in Figure 4. While this reduces (to one) the number of high inputs needed to eliminate static NBTI stress in a NOR gate, it does not help with the problem of inverting logic. A single high input to a NOR gate will force the output low and thus will still potentially stress the subsequent gate.

To eliminate static NBTI stress on all the PMOS transistors in a circuit, the outputs of most gates must be forced high. Gates feeding only into the lower PMOS transistors of NOR gates are the exception. For a single gate, the delay increase due to INC is similar to the delay increase due to NBTI stress. Consequently, covering every gate with INC will not lead to a net improvement in delay. Focused mitigation is required. That is, it is necessary to find the set of nodes for INC insertion that minimizes the overall circuit delay in the presence of NBTI.

The relevant transistors for NBTI stress removal are those on the critical path or those that, due to NBTI degradation over circuit lifetime, may ultimately be on this path. That is, a critical transistor is one with a timing slack less than its NBTI-induced increase in delay when considering the timing impacts of the NBTI mitigation technique. If all of these transistors can be placed in unstressed states, static NBTI will not increase system delay. Unfortunately, identifying these critical transistors is hard because the application of the mitigation technique (in our case, INC) changes the set of gates that are or could become critical. The slack for each gate depends on the delays of all the prior and subsequent gates along its path. Therefore, it is dependent on

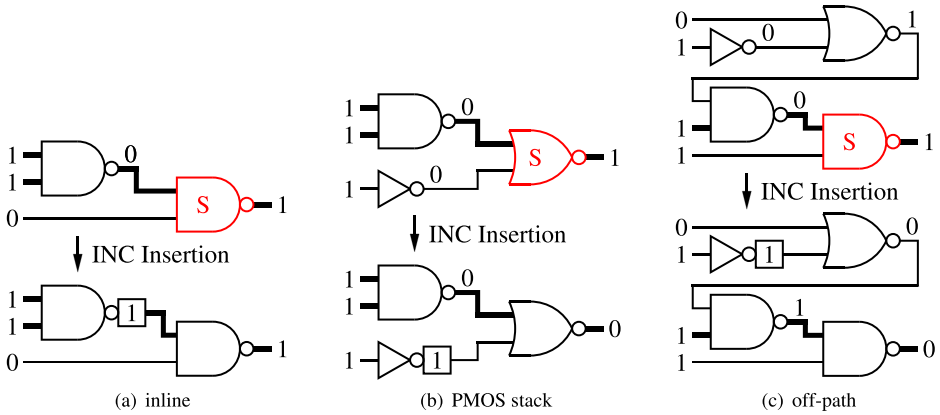


Fig. 6. Three example INC insertion scenarios. Gates affected by NBTI are drawn in red and labeled with an "S". Critical path lines are darkened. An INC insertion is shown by a square at the output of the affected gate.

the NBTI stress and node control delay of each of those gates as well. The addition of node control to a single gate can, in the worst case, change the slack of every other gate in the circuit. These control nodes introduce additional delay that, depending on their locations, may adversely affect the critical path. This interplay between the mitigation technique and the critical gates implies that noniterative static timing analysis-based methods [Wang et al. 2007], will not work for INC. It is thus necessary to optimally trade off the reduction in NBTI-induced delay and the increase in delay due to the addition of INC.

Figure 6 shows three example INC insertion scenarios. These are intended as examples of a subcircuit far removed from the primary inputs of the circuit containing it. IVC loses effectiveness as circuit depth increases, and thus the input vectors for these examples are fixed and IVC is not considered. The critical path in each circuit is shown in bold, and critical path gates stressed by NBTI are labeled with an "S". For simplicity, in these examples we assume that INC insertion does not change the location of the critical path, but only its delay. More specifically, we assume sufficient slack for INC insertion. Note that the MILP and heuristic solutions developed in Sections 4 and 5 seek the delay-optimal solution considering both NBTI and INC delays.

Figure 6(a) shows the insertion of node control inline with the critical path. In this case, the delay added by the node control on the first gate must be less than the NBTI delay on the second gate for there to be a net decrease in delay. Figure 6(b) illustrates the removal of NBTI from a NOR gate using the previously explained observation about the series PMOS stack. The last example is more complicated. In Figure 6(c), NBTI stress is removed from the second NAND gate by inserting an INC node off the critical path such that the correct value propagates through to the critical gate. In this scenario, the node control can be added to a gate with sufficient slack, even if that gate is several gates removed from the critical path. Note that although the second NAND gate is stressed by NBTI after INC insertion, the stress does not occur on a critical path input and therefore does not increase total circuit delay. In Sections 4 and 5, we describe optimal and heuristic methods to select the gates to modify with INC.

3.3. Problem Definition

We consider INC insertion as a post-synthesis step in the design process intended to reduce NBTI guard bands. Figure 7 illustrates the design flows both with and without

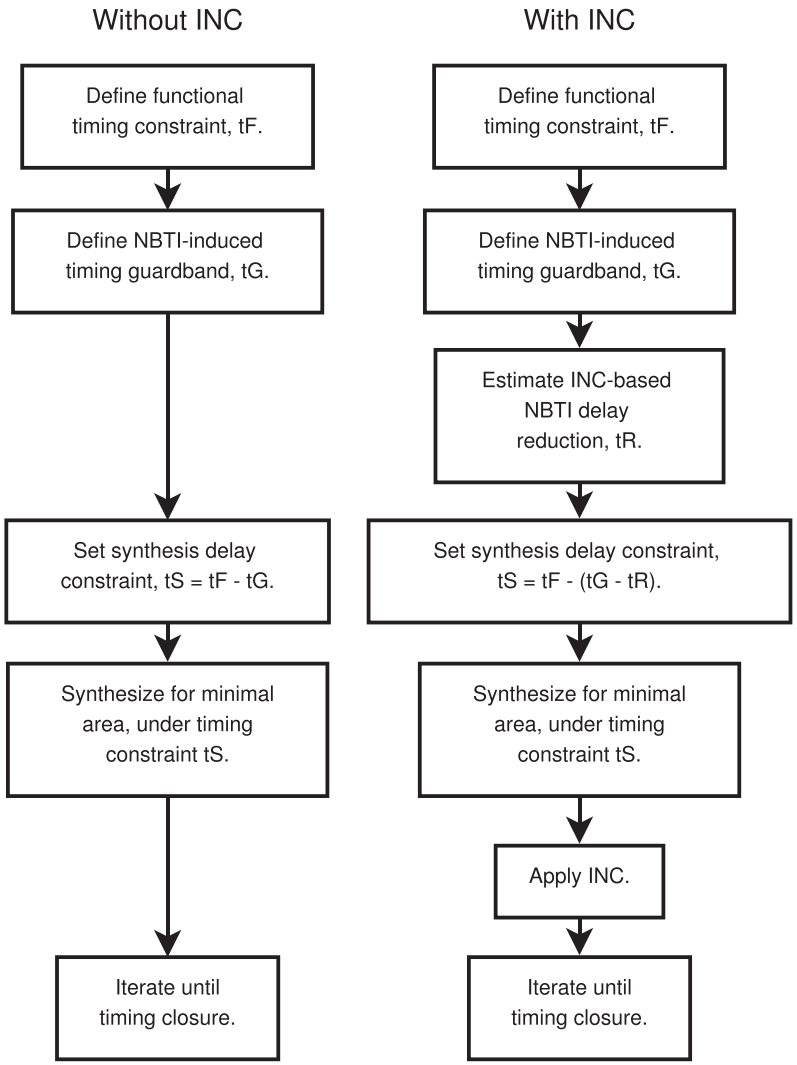


Fig. 7. Comparison of synthesis design flows with and without our INC-based NBTI reduction technique.

the use of INC. In the traditional design flow, an NBTI guard band is subtracted from the target delay to determine the synthesis delay constraint, under which the synthesis process optimizes area and power. With INC, the flow is largely the same, but the guard band is reduced by first estimating the impact of INC when determining the synthesis delay constraint and then applying INC after synthesis.

We formalize INC insertion for this design flow as the following optimization problem. The input consists of the combinational circuit output by the synthesis routine, represented as a graph of connected gates. For each gate, three delays are specified: (i) the basic delay for an unmodified gate, (ii) the increase in delay if INC is added, and (iii) the increase in delay after some period of NBTI stress, for example, 10 years. The task is to find the input vector and node control insertion points that minimize the critical path delay after it has been subjected to NBTI stress. In other words, the

goal is to minimize the increase in delay between the original unstressed circuit and the INC-modified, stressed circuit.

3.4. Alternate Problem Definition

The preceding formulation (the one used in this article) assumes that area and power are essentially fixed when INC is applied post-synthesis, and thus delay is the optimization objective. This formulation has two advantages: (1) the impact of INC is easy to measure as the reduction in circuit delay and (2) it integrates easily in existing design flows with existing synthesis tools. The first benefit simplifies comparing alternative solutions in Section 6. It is, however, the second benefit that led us to choose the constrained-area formulation.

