Minimization of NBTI Performance Degradation Using Internal Node Control

David R. Bild
EECS Department
University of Michigan
Ann Arbor, MI 48109, USA
drbild@umich.edu

Gregory E. Bok
Nico Trading
311 S. Wacker Drive, Suite 900
Chicago, IL 60606, USA
greg.bok@gmail.com

Robert P. Dick
EECS Department
University of Michigan
Ann Arbor, MI 48109, USA
dickrp@eecs.umich.edu

***Abstract*—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 paper proposes internal node control, in which the inputs to individual gates are directly manipulated to prevent this static NBTI fatigue. We give a mixed integer linear program formulation for an optimal solution to this problem. The optimal placement of internal node control yields an average 26.7% reduction in NBTI-induced delay over a ten year period for the ISCAS85 benchmarks. We find that the problem is $\mathcal{NP}$-complete and present a linear-time heuristic that can be used to quickly find near-optimal solutions. The heuristic solutions are, on average, within 0.17% of optimal and all were within 0.60% of optimal.**

I. 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 [1].

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 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} = V_{dd}), the reaction reverses, with some of the hydrogen diffusing back towards the interface and bonding with the Si [1].

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 $\sim$10% increase in delay after 10 years of operation for the ISCAS85 benchmarks [2], [3].

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 unnecessary switching, 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.

This work was performed while the authors were with the Department of Electrical Engineering and Computer Science, Northwestern University, Evanston, IL 60208, USA. It 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.

In this paper, we propose and evaluate an internal node control technique to limit the effect of this static NBTI stress. Internal node controls can be inserted at the output of individual gates in order to force their outputs 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 non-critical path gates to circuit structure dependent values, such that a low value is propagated to the NBTI-sensitive transistor.

We formulate finding the optimal set of insertion points leading to the minimal degradation in circuit delay after some elapsed time as a mixed integer linear program. For the ISCAS85 benchmarks, we find that the optimal application of internal node control leads to an average 26.7% reduction in NBTI-induced delay relative to input vector control [4], a previously proposed technique. We have found that the problem is $\mathcal{NP}$-complete. Due to the time complexity of the optimal formulation, we present a linear-time heuristic to find good solutions in a reasonable amount of time. The heuristic is within 0.17% of optimal on average and within 0.60% of optimal for all benchmarks. The INC placement requires only a 1.6% increase in area.

II. Related Work

Several techniques have been proposed for dealing with the impacts of NBTI. One class of methods, which includes guard banding, gate sizing, V_{dd} tuning, and V_{th} tuning, has 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 attempt to minimize 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 [5]. 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 still meets the timing requirements. However, this technique imposes an 8%–12% area overhead and increases power consumption [6]. Similarly, in V_{dd} and V_{th} tuning, the voltage of the circuit is adjusted to increase the initial operating speed [6]. The problems with this technique are two-fold. 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 and therefore temperature of the circuit. Techniques that minimize the NBTI degradation are needed.

Power gating and clock gating methods have been used to reduce the power consumption of idle functional units [7]. In power gating, a *sleep transistor*, which can be turned off to prevent static and 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 usually 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-gated transistors do not experience NBTI degradation. However, the wake-up time for a power-gated functional unit is orders of magnitude longer than for a clock-gated, input vector-controlled unit [7]. Clock gating and IVC methods allow for more temporally fine-grained control.

Wang et al. investigated the use of IVC to reduce NBTI degradation [4]. In practice, this control can be implemented either by placing MUXes on the inputs or by using a scan-chain. Unfortunately, for many circuits, the input vector may only be able to control a few levels of the circuit's internal gates. Consequently, they observed only an average 3% improvement in delay for the ISCAS85 benchmarks. They predict that for future technologies, smaller gate sizings and higher temperatures may increase the benefit of this technique.

In contrast with IVC, internal node control (our proposed technique) permits much greater control of all levels of the circuit, allowing for greater reduction in the NBTI-induced delay.

III. 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 additional 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 a specific implementation of INC originally developed for static power consumption minimization. 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.