It would be possible to instead constrain delay and optimize some combination of power and area (instead of optimizing delay). However, this would require that the problem definition be broadened to provide a mechanism for controlling area, such as changing the technology mapping solution. As a result, this apparently straightforward change in optimization constraints and objectives would couple multiple steps of the design process. Solving this new problem well would require integrating the solutions to the technology mapping and INC insertion problems, requiring a global change to the design flow that would interfere with practical use. However, future research on the alternate formulation has the potential to yield better results.

3.5. INC Insertion is $\mathcal{NP}$-Complete

We show that the decision version of this problem is $\mathcal{NP}$-complete. In the decision problem, instead of minimizing the critical path delay, a delay bound b is specified and the task is to determine whether an input vector and set of INC placements exist such that the critical path delay is less than or equal to b.

LEMMA 3.1. *The problem of IVC selection and INC placement for NBTI minimization is in $\mathcal{NP}$.*

PROOF. A solution can be easily checked in polynomial time by computing the associated critical path delay. Specifically, the delay can be determined in time linear to the number of gates via a simple topological traversal of the circuit, computing gate output values and arrival times. □

LEMMA 3.2. *The problem of IVC selection and INC placement for NBTI minimization is $\mathcal{NP}$-hard.*

PROOF. To prove that the problem is $\mathcal{NP}$-hard, we use a reduction from circuit-SAT. In circuit-SAT, the task is to decide, for a given Boolean circuit C with a single output, if there is an assignment to the inputs such that the output is true [Garey and Johnson 1979]. We give a polynomial-time transformation from an instance of circuit-SAT to our problem with specified bound $b = 0$.

For each gate in the circuit C, the intrinsic delay and NBTI delay are set to 0. This ensures that for any inputs, the critical path delay is 0. The INC delays are set to a positive value (e.g., 1), thus ensuring that a solution satisfying the bound $b = 0$ will not have any INC nodes and that the Boolean function implemented by the circuit remains unmodified.

We add a *gadget* to the output of the circuit in such a way that the critical path delay is greater than 0 if the output is false, and 0 if the output is true. Specifically, an inverter is inserted at the output of the circuit, as shown in Figure 5. The basic delay is set to 0 and the NBTI delay is set to a positive value (e.g., 1). The INC delay is unimportant but can be set to 0. If the output of the original circuit C is true, the

inserted inverter will not be stressed by NBTI, and the critical path delay will be 0. If the output is false, the inverter will be stressed and the delay will be positive, thereby exceeding the bound $b = 0$.

In short, any circuit-SAT problem instance can be solved as an instance of our problem using this transformation. The circuit-SAT instance has an accepting input assignment if and only if the transformed problem has an input assignment leading to a critical path delay of 0. □

THEOREM 3.3. *The problem of IVC selection and INC placement for NBTI minimization is $\mathcal{NP}$-complete.*

PROOF. This follows from Lemma 3.1 and Lemma 3.2. □

4. OPTIMAL FORMULATION

To further formalize our problem description, we describe it as a Mixed Integer Linear Program (MILP) formulation that may be used to find optimal solutions. Additionally, this formulation is used in Section 6 to show that our proposed heuristic produces near-optimal results.

A combinational circuit is modeled as a directed acyclic graph $G = (V, E)$. V is a set of primary inputs ($I \subset V$), gate outputs ($N \subset V$), and primary outputs ($Q \subset V$). E is a set of directed edges modeling connections between two gates. The gate outputs N, are further divided into three sets N_I, N_R, and N_D representing NOT, NOR, and NAND gates. P_v are the predecessors of gate v.

The intrinsic delay of a gate is $d_{int}(n)$. The increase in delay due to NBTI stress is $d_{nbti}(n)$, and the increase in delay due to the addition of node control on the gate output is $d_{inc}(n)$.

The following variables are used. c_n is a binary variable.

$$c_n = \begin{cases} 1 & \text{if INC is added to gate } n \\ 0 & \text{otherwise} \end{cases}$$

f_n is a binary variable representing the forced value of node n, if c_n is 1. $0 \le x_v \le 1$ is the value of node v. If c_v is 1, then x_v is f_v. Otherwise, it is determined by the inputs to the gate. For $v \in I$, x_v is explicitly constrained to be binary. t_v is the earliest arrival time at node v.

We optimize the circuit delay by minimizing the maximum output arrival time.

$$\text{minimize} \max_{\forall q \in Q} t_q \tag{1}$$

The Boolean function of the gates, combined with the node control, is modeled by a set of constraints that force each output x_v to the proper value based on c_v, f_v, and the inputs to node v. These constraints are equivalent to those specifying the convex hull of the function, where each input and output represents one dimension. For example, the truth table for an inverter, shown in Figure 8, leads to the following constraints. NAND and NOR gates are similarly determined.

$$\begin{aligned} \forall n \in N_I : \; & c_n + f_n - x_n \le 1 \\ & c_n - f_n + x_n \le 1 \\ & -f_n + x_n + x_p - 1 \le 0 \\ & -c_n + x_n + x_p - 1 \le 0 \\ & -x_p + f_n - x_n \le 0 \\ & -x_p - c_n - x_n \le -1 \end{aligned}$$

x_p	c_n	f_n	x_n
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Fig. 8. Truth table for an inverter with INC. x_p is the input, c_n is the INC selection variable, f_n is the forced value, and valn is the output.

The earliest arrival times are modeled by constraining a node v's arrival time to be later than or equal to all of its inputs' arrival times plus any delays associated with the gate. The intrinsic delay $d_{int}(v)$ of each gate is always included. The internal node control delay $d_{inc}(v)$ is only included if c_v is 1. The NBTI delay $d_{nbti}(v)$ is included when, based on the inputs, the gate is stressed. For NOT and NAND gates, the following constraint enforces this relationship.

$$\forall n \in N_I \cup N_D, \forall p \in P_n : t_n \geq t_p + d_{int}(n) + (1 - x_p)d_{nbti}(n) + c_n d_{inc}(n)$$

As discussed in the previous section, if any input to a NOR gate is high, we assume that the whole gate is unstressed. The variable $0 \leq s_n \leq 1$ is 1 if NOR gate $n \in N_R$ is stressed and 0 otherwise. Thus, the following constraints implement the arrival time computation for NOR gates.

$$\forall n \in N_R, \forall p \in P_n : 1 - s_n \geq x_p$$
$$1 - s_n \leq \sum_{r \in P_n} x_r$$
$$t_n \geq t_p + d_{int}(n) + s_n d_{nbti}(n) + c_n d_{inc}(n)$$

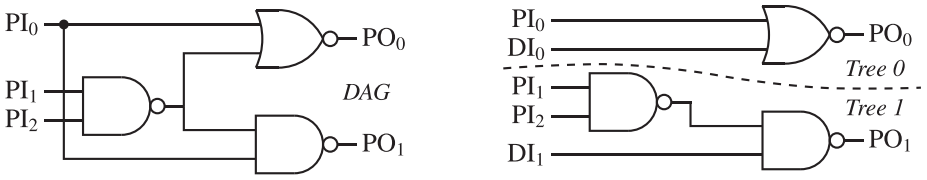
Optimization Objective 1 ensures that the arrival times on the critical path are minimal.

5. HEURISTIC SOLUTION

The MILP-based optimal solution method is not practical for large circuits because this problem is $\mathcal{NP}$-complete. A heuristic solution that provides good, and ideally near-optimal, solutions in a reasonable amount of time is necessary. In this section, we describe a linear-time algorithm for input vector selection and internal node control placement. Our technique is inspired by work on leakage power minimization by Cheng et al. [2008], but differs by appropriately handling the nonadditive cost function required for the INC placement problem.

5.1. Overview

Our heuristic (see Algorithm 1) takes advantage of the fact that the problem can be solved optimally for rooted-tree structures in linear time. It first partitions a given circuit into rooted trees by removing some connections between gates (line 1). This partitioning creates *dangling inputs* at these gates whose input connections were removed, as illustrated in Figure 9. Values are assigned to these dangling inputs (line 2) and the optimal values for the primary inputs and INC placements are chosen for each partition (lines 4–6). The values for the dangling inputs are updated based on the new outputs of their parent gates in the original circuit (line 7) and the solutions for the



(a) unpartitioned directed acyclic graph (b) partitioned into two trees

Fig. 9. A circuit partitioned into rooted trees, with the *dangling inputs* labeled DI_x.

ALGORITHM 1: INC Placement Heuristic Overview

Require: circuit $\mathcal{G}$
Require: maximum number of iterations, N
1: partition circuit into trees
2: select initial values for dangling inputs
3: **for** $i = 0$ to N **do**
4: **for all** partitions **do**
5: choose IVC and INC using tree-optimal algorithm
6: **end for**
7: update dangling input values
8: **if** solution is the same as previous **then**
9: break {Solution has converged.}
10: **end if**
11: **if** oscillation is detected **then**
12: repartition the circuit
13: **end if**
14: **end for**
15: greedily remove INCs which do not affect delay
16: **return** input vector and INC placements

partitions are recomputed based on these new dangling input values (line 3). This iteration continues until the solution has converged (lines 8–10) or a preset number of iterations has been reached (line 3). Convergence is identified when the values for the dangling inputs do not change between two consecutive iterations. To ensure convergence, when the revisitation of a solution is detected, the circuit is repartitioned (lines 11–13). Empirical results show that this repartitioning breaks oscillations and leads to convergence.

5.2. Partitioning and Initial Solution

Solution quality is highly dependent on the method used to partition the circuit and the initial values assigned to the dangling inputs. Tree-based partitioning has been proposed for several circuit design problems in the past, including leakage power minimization and technology mapping [Cheng et al. 2008]. For these problems, the cost function (e.g., total leakage power and circuit area) is additive: the overall cost is essentially the sum of the costs of the individual partitions. It is thus important to maximize the sizes of the partitions in order to maximize the effectiveness of the tree-optimal algorithm. The specific choice of which connections to remove, though, is not as critical.

For INC placement, the cost function is not additive: the critical path delay for the entire circuit is not the sum of the critical path delays of each partition. Thus, in addition to maximizing the sizes of the partitions, it is also important to keep the

ALGORITHM 2: Tree-Optimal Algorithm

Require: tree-structured circuit partition $\mathcal{P}$
Require: arrival times and node values for dangling inputs {Forward Pass}
for all gates g in a topological ordering of $\mathcal{P}$ **do**
  for all combinations of inputs i **do**
    compute arrival time and output value using the arrival times of g's parent gates
    compute arrival time and output value if INC is added
  end for
  store i with a 0 output and the smallest arrival time
  store i with a 1 output and the smallest arrival time
end for
{Backward Pass}
choose primary output value with smallest arrival time
for all gates g in a reverse topological ordering of $\mathcal{P}$ **do**
  select the stored i with the output that matches the child's selected i
end for
return the input values and the INC placements

original critical path in a single partition. Of course, for circuits with parallel critical paths, this will not always be possible. Our partitioning algorithm avoids cutting these critical paths by using slack information to determine which connections to remove. In a rooted-tree structured circuit, each gate has a fanout of 1. Thus, for each gate with a fanout greater than 1, our partitioning algorithm keeps the connection with the smallest slack, removing the others. Dangling inputs are inserted at the broken connections.

Computing the slack values for initial partitioning requires an initial IVC and INC assignment. We choose these initial values by applying the tree-optimal algorithm to the unmodified directed acyclic circuit. The circuit is not tree-structured and as a result, conflicts will occur in the backward pass phase of the algorithm. At each gate with a fanout greater than 1, the child gates may require differing output values from their shared parent. In these cases, the value required by the majority of the children is chosen. In the case of a tie, 1 is chosen because, in general, it will prevent NBTI stress on the child gates. After partitioning, the dangling inputs are set to the values from this initial partitioning.

For circuits with reconvergent critical paths, it will not be possible to place both critical paths in the same partition. This can occur, for example, in highly optimized circuits in which many paths are critical or near-critical. To show that the heuristic still performs well in such cases, its performance on large industrial-scale benchmarks with many critical and near-critical paths is presented in Section 6.1.

When oscillation is detected during the iterative solution process, the circuit is repartitioned. Slack values are computed using the current solution, with dangling inputs reconnected to and taking the value of their original driving gates. Because the slack values are likely different from the initial partitioning, this results in a different partitioning, breaking the oscillation and commonly reassigning the possibly different critical path to a single partition.

5.3. Tree-Optimal Algorithm

The tree-optimal algorithm is shown in Algorithm 2. The algorithm takes as input a tree-structured circuit partition and, for each of the dangling inputs, the arrival time and node value. For primary inputs, the arrival time is assumed to be 0 and the node value is determined by the algorithm. The algorithm consists of two phases, the

forward pass and the backward pass. In the forward pass, two pieces of information are computed for each gate, the input combination and INC state with a 0 output and the smallest arrival time, and the input combination and INC state with a 1 output and smallest arrival time. Specifically, the gates are examined in topological order (line 1). For each gate, each possible input combination is examined (line 2). The output value is computed and, using the arrival times previously computed for the parent gates, the arrival time is computed (line 3). The value and arrival time if INC are added is also computed (line 4). For each output value, 0 and 1, the input combination and INC state with the smallest arrival time is stored (lines 6–7). It is possible for several input vectors to lead to the same minimum arrival time. In the case of such ties, the number of nonstressing inputs is maximized by choosing the *covering* input vector. An input vector x is said to cover input vector y if x has 1 values in each position that y does and at least 1 additional position. For example, the vector "1011" covers the vector "1010". The intuition is that nonstressing values are preferred for NBTI minimization and thus are less likely to conflict via the dangling inputs with assignments in other partitions. In the backward phase, a specific value (and thus INC state) is chosen for each of the gates. Specifically, the primary output value and corresponding input combination with the smallest arrival time is chosen (line 9). The remaining gates are then examined in a reverse topological order (line 10). For each gate, the required output value is specified by the chosen input combination for its child. The corresponding inputs are chosen for the gate (line 11).

5.4. Runtime

The heuristic requires time linear in the number of gates. Partitioning is performed with a single topological traversal. The tree-optimal algorithm requires one traversal for each phase. Although all the input combinations for each gate must be examined, this is effectively constant time because the number of inputs is restricted. Finally, the overall algorithm iterates multiple times, but empirical results show that it converges rapidly and thus the maximum number of iterations can be limited to a small constant. 15 iterations were used in all our reported results and were sufficient for problem instances with widely varying sizes.

6. EXPERIMENTAL RESULTS

We implemented the proposed heuristic in Python and evaluated it on both the ISCAS85 combinational benchmarks [Brglez and Fujiwara 1985] and a set of larger benchmarks from the Synopsys DesignWare Library. After a brief explanation of the experimental setup, this remainder of this section presents the results and analysis of this evaluation.

The ISCAS85 benchmarks are used because of their ubiquity and small size. Due to their relatively small gate counts, the ISCAS85 benchmarks are appropriate for use with the optimal MILP formulation and thus are used in Section 6.2 to show the near-optimality of the heuristic. However, real industrial designs for which INC is most useful generally will be much larger. The DesignWare benchmarks are used to verify the effectiveness on INC on larger circuits and are described in Table I. We have included all available architectures for designs with multiple architectures. Each design was synthesized for the strictest timing deadline that could be met by Design Compiler.

The benchmarks circuits were mapped to a seven gate library {inv, nor2, nor3, nor4, nand2, nand3, nand4} using Synopsys Design Compiler. For consistency, the gates were sized for a maximum fanout of three. Timing information for the gates (with and without node control) was obtained through HSpice simulations using the 65 nm

Table I. Descriptions of the DesignWare Benchmarks

Circuit	Function	Architecture	Timing Constraint (ps)	Gate Count	Heuristic Runtime (s)
add64_rpl	64-bit integer adder	ripple-carry	1100	1263	1.3
add64_cla		carry-lookahead	1100	1249	1.0
add64_pparch		parallel-prefix	1100	1172	2.0
addsub64_rpl	64-bit integer adder/subtractor	ripple-carry	1200	1522	1.9
addsub64_cla		carry-lookahead	1200	1487	2.7
addsub64_pparch		parallel-prefix	1200	1476	1.7
mult32_csa	32-bit integer multiplier	carry-save	8500	15157	32.7
mult32_pparch		parallel-prefix	2700	9736	20.5
ash64_mx2	64-bit arithmetic shifter	2:1 multiplexers	700	1617	2.1
ash64_str		speed-optimized	900	1556	2.3
ash64_astr		area-optimized	900	1556	2.3
bsh64_mx2	64-bit barrel shifter	2:1 multiplexers	600	981	0.7
bsh64_str		speed-optimized	600	981	0.7
bsh64_astr		area-optimized	600	981	0.7
fp_addsub64_rtl	64-bit floating point adder/subtractor	area-optimized	5400	5705	11.2
fp_addsub64_str		speed-optimized	5100	7362	14.2
fp_mult32_rtl	32-bit floating point multiplier	only choice	4200	8845	17.7
crc64_str	64-bit CRC-32 cyclic redundancy checker	only choice	3400	2275	4.1

Berkeley Predictive Technology Model [Cao et al. 2000; PTM 2010]. The timings for these self-developed gates, without node control, were calibrated to similar gates in a TSMC 65 nm library to ensure that the timings were representative of real-world libraries [TSMC 2006]. The static NBTI model presented in Section 2.1 is used to calculate the static NBTI delay. Separate results are reported for the 3.3% and 16.3% path degradation cases (corresponding to 10% and 50% stressed-gate degradation percentages).

All tests were done on a 3.0 GHz Intel Core2 Duo E8400 processor with 4GB of RAM. The runtimes are shown in the "Heuristic Runtime" column of Table I.

6.1. Heuristic Experimental Results

The results for all benchmarks are presented in Table II. Column "Baseline Delay" shows the circuit delay before NBTI stress and without the addition of INC. Separate results for the 3.3% and 16.3% path delay degradation percentages are shown side-by-side. For columns expressing a percentage change in delay, the bottom three rows of Table II show the average change over the ISCAS85 benchmarks, DesignWare benchmarks, and all benchmarks, respectively.