III.A. Internal Node Control Implementation

In order to force a node to a specific value, we borrow a technique from work by Abdollahi, Fallah, and Pedram on leakage minimization [8]. In this technique, a gate can be modified to allow its output to be forced either high or low, although not to both levels. 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. This is controlled by an active-low sleep signal that pulls the output high when enabled. In order to prevent a short through the gate, the pull-down network is then placed in series with an NMOS transistor. This transistor is responsible for the majority of the increase in gate delay. To force an output low, a similar modification is made. This is illustrated in Figure 1.

Unfortunately, the addition of this extra circuitry required for INC increases circuit delay. For a 65 nm Berkeley Predictive Technology Model, [9], [10], this technique results in an ∼12.5% increase in delay for a simple inverter. The absolute delay appears to be independent of gate type, so the percentage decreases for larger gates.

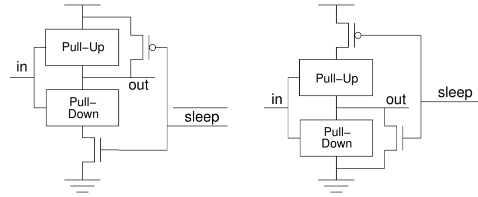

Figure 1. CMOS gates modified to include node control [8].

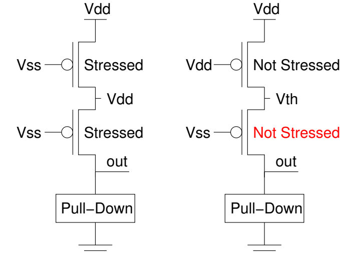

Figure 2. For a NOR gate, destressing the top PMOS destresses all subsequent transistors in the stack.

III.B. Potential of Internal Node Control for CMOS

For an inverting logic implementation technology such as 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 non-stressing (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 input value. 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 [11]. 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 2. 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. Because of the increase in delay associated with INC insertion, the performance gained (rather, retained) due to NBTI minimization is less than that lost due to INC insertion at every gate. It is not practical to cover every gate with INC. 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 which, 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. 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. The slack for each gate depends on the delays of all the prior and subsequent gates along its path. Therefore, it is dependent on

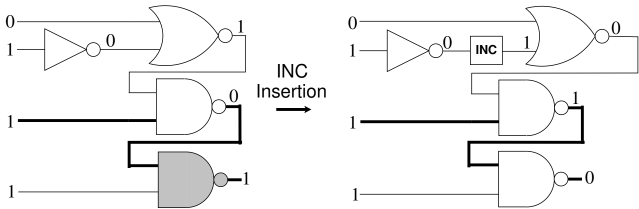

Figure 3. Off path INC insertion example. Gates affected by NBTI are shaded and critical path lines are darkened.

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. 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.

We now describe three example INC insertion scenarios. First, we will consider an NBTI-stressed gate on the critical path. If the degradation in delay is worse than the delay associated with adding INC to the previous gate, INC can be added to remove the stress. For the second example, consider an NBTI-stressed NOR gate on the critical path. Taking advantage of the aforementioned stack effect, INC can be added to a parent gate that is *not on the critical path*. The last example, illustrated in Figure 3, is more complicated. 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.

III.C. Problem Definition

We formulate the task of NBTI-induced delay reduction via INC as an optimization problem. The locations of internal node controls are selected in order to minimize the total combinational delay due to both INC overhead and static NBTI after some specified period of time. The input to the problem consists of a combinational circuit 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, e.g., 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 circuit and the INC-modified circuit.

The decision version of this problem, in which the minimization objective is replaced with a bound on the delay, is $\mathcal{NP}$-complete. We have shown this by reducing circuit-SAT to the INC insertion problem. The full proof is omitted due to space constraints, but can be found in Bild's master's thesis [12].

IV. Optimal Solution

In order to find the optimal solution, we describe the optimal mixed integer linear program (MILP) formulation.

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 v.

The instrinsic delay of a gate is $\tau_{n \in N}$. The increase in delay due to NBTI stress is $\rho_{n \in N}$, and the increase in delay due to the addition of node control on the gate output is $\phi_{n \in N}$.