The "IVC Delay" columns present the delay considering NBTI stress and the application of near-optimal Input Vector Control, the prior work against which we compare. In Section 6.2 we note that minimum delay seen over a set of 10,000 random input vectors is close to the delay for the optimal input vector. Thus, we use this minimum delay here as proxy for the theoretical optimal input vector. These delays are reported both in absolute terms (ps) and as percentage increases in delay over the baseline. The 3.3% (16.3%) path delay degradation corresponds to 10% (50%) per-stressed gate delay increase.

The absolute delays when INC is added are shown in the "INC Delay" columns. The percent improvement in delay with respect to IVC is shown in the "Improvement"

Table II. Heuristic Results

Circuit	Baseline Delay (ps)	3.3% NBTI Path Degradation					16.3% NBTI Path Degradation				
		IVC Delay		INC Delay	Improvement		IVC Delay		INC Delay	Improvement	
		(ps)	(%)	(ps)	NBTI (%)	Total (%)	(ps)	(%)	(ps)	NBTI (%)	Total (%)
ISCAS85 Benchmarks											
c432	1650.6	1701.3	3.1	1699.4	3.7	0.1	1910.5	15.8	1748.1	62.5	8.5
c499	1588.7	1641.9	3.4	1628.0	26.3	0.9	1858.9	17.0	1690.1	62.5	9.1
c880	1884.3	1938.3	2.9	1914.0	45.1	1.3	2172.6	15.3	1982.2	66.0	8.8
c1355	1505.1	1557.4	3.5	1547.7	18.5	0.6	1774.6	17.9	1594.2	66.9	10.2
c1908	2112.0	2176.9	3.1	2175.7	1.8	0.1	2440.5	15.6	2257.3	55.8	7.5
c2670	1607.1	1657.9	3.2	1629.1	56.8	1.7	1861.4	15.8	1643.5	85.7	11.7
c3540	2546.4	2624.7	3.1	2595.7	37.1	1.1	2975.0	16.8	2614.0	84.2	12.1
c5315	2396.9	2455.7	2.5	2435.8	33.9	0.8	2694.9	12.4	2454.1	80.0	8.9
DesignWare Benchmarks											
add64_rpl	1097.0	1134.9	3.5	1131.8	8.2	0.3	1321.8	17.0	1267.7	24.1	4.1
add64_cla	1099.9	1135.4	3.2	1121.0	40.7	1.3	1437.5	23.5	1268.7	62.5	10.5
add64_pparch	1068.4	1101.9	3.1	1094.4	22.5	0.7	1256.5	15.0	1127.6	68.5	10.3
addsub64_rpl	1198.5	1240.8	3.5	1224.9	37.6	1.3	1434.2	16.4	1389.5	19.0	3.1
addsub64_cla	1198.5	1241.2	3.6	1243.5	−5.4	−0.2	1440.5	16.8	1313.8	52.4	8.8
addsub64_pparch	1195.9	1227.1	2.6	1215.5	37.0	0.9	1413.8	15.4	1322.6	67.4	6.5
mult32_csa	8476.2	8769.8	3.5	8636.6	45.4	1.5	10127.6	16.3	9218.5	55.1	9.0
mult32_pparch	2694.8	2785.1	3.4	2754.1	34.3	1.1	3231.7	16.6	3037.7	36.1	6.0
ash64_mx2	698.5	716.4	2.6	708.9	41.8	1.0	829.5	15.8	741.0	67.6	10.7
ash64_str*	899.8	927.0	3.0	925.8	4.3	0.1	1066.6	15.6	944.8	73.0	11.4
ash64_astr*	899.8	927.0	3.0	925.8	8.3	0.1	1066.6	15.6	944.8	73.0	11.4
bsh64_mx2	579.4	601.0	3.7	590.2	50.0	1.8	687.5	15.7	597.0	83.7	13.2
bsh64_str	579.4	601.0	3.7	590.2	50.0	1.8	687.5	15.7	597.0	83.7	13.2
bsh64_astr	579.4	601.0	3.7	590.2	50.0	1.8	687.5	15.7	597.0	83.7	13.2
fp_addsub64_rtl	5398.0	5601.5	3.8	5538.8	30.8	1.1	6521.7	17.2	5832.2	61.4	10.6
fp_addsub64_str	5095.0	5273.3	3.5	5196.8	42.9	1.5	6096.0	16.4	5474.0	62.1	10.2
fp_mult32_rtl	4199.4	4359.0	3.8	4303.3	34.9	1.3	5041.8	16.7	4589.5	53.7	9.0
crc64_str	3394.7	3516.2	3.6	3459.0	47.1	1.6	4033.5	15.5	3786.5	38.7	6.1
Average											
ISCAS85		**3.1**		**27.9**	**0.8**			**15.8**		**70.5**	**9.6**
DesignWare		**3.4**		**32.3**	**1.1**			**16.5**		**59.2**	**9.3**
All Circuits		**3.3**		**30.9**	**1.0**			**16.3**		**62.7**	**9.4**

* The synthesized netlist was nearly identical for both architectures.
The synthesized netlist was nearly identical for all three architectures.

columns and is computed in two different ways: percent reduction in NBTI-induced delay (Column "NBTI")

$$\%_{improve,nbti} = -100 \times \frac{(D_{inc} - D_{base}) - (D_{ivc} - D_{base})}{(D_{ivc} - D_{base})} \tag{2}$$

and percent reduction in total path delay (Column "Total")

$$\%_{improve,total} = -100 \times \frac{D_{inc} - D_{ivc}}{D_{ivc}}. \tag{3}$$

We expect the percent improvement in NBTI-induced delay to increase when the NBTI-induced delay grows relative to the INC overhead delay. This hypothesis is confirmed, with the results showing an average 30.9% decrease for the 3.3% NBTI path degradation and a 62.7% decrease for the 16.3% degradation. These improvements indicate the effectiveness of INC; one-third to over one-half of the NBTI-induced delay not handled by IVC is prevented by the addition of INC.

The percent improvement in total path delay averages 1.0% for the 3.3% degradation case. This is expected because the NBTI degradation is such a small percentage.

For the 16.3% degradation case, the results are more significant. The total path delay is reduced by 9.4%, which is equivalent to one speed bin for a typical microprocessor.

The INC heuristic reduces the delay compared with near-optimal IVC for all benchmarks except "addsub64_cla", which shows a 5% increase in NBTI delay over just IVC for the 3.3% degradation case. On the other extreme, the three "bsh_X" shifter circuits show a 50% decrease in NBTI delay for the 3.3% case and a 83.7% decrease for the 16.3% case. There is significant variation across benchmarks. For example, the ISCAS85 benchmarks with the 3.3% degradation estimate have NBTI-induced delay reductions ranging from 3.7% to 56.8%. Section 8 discusses two potential causes of this variation.

These results show that INC can significantly reduce the amount of NBTI-induced performance degradation on idle-mode functional units. They also show that the impact on total delay is substantial when NBTI-induced delay is significant. Our proposed heuristic is effective for INC placement and efficient even for large-scale circuits.

6.2. Comparison with Optimal Solutions

The preceding results indicate the effectiveness of INC but, as heuristic solutions, do not indicate whether better INC placements, and thus further reductions, are possible. To answer this question, we obtained optimal solutions for the ISCAS85 benchmarks using the MILP formulation in Section 4. In this section, we use those results to show that, for small problem instances, the heuristic produces near-optimal solutions. We also show that the theoretical minimal IVC delay is closely approximated by the minimum delay observed over a set of 10,000 random input vectors.

The INC problem is $\mathcal{NP}$-complete, so determining optimal values is feasible only for small problem instances; even for those it is time-consuming. Consequently, we present optimal results only for the ISCAS85 benchmarks for the conservative 3.3% path degradation (10% stressed-gate delay degradation) estimate. The MILP formulation (Section 4) was solved using the open-source software SYMPHONY [Ralphs and Guzelsoy 2005] for two different cases.

(1) To determine the delay using optimal input vector control only, the solver was run with INC disabled (i.e., the node control selection variables were forced to 0).
(2) The solver was also run with internal node control enabled.

The resulting problem instances are rather large for an MILP solver. Therefore, we stopped the solver when the upper and lower bounds for the optimal delay were within 0.2% of each other. Some benchmarks did meet the 0.2% stop gap after several days of execution. Thus, we also terminated execution after 24 hours and report the lower and upper bounds determined by the solver. For instances that were successfully solved, these bounds are equal.

The results are presented in Table III. For columns representing a percent difference, the average difference across all benchmarks is presented in the last row.

The columns under the "IVC Delay" heading show the minimum delay over a set of 10,000 random input vectors ("Random Set" columns), the lower and upper bounds[2] on the optimal delay for the IVC technique ("Optimal" columns), and the percent difference between the random set estimate and optimal ("Change" columns). The random set estimates are on average only 0.12% higher than the MILP upper bound and can be computed much more quickly. The lower bounds reported by the MILP solver may

[2] The upper bound is the delay for the best solution found by the MILP solver. The lower bound is the highest value above which the solver proved that the true minimum must lie. There may not exist a solution with the lower bound delay.

Table III. Comparison with Optimal Results for ISCAS85 Circuits

Circuit	IVC Delay					INC Delay				
	Random Set (ps)	Optimal (ps)		Change (%)		Heuristic (ps)	Optimal (ps)		Change (%)	
		LB	UB	LB	UB		LB	UB	LB	UB
c432	1701.3	1697.9	1701.3	0.20	0.00	1699.4	1690.7	1691.8	0.51	0.45
c499*	1641.9	1633.0	1641.2	0.55	0.04	1628.0	1626.7	1629.6	0.08	−0.10
c880	1938.3	1926.7	1927.9	0.60	0.54	1914.0	1911.4	1914.0	0.14	0.00
c1355*	1557.4	1546.8	1555.5	0.69	0.12	1547.7	1534.6	1541.5	0.85	0.40
c1908	2176.9	2175.0	2176.1	0.09	0.04	2175.7	2171.3	2175.7	0.20	0.00
c2670*	1657.9	1651.3	1657.1	0.40	0.05	1629.1	1627.5	1629.1	0.10	0.00
c3540*	2624.7	2615.6	2624.7	0.35	0.00	2595.7	2595.7	2598.4	0.00	−0.10
c5315	2455.7	2448.2	2450.9	0.31	0.20	2435.8	2435.6	2435.8	0.01	0.00
Average				**0.40**	**0.12**				**0.24**	**0.08**

* Solver was stopped after 24 hours but before the 0.2% stop gap was reached.

not be tight, so the average 0.40% difference with the MILP lower bound is a conservative estimate of the quality of the random set estimate. In the worst case, benchmark c1355, the heuristic solution is still less than 1% slower than the conservative optimal lower bound. Based on these comparisons, we conclude that the random set estimate for IVC is nearly optimal.

The columns under the "INC Delay" headings show similar data for the INC heuristic. In this case, we see that our heuristic produces solutions with delays 0.08% higher on average than the best solutions found by the MILP solver and only 0.24% higher than the conservative lower bound. We conclude that for circuits amenable to optimal solution by the MILP solver, our heuristic produces near-optimal results.

6.3. Consideration of PBTI

Our presentation has focused on mitigating NBTI stress, traditionally a more significant problem than its PBTI counterpart [Li et al. 2004]. However, PBTI is also becoming important with the adoption of high-K gates [Zafar et al. 2006], so we show that INC is applicable to the joint reduction of NBTI and PBTI stress. This claim might seem counter-intuitive—placing high values on internal nodes to reduce NBTI degradation would increase PBTI degradation—but our heuristic for INC is more general. It seeks the input vector and INC placements that minimize the aged-circuit delay, whether the delay is due to NBTI, PBTI, or some combination thereof.

Table IV compares the reductions in delay from our INC heuristic for the benchmark circuits considering NBTI only and considering both NBTI and PBTI. The PBTI degradation is modeled as 50% of the NBTI degradation, based on prior publications [Kumar et al. 2011]. The absolute circuit delays are obviously higher when considering PBTI, but the resulting improvements in delay versus IVC alone are similar for both cases—30.9% versus 28.0% average improvement—indicating the effectiveness of our technique for simultaneous NBTI and PBTI mitigation.

6.4. Scaling with Severity of NBTI

The previous sections presented the relative impact of INC for a conservative value of total NBTI degradation, 3.3% increase in path delay over 10 years (10% increase per stressed gate). This relative improvement is dependent on the overhead of the INC technique, the inverse of which is given by the NBTI delay expressed as percentage of INC delay. Figure 10 shows this relationship averaged over the full set of DesignWare benchmarks. For typical INC delays, the range of NBTI delay percentages (50% to 1,300%) represent path NBTI degradations ranging from 2% to 40% (i.e., gate-level NBTI degradations ranging from 5% to 90%). The relative improvement ranges from less than 25% to over 60%.

Table IV. Heuristic Results when Modeling PBTI

Circuit	NBTI Only Improvement		NBTI+PBTI Improvement	
	NBTI (%)	Total (%)	BTI (%)	Total (%)
ISCAS85 Benchmarks				
c432	3.7	0.1	15.0	0.7
c499	26.3	0.9	28.8	1.4
c880	45.1	1.3	44.9	1.8
c1355	18.5	0.6	27.9	1.4
c1908	1.8	0.1	2.6	0.1
c2670	56.8	1.7	53.1	2.3
c3540	37.1	1.1	41.7	2.0
c5315	33.9	0.8	36.2	1.3
DesignWare Benchmarks				
add64_rpl	8.2	0.3	−2.1	−0.1
add64_cla	40.7	1.3	33.0	1.7
add64_pparch	22.5	0.7	10.2	0.5
addsub64_rpl	37.6	1.3	31.4	1.5
addsub64_cla	−5.4	−0.2	29.5	1.6
addsub64_pparch	37.0	0.9	15.5	0.6
mult32_csa	45.4	1.5	52.8	2.7
mult32_pparch	34.3	1.1	31.1	1.5
ash64_mx2	41.8	1.0	1.3	0.1
ash64_str*	4.3	0.1	25.0	1.2
ash64_astr*	8.3	0.1	25.0	1.2
bsh64_mx2	50.0	1.8	32.3	1.8
bsh64_str	50.0	1.8	32.3	1.8
bsh64_astr	50.0	1.8	32.3	1.8
fp_addsub64_rtl	30.8	1.1	29.5	1.6
fp_addsub64_str	42.9	1.5	34.1	1.8
fp_mult32_rtl	34.9	1.3	27.6	1.5
crc64_str	47.1	1.6	37.4	1.9
Average				
ISCAS85	**27.9**	**0.8**	**31.3**	**1.4**
DesignWare	**32.3**	**1.1**	**26.6**	**1.4**
All Circuits	**30.9**	**1.0**	**28.0**	**1.4**

* The synthesized netlist was nearly identical for both architectures.
The synthesized netlist was nearly identical for all three architectures.

Figure 11 shows the impact of INC on total path delay, once again averaged over the full DesignWare benchmark set. The tested NBTI degradations range from less than 2% to over 40% increase in total path delay, representing the range of values reported in the literature [Paul et al. 2005; Wang et al. 2010]. INC reduces the total path delay by over 15% for the most severe NBTI degradation, a significant improvement in post-stress circuit speed.

6.5. Power and Area Overhead

Internal node controls are only added to a small percentage of the gates; 3% on average for our benchmarks. The percent increase in transistor count ranged from 0.4%–3.5% with an average of 1.6%, in contrast to the 8–12% overhead required for gate sizing [Vattikonda et al. 2006]. This small increase suggests that INC does not have significant power or area overhead.

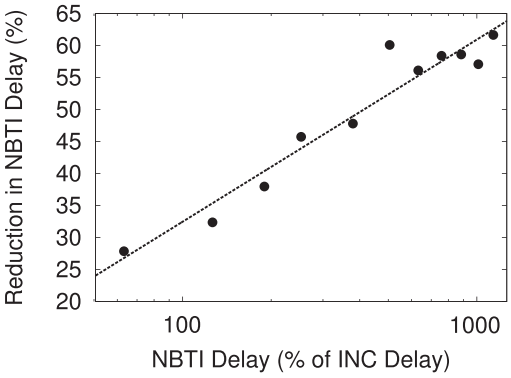


Fig. 10. Reduction in NBTI degradation scales with the ratio of NBTI delay and INC delay.

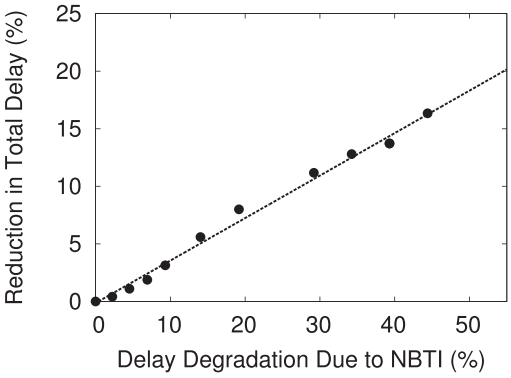


Fig. 11. Reduction in total circuit delay scales with NBTI degradation delay.

The average dynamic power consumption overhead will be less than 1.6%. There is an average 1.6% increase in transistor count, but only half of these, specifically the pull-up or pull-down transistors, present significant switching capacitance. The sleep transistor drain voltages will not change significantly due to stack effect, and thus will not appreciably increase dynamic power.

The leakage power consumption during active mode will also increase slightly. Each pull-up transistor provides an additional leakage path when the gate output is low. The converse is true for pull-down transistors. When this transistor is parallel to a transistor stack, it eliminates the leakage reduction due to stack effect. However, because on average only 3% of gates are modified with INC, this increase is negligible.