The following variables are used. $\sigma_{n \in N}$ is a binary variable which is 1 if INC is added to gate n and 0 otherwise. $\kappa_{n \in N}$ is a binary variable representing the forced value of node n, if σ_n is 1. $0 \leq \psi_{v \in V} \leq 1$ is the value of node v. If σ_v is 1, then ψ_v is κ_v. Otherwise, it is determined by the inputs to the gate. For $v \in I$, ψ_v is explicitly constrained to be binary. $\alpha_{v \in V}$ is the earliest arrival time at node v.

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

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

The Boolean function of the gates, combined with the node control, is modeled by a set of constraints that force each output ψ_v to the proper value based on σ_v, κ_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 following are the constraints for an inverter. Table I shows the corresponding truth table. NAND and NOR gates are similarly determined.

$$\begin{array}{rlcr} \forall n \in N_I: & \sigma_n + \kappa_n - \psi_n & \leq & 1 \\ & \sigma_n - \kappa_n + \psi_n & \leq & 1 \\ & -\kappa_n + \psi_n + \psi_p - 1 & \leq & 0 \\ & -\sigma_n + \psi_n + \psi_p - 1 & \leq & 0 \\ & -\psi_p + \kappa_n - \psi_n & \leq & 0 \\ & -\psi_p - \sigma_n - \psi_n & \leq & -1 \end{array}$$

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 instrinsic delay τ_v of each gate is always included. The internal node control delay ϕ_v is only included if σ_v is 1. The NBTI delay ρ_v is included when, based on the inputs, the gate is stressed. For NOT and NAND gates, the following constraint enforces this relationship.

$$\forall n \in N_I \cup N_D, \forall p \in P_n: \ \alpha_n \geq \alpha_p + \tau_n + (1 - \psi_p)\rho_n + \sigma_n \phi_n$$

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 $0 \leq \gamma_{n \in N_R} \leq 1$ is 1 if NOR gate n is stressed and 0 otherwise. Thus, the following constraints implement the arrival time computation for NOR gates.

$$\begin{array}{rl} \forall n \in N_R, \forall p \in P_n: & 1 - \gamma_n \geq \psi_p \\ & 1 - \gamma_n \leq \sum_{r \in P_n} \psi_r \\ & \alpha_n \geq \alpha_p + \tau_n + \gamma_n \rho_n + \sigma_n \phi_n \end{array}$$

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

V. Optimal Experimental Results

We evaluated the proposed technique on the ISCAS85 combinational benchmarks [3]. The experimental setup and results are presented below. We also provide some analysis of the variance in results seen across the benchmark set.

TABLE I
TRUTH TABLE FOR AN INVERTER WITH INC

ψ_p	σ_n	κ_n	ψ_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

ψ_p is the input, σ_n is the INC selection, κ_n is the forced value, and ψ_n is the output.

TABLE II
PATH DELAYS (NS) FOR ISCAS85 CIRCUITS

Circuit	Baseline	Optimal IVC Only			Optimal IVC + INC			% Improvement		
		LB	UB	% Gap	LB	UB	% Gap	LB	Mid	UB
c432	1650.6	1697.9	1701.3	0.20	1690.7	1691.8	0.07	12.9	17.0	20.9
c499*	1588.7	1633.0	1641.2	0.50	1626.7	1629.6	0.18	7.8	18.6	27.7
c880	1884.3	1926.7	1927.9	0.06	1911.4	1914.0	0.14	30.1	34.1	37.9
c1355*	1505.1	1546.8	1555.5	0.56	1534.6	1541.5	0.45	12.7	28.4	41.5
c1908	2112.0	2175.0	2176.1	0.05	2171.3	2175.7	0.20	-1.1	3.2	7.5
c2670*	1607.1	1651.3	1657.1	0.35	1627.5	1629.1	0.10	50.3	55.0	59.2
c3540*	2546.4	2615.6	2624.7	0.35	2595.7	2598.4	0.11	24.8	31.3	37.1
c5315	2396.9	2448.2	2450.9	0.11	2435.6	2435.8	0.01	24.2	26.3	28.4

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

V.A. Experimental Setup

In order to gauge the performance of the method, it was tested on the ISCAS85 combinational benchmark suite for a 7 gate library {inv, nor2, nor3, nor4, nand2, nand3, nand4}. 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 Berkeley Predictive Technology Model [10], [9]. 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 [13]. Static NBTI delay was assumed to be about 10% of the initial gate delay after 10 years of stress [2]. The benchmarks were mapped to the library using Synopsis Design Compiler.