In sleep mode, the leakage power is also influenced by the chosen input vector. In this work we focus strictly on minimizing NBTI degradation using IVC and INC. These techniques have been previously studied for sleep-mode leakage power minimization and could be applied to simultaneously reduce NBTI-induced delay and leakage power. This could be achieved by first minimizing delay (leakage) and then minimizing leakage (delay) secondarily, with the optimal delay (leakage) as a constraint, or by minimizing some function of delay and leakage. The most desirable combination of reduction in leakage power and NBTI will depend upon a designer's preferences, and will fall somewhere between the proposed technique and Abdollahi, Fallah, and Pedram's leakage minimization technique [Abdollahi et al. 2004].

The area impact of the increased transistor count is quite small, less than 1.6% on average. However, the global (within the functional unit) sleep signal has the potential to increase routing congestion and thus area. Such congestion may be a problem for fine-grained power-gating techniques, where the signal must be routed to a large percentage of the gates. To determine whether INC has a similar problem, we compared the areas of placed-and-routed implementations of one of the six largest DesignWare benchmarks, both with and without INC. The results are shown in Table V. The average total increase in area is 2.4%, much of which is attributable to the increased transistor count, indicating that routing congestion is not a problem for INC. INC does not lead to routing congestion or require an additional routing layer because the sleep signal is routed to only a small fraction of gates, has loose timing and skew requirements (unlike a clock net), and has low drive strength requirements (unlike a power net). Consequently, it can be routed in local and intermediate metal layers (e.g., M2–M4) along with the rest of the logic signals with almost no area overhead.

With its low area and power consumption overhead and good degradation reduction capabilities, INC is an efficient method for NBTI mitigation.

Table V. Area Impact of INC considering Place-and-Route

Benchmark	Stressed Delay (ps)		Area (μm^2)		Area
	no INC	with INC	no INC	with INC	Increase (%)
mult32_csa	8869	8729	67026	68540	2.3
mult32_pparch	2717	2668	46556	46990	0.9
fp_mult32_rtl	4382	4303	47938	48658	1.5
fp_addsub64_rtl	5621	5537	29808	30225	1.4
fp_addsub64_str	5317	5217	34345	35023	2.0
crc64_str	3437	3374	14026	14875	6.1

7. MULTIPLE-VECTOR INC

In the preceding analysis, we have assumed that only a single input vector and set of INC placements are chosen for a given functional unit. Consequently, the same set of PMOS transistors are stressed whenever the unit is idle and the sleep mode activated. If several input vectors were chosen and alternately applied when the unit is idle, it is possible that the NBTI stress could be spread among transistors on multiple paths, resulting in a smaller increase in delay. This idea was proposed by Abella, Vera, and Gonzalez, but they did not perform analysis of the potential benefits of the technique and left development of input vector selection algorithms for future work [Abella et al. 2007]. In this section, we investigate the potential of multiple-vector INC for the ISCAS85 benchmarks. We find that in most cases, the use of multiple input vectors does not decrease the critical path delay. In the few cases where some benefit is seen, the decrease in delay is small enough that it is unlikely to justify the increase in area required to implement the multiple-vector strategy. We explain our findings for the sake of researchers and designers who might work on related problems in the future.

Internal node control requires the selection of both an input vector and a set of INC placements. A set of input vectors and sets of INC placements could be cycled through to potentially reduce the NBTI degradation. However, the implementation of such a technique does have drawbacks. In the case of multiple input vectors, all the vectors must be stored and MUXes or scan-chain logic are needed to route the chosen input vector to the functional unit. In the case of INC, additional gates must be modified to support INC and multiple sleep signals must be routed across the functional unit, further increasing area, delay, and leakage power. In both cases, a small state machine must be included to select the current input vector and sleep signal.

For the remainder of this discussion, the application of a particular input vector and INC assignment pair is referred to as a cycle, and thus the total number of pairs chosen is the number of cycles. For example, a 4-cycle solution refers to a rotation among four input vector and INC assignment pairs. We assume that with any reasonable strategy for cycling among the input vectors, in the long run all the vectors will be used for approximately the same amount of time. Thus, the NBTI delay for an individual gate is computed as the fraction of the cycles for which the gate is stressed times the delay if the gate were stressed for the entire 10 year period. Note that the cycles are not limited to distinct pairs; a particular pair can be used more frequently than another pair by assigning it to more cycles.

We modified our MILP formulation to support an arbitrary number of cycles and ran it for 2, 3, and 4 cycles on the ISCAS85 benchmarks. Figure 12 presents the results. As with the single-cycle problem instances presented earlier, the solver was not able to find true optimal solutions in a reasonable amount of time, and therefore we present the lower- and upper-bounded ranges for optimal. For each benchmark, the results for 1, 2, 3, and 4 cycles are shown side-by-side. The range in which the optimal solution lies for each problem instance is represented by a bar extending from the lower bound to the upper bound. The upper bound for any multicycle problem instance

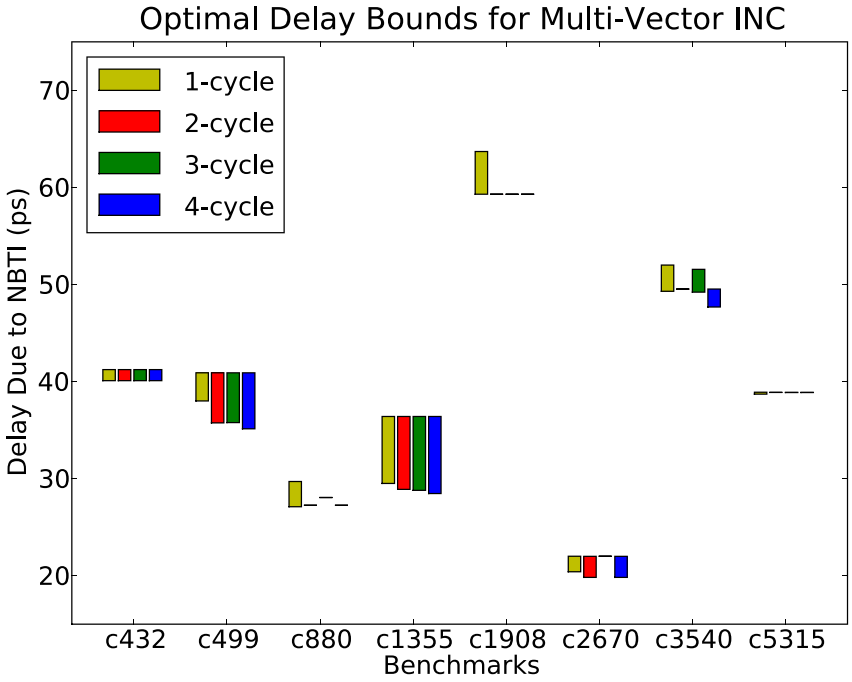


Fig. 12. Upper and lower bounds for multicycle INC for the ISCAS85 benchmarks.

can be no higher than the upper bound for the corresponding single-cycle problem instance because there is no requirement that the vectors be distinct. Thus, for instances where the upper bound proved by the MILP solver is greater than the corresponding single-cycle upper bound, we report the single-cycle upper bound in its place.

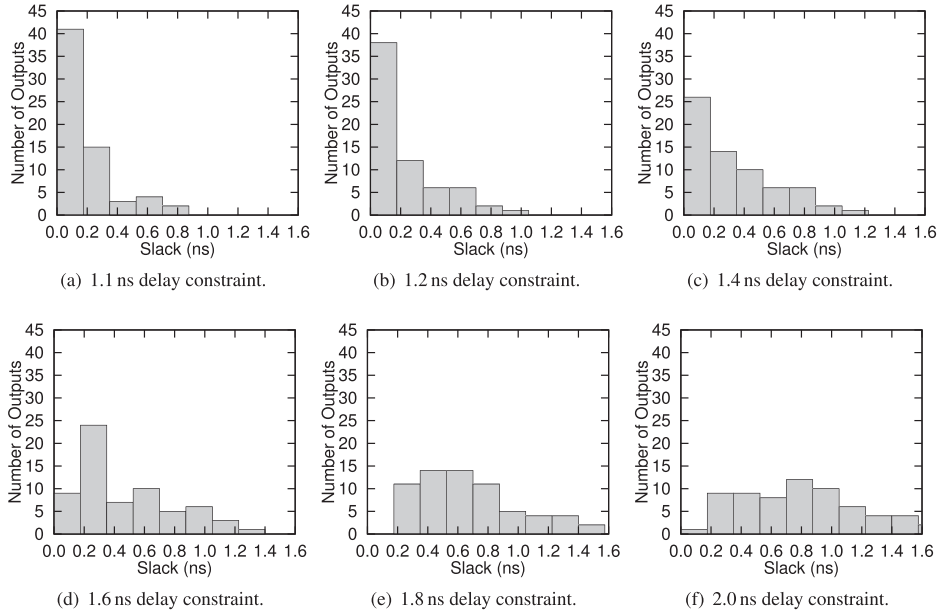
As can be seen from Figure 12, the use of multiple input vectors does not significantly decrease the circuit delay. Only benchmark c499 shows a significant decrease in the lower bounds for the multiple-vector cases. Even in this case, there may still not be any real improvement because the single-vector range still overlaps with multiple-vector ranges. It is likely that the potential for slightly improving the delay of c499 is due to its many parallel critical paths; the use of multiple input vectors may allow some degradation balancing between them. For the other benchmarks, which have few critical paths, the use of multiple vectors has little impact. Although the idea sounds promising, it appears to have little impact. Therefore, we recommend using the single-cycle technique described in the preceding sections.

8. SENSITIVITY ANALYSIS

As shown in the previous sections, internal node control substantially reduces NBTI degradation on average. However, there is great variability among the benchmarks, with reductions ranging from 1.8% to 56.8% for the ISCAS85 circuits and 4.3% to 50.0% for the DesignWare benchmarks. In this section, we analyze two sources of variance, tightness of the synthesis timing constraint and combinational logic path length.

8.1. Tightness of Synthesis Constraint

The tightness of the synthesis timing constraint can impact the effectiveness of INC because it affects the distribution of slack among paths. For example, Figure 13 shows the histograms of primary output slacks for the add64_rpl benchmark for timing



(a) 1.1 ns delay constraint. (b) 1.2 ns delay constraint. (c) 1.4 ns delay constraint.

(d) 1.6 ns delay constraint. (e) 1.8 ns delay constraint. (f) 2.0 ns delay constraint.

Fig. 13. Histograms of primary output slacks for benchmark add64_rpl with timing constraints ranging from 1.1 ns to 2.0 ns.

constraints ranging from 1.1 ns to 2.0 ns. For the tightest constraint, the large majority of the outputs have little slack, while for the loosest constraint, many of the outputs have slacks that are large fractions of the total delay. This observation suggests that the effectiveness of INC may be correlated with the tightness of the timing constraint because adding internal node control incurs some additional delay. More gates on noncritical paths can be modified with INC without increasing the overall delay. In this section we analyze this effect and show that our technique is useful even under tight timing constraints. We also highlight the importance of including path slacks when reporting experimental results for techniques that incur some additional path delay, for example, INC.

We synthesized each benchmark for ten different timing constraints in 0.1 ns intervals, starting from the tightest successful constraint and, using the heuristic, determined the percent reduction in NBTI-induced delay. Figure 14 shows scatter plots for the add64, addsub64, and mult32 benchmarks[3]. Plots are not shown for the ash64, bsh64, fp_addsub64, fp_mult32, and crc64 benchmarks because their output slack distributions were not dependent on the timing constraints and thus the delay reductions were not correlated with the constraints.

The results are mixed. Benchmarks add64_cla, addsub64_cla, and mult32_csa show positive correlation, with p-values of 0.242, 0.046, and 0.004, respectively. Benchmarks add64_pparch and addsub64_pparch show slight negative correlation, with p-values of 0.409 and 0.036, respectively. Finally, add64_rpl, addsub64_rpl, and mult32_pparch show little or no correlation, with p-values of 0.954, 0.682, and 0.448. The p-value indicates the probability that the same correlation would be observed in samples from a population with no correlation. Thus, using the traditional 95% significance level, a p-value less than 0.05 indicates that the correlation is statistically significant. As

[3]We show the scatter plots instead of reporting correlation coefficients, which can be significantly affected by outliers due to the small dataset size.

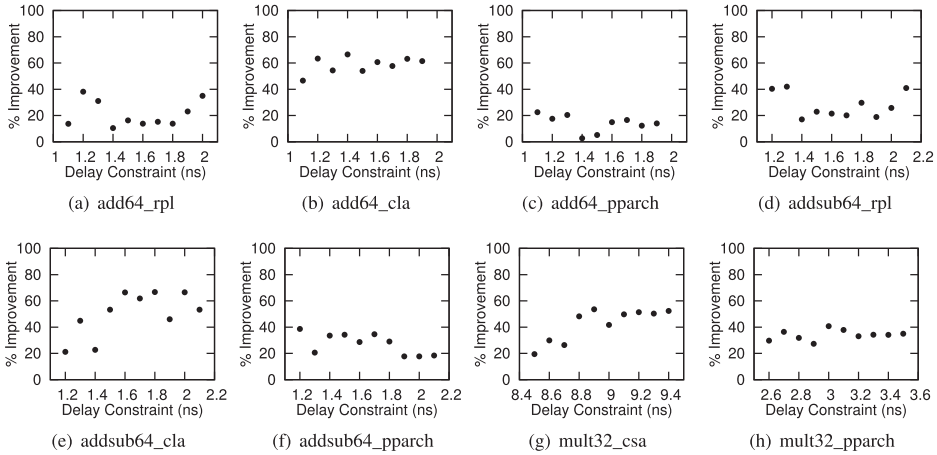


Fig. 14. Timing constraint correlation results for large benchmarks.

mentioned in the previous paragraph, outliers due to the discrete nature of the circuits and imperfections in the heuristic can increase the p-value.

For the addition-type circuits, the results are divided by architecture, with the carry-lookahead circuits showing strong correlation, the parallel-prefix circuits showing slight negative correlation, and the ripple-carry circuits showing no correlation. The carry-select multiplication circuit shows positive correlation, while the parallel-prefix shows little correlation. The negative correlation for the parallel-prefix adders could be inherent in the circuit structure or could be due to the heuristic. The circuits are too large for optimal solutions to be determined using the MILP solver, so it is not guaranteed that this trend exists for the optimal solutions.

From this, we conclude that the benefit of the proposed technique often depends on slack distributions. However, this dependence is highly dependent on design style. It is therefore important to include slack information when evaluating and discussing INC and other similar techniques.

8.2. Combinational Logic Path Length

The number of gates in the critical path may affect the variance in the results, especially for benchmark sets with relatively short critical paths, such as the ISCAS85 set. Due to the short critical paths of these circuits, the removal of NBTI from a single gate on the critical path has a large impact on the percentage improvement. In the best case, INC removes NBTI stress from all critical path gates. Thus, for circuits such as these with critical path lengths of 10 to 20 gates, each gate represents 5% to 10% of the total delay. Therefore, removing NBTI stress from one additional gate can add 5–10 percentage points to the delay improvement.

Although INC leads to only small NBTI delay reductions for some of the ISCAS85 benchmarks (e.g., a 3.2% decrease for c1908) and showed great variance across the set (e.g., up to a 55.0% improvement for c2670), we suspect that, as mentioned in the preceding paragraph, sets of larger circuits will show higher minimum reductions and lower variance. In this section, we provide experimental evidence indicating that INC may work better for larger circuits than is suggested by the results on the small ISCAS85 benchmarks.

We provide an intuitive argument and experimental results consistent with this intuition. This explanation is not a proof, but merely supporting evidence that our INC technique is more consistently effective on larger circuits. We treat the removal

Table VI. Analysis of Improvement Dependent on Path Length

Circuit Set	Average (%)	Standard Deviation (%)
Short	22.4	19.8
Long	42.0	9.3

of NBTI stress from each gate as independent events. In reality, there is some dependence between successive gates. However, this dependence is limited to a few levels of logic because of the filtering effect of this logic, and thus we can safely assume independence. Under this assumption, the variance in the number of gates with NBTI removed will be smaller for longer paths.

We tested this hypothesis by developing two sets of random circuits, one with short critical paths and one with long critical paths. To make the random circuits as realistic as possible, we employed the CGEN circuit parametrization and generation package [Kundarewich 2002; Kundarewich and Rose 2004]. CGEN was designed to help CAD researchers develop randomized circuits with structures approximating those in real designs to test their algorithms. The package includes two utilities. The first is used to characterize a circuit by extracting characteristics such as circuit shape, average fanin, and average fanout. The second utility takes a set of these parameters as input and generates a random circuit with similar characteristics.

We characterized ISCAS85 benchmark c880 and generated 50 random circuits for the short critical path circuit set. To create a long critical path circuit to characterize, we linked four instances of the c880 circuit serially, arbitrarily connecting the outputs of one stage to the inputs of the next. 50 random circuits were generated from the characterization of this circuit to create the long critical path circuit set. Each of these circuits was then solved for the IVC and INC cases as described in Section 6.2. Due to the size of the long circuits, the MILP solver was set to stop when the upper and lower bounds were within 0.5% of each other.

The results of this test are shown in Table VI. We find that the long circuits have a standard deviation in improvement of 9.3%, about half the 19.8% deviation of the short circuits. This result is in line with our hypothesis and provides evidence that our INC technique has better, more consistent results for longer paths.

9. CONCLUSION

We have proposed the use of internal node control to minimize the impact of static NBTI on circuits with frequently idle functional units. Placement of internal node controls, which allows the outputs of an INC-modified gate to be forced to a specific value during sleep mode, lead to a 30–60% decrease in static NBTI-induced delay and can decrease total path delay by 9.4% when NBTI degradation is severe.

The problem is $\mathcal{NP}$-complete, so we developed a linear-time heuristic that quickly produces good solutions. The problem is tractable for tree-structured circuits, so the heuristic first partitions a circuit into trees by removing edges. By ensuring that the gates on the critical path in the original circuit remain in the same partition, the optimal solutions to these partitions can be used to provide good solutions for the overall circuit. The heuristic solutions were within 0.24% of optimal post-wear delay on average and resulted in only a 1.6% increase in area. Using the optimal formulation, we see that using multiple input vectors does not lead to significant reduction in the degradation. A single static input vector, combined with internal node control, is sufficient.

REFERENCES

ABDOLLAHI, A., FALLAH, F., AND PEDRAM, M. 2004. Leakage current reduction in CMOS VLSI circuits by input vector control. *IEEE Trans. VLSI Syst. 12*, 2, 140–154.

ABELLA, J., VERA, X., AND GONZALEZ, A. 2007. Penelope: The NBTI-aware processor. In *Proceedings of the International Symposium on Microarchitecture*. 85–95.

ALAM, M. AND MAHAPATRA, S. 2005. A comprehensive model of PMOS NBTI performance degradation. *Microelectron. Reliab. 45,* 1, 71–81.

BHARDWAJ, S., WANG, W., VATTIKONDA, R., CAO, Y., AND VRUDHULA, S. 2006. Predictive modeling of the NBTI effect for reliable design. In *Proceedings of the Custom Integrated Circuits Conference*. 189–192.

BILD, D. R., BOK, G. E., AND DICK, R. P. 2009. Minimization of NBTI performance degradation using internal node control. In *Proceedings of the Design, Automation and Test in Europe Conference.* 148–153.

BRGLEZ, F. AND FUJIWARA, H. 1985. A neutral netlist of 10 combinational benchmark circuits and a target translator in FORTRAN. In *Proceedings of the International Symposium on Circuits and Systems*. 695–698.

CALIMERA, A., BENINI, L., MACII, A., MACII, E., AND PONCINO, M. 2009. Design of a flexible reactivation cell for safe power-mode transistion in power-gated circuits. *IEEE Trans. Circ. Syst. I 56*, 9, 1979–1993.

CAO, Y., SATO, T., SYLVESTER, D., ORSHANSKY, M., AND HU, C. 2000. New paradigm of predictive MOSFET and interconnect modeling for early circuit design. In *Proceedings of the Custom Integrated Circuits Conference.* 201-204.

CHENG, L., CHEN, D., AND WONG, M. D. 2008. A fast simultaneous input vector generation and gate replacement algorithm for leakage power reduction. *ACM Trans. Des. Autom. Electron. Syst. 13*, 2, 1–15.

GAREY, M. R. AND JOHNSON, D. S. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman & Company, New York.

HUARD, V., DENAIS, M., AND PARTHASARATHY, C. 2006. NBTI degradation: From physical mechanisms to modeling. *Microelectron. Reliab. 46*, 1, 1–23.

JOSEPH, R., BROOKS, D., AND MARTONOSI, M. 2003. Control techniques to eliminate voltage emergencies in high performance processors. In *Proceedings of the International Symposium on High-Performance Computer Architecture.* 79–90.

KANG, K., KIM, K., ISLAM, A. E., ALAM, M. A., AND ROY, K. 2007. Characterization and estimation of circuit reliability degradation under NBTI using on-line I_{DDQ} measurement. In *Proceedings of the Design Automation Conference*. 358-363.

KUMAR, S. V., KIM, C. H., AND SAPATNEKAR, S. S. 2006. An analytical model for negative bias temperature instability. In *Proceedings of the International Conference on Computer-Aided Design*. 493–496.

KUMAR, S. V., KIM, C. H., AND SAPATNEKAR, S. S. 2007. NBTI-aware synthesis of digital circuits. In *Proceedings of the Design Automation Conference*. 370–375.

KUMAR, S. V., KIM, C. H., AND SAPATNEKAR, S. S. 2011. Adaptive techniques for overcoming performance degradation due to aging in CMOS circuits. *IEEE Trans. VLSI Syst. 19*, 4, 603–614.

KUNDAREWICH, P. D. 2002. Synthetic circuit generation using clustering and iteration. M.S. thesis, University of Toronto.

KUNDAREWICH, P. D. AND ROSE, J. 2004. Synthetic circuit generation using clustering and iteration. *IEEE Trans. Comput.-Aid. Des. Integr. Circ. Syst. 23*, 6, 869–887.

LI, M. F., CHEN, G., SHEN, C., WANG, X. P., YU, H. Y., YEO, Y.-C., AND KWONG, D. L. 2004. Dynamic bias-temperature instability in ultrathin SiO_2 and HfO_2 metal-oxide-semiconductor field effect transistors and its impact on device lifetime. *Japan. J. Appl. Phys. 43*, 11B, 7807–7814.

PAUL, B. C., KANG, K., KUFLUOGLU, J., ALAM, M. A., AND ROY, K. 2005. Impact of NBTI on the temporal performance degradation of digital circuits. *IEEE Electron. Dev. Lett. 26*, 8, 560–562.

PTM. 2010. Predictive technology model. http://www.eas.asu.edu/~ptm.

RALPHS, T. AND GUZELSOY, M. 2005. The SYMPHONY callable library for mixed integer programming. In *Proceedings of the Conference of the INFORMS Computing Society.* Springer, 1–13.

ROY, K., MUKHOPADHYAY, S., AND MAHMOODI-MEIMAND, H. 2003. Leakage current mechanisms and leakage reduction techniques in deep-submicrometer CMOS circuits. *Proc. IEEE 91*, 2, 305–327.

SALUJA, K. K., VIJAYAKUMAR, S., SOOTKANEUNG, W., AND YANG, X. 2008. NBTI degradation: A problem or a scare? In *Proceedings of the International Conference on VLSI Design.* 137–142.

SCHRODER, D. K. 2007. Negative bias temperature instability: What do we understand? *Microelectron. Reliab. 47*, 6, 841–852.

SCHRODER, D. K. AND BABCOCK, J. A. 2003. Negative bias temperature instability: Road to cross in deep submicron silicon semiconductor manufacturing. *J. Appl. Phys. 94*, 1, 1–18.

STATHIS, J. AND ZAFAR, S. 2006. The negative bias temperature instability in MOS devices: A review. *Microelectron. Reliab. 46*, 2–4, 270–286.

TSAI, Y. F., DUARTE, D., VIJAYKRISHNAN, N., AND IRWIN, M. 2004. Characterization and modeling of run-time techniques for leakage power reduction. *IEEE Trans. VLSI Syst. 12*, 11, 1221–1232.

TSMC. 2006. Taiwan Semiconductor Manufacturing Company 65 nm standard cell library. http://www.synopsys.com/.

USAMI, K., SHIRAI, T., HASHIDA, T., MASUDA, H., TAKEDA, S., NAKATA, M., SEKI, N., AMANO, H., NAMIKI, M., IMAI, M., KONDO, M., AND NAKAMURA, H. 2009. Design and implementation of fine-grain power gating with ground bounce suppression. In *Proceedings of the International Conference on VLSI Design*. 381–386.

VATTIKONDA, R., WANG, W., AND CAO, Y. 2006. Modeling and minimization of PMOS NBTI effect for robust nanometer design. In *Proceedings of the Design Automation Conference*. 1047–1052.

WANG, W., WEI, Z., YANG, S., AND CAO, Y. 2007. An efficient method to identify critical gates under circuit aging. In *Proceedings of the International Conference on Computer-Aided Design*. 735–740.

WANG, W., YANG, S., BHARDWAJ, S., VRUDHULA, S., LIU, F., AND CAO, Y. 2010. The impact of NBTI effect on combinational circuit: Modeling, simulation and analysis. *IEEE Trans. VLSI Syst. 18,* 2, 173–183.

WANG, Y., LUO, H., HE, K., LUO, R., YANG, H., AND XIE, Y. 2007b. Temperature-aware NBTI modeling and the impact of input vector control on performance degradation. In *Proceedings of the Design, Automation and Test in Europe Conference*. 546–551.

WANG, Y., CHEN, X., WANG, W., BALAKRISHNAN, V., CAO, Y., XIE, Y., AND YANG, H. 2009a. On the efficacy of input vector control to mitigate NBTI effects and leakage power. In *Proceedings of the International Symposium on Quality of Electronic Design*. 19–26.

WANG, Y., CHEN, X., WANG, W., CAO, Y., XIE, Y., AND YANG, H. 2009b. Gate replacement techniques for simultaneous leakage and aging optimization. In *Proceedings of the Design, Automation and Test in Europe Conference.* 328–333.

ZAFAR, S., KIM, Y., NARAYANAN, V., JR., C. C., PARUCHURI, V., DORIS, B., STATHIS, J., CALLEGARI, A., AND CHUDZIK, M. 2006. A comparative study of NBTI and PBTI (charge trapping) in SiO_2/HfO_2 stacks with FUSI, TiN, Re gates. In *Proceedings of the Symposium on VLSI Technology*. 23–25.

Received February 2010; revised June 2011; accepted January 2012