The MILP was solved using the open-source software SYMPHONY [14] for three different cases.

1) In order to obtain a baseline optimal circuit delay without internal node control or NBTI degradation, the solver was first run with the node control variables forced to 0 and the NBTI delays set to 0 ns.
2) The solver was then run with the actual NBTI delays to determine the delay using only input vector control.
3) Finally, the solver was run using internal node control.

The resulting problem instances are rather large for an MILP solver. Therefore, the solver was set to stop solving and report the results when the upper and lower bounds for the optimal delay were within 0.2% of each other.

V.B. Analysis of Results

The circuit delay results for each benchmark are shown in Table II. Even with the 0.2% stop gap for the solver, it did not finish for several of the benchmarks after several days. Thus, for the reported results, we manually terminated execution when the solver had been running for more than 24 hours. For all circuits we report both the lower and upper bounds as well as the percentage difference between them. The improvement is reported as the percent reduction in NBTI-induced delay between the IVC-only and the IVC+INC cases:

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

We report lower and upper bounds on this improvement as well. The lower bound is computed using the IVC lower bound and the IVC+INC upper bound. The upper bound is computed using the IVC upper bound and the IVC+INC lower bound. We also report a midpoint improvement which is calculated using the midpoints of the IVC bounds and the IVC+INC bounds.

The average midpoint improvement is 26.7% with a standard deviation of 15.1%. The average of the lower bounds is 20.2% and the average of the upper bounds is 32.5%. However, the improvement depends quite heavily on the benchmark. For benchmark c2670 over 50% of the degradation is prevented, while for benchmark c1908 the upper bound shows only single-digit improvement. The calculated lower bound on improvement for c1908 is negative. Obviously, the optimal worst case lower bound is 0%, if no INC placements are added. However, the best IVC+INC solution found by the solver (an upper bound on optimal) is worse than the IVC lower bound, leading to the negative improvement.

We do not discuss the area or power impacts of INC here because the MILP formulation only optimizes critical path delay, but not the total number of INC placements. Section VI describes a near-optimal heuristic that optimizes the delay while attempting to minimize the number of modified gates. The results in Section VII show that near-optimal delays can be achieved with little impact on area and power consumption.

One potential cause for the high variance of the improvements among the benchmarks is that, 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 will remove 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.

More formally, we can model the removal of NBTI stress from each critical path gate as an independent Bernoulli trial. In reality, there is some dependence between successive gates. However, we can safely assume independence because the actual dependence is limited to a few levels of logic. By the law of large numbers, as the number of gates on the critical path increases, one can expect the observed improvements to be closer to an expected or average improvement.

VI. 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 would therefore be useful. In this section, we describe a linear-time algorithm for input vector selection and internal node control placement. Our technique draws from work on leakage power minimization by Cheng, Chen, and Wong [15].

VI.A. 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 using dynamic programming. It first partitions a given

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 dynamic programming
6: **end for**
7: update dangling input values
8: **if** solution is the same as previous **then**
9: break {Check for convergence}
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

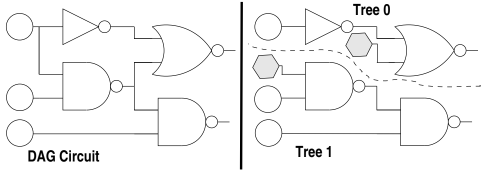

Figure 4. A circuit partitioned into rooted trees, with the *dangling inputs* shaded.

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 4. 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 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 pre-set 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 convergent solutions.

VI.B. 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 [15]. For these problems, the cost function (e.g., total leakage power, 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 optimal dynamic programming 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 original critical path in a single partition. Of course, for circuits with reconvergent critical paths, this will not always be possible. Our partitioning algorithm maintains 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.

Algorithm 2 Dynamic Programming Algorithm

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

As mentioned in the previous paragraph, the choice of the initial values for the dangling inputs is also important. We choose these initial values by applying the optimal dynamic programming algorithm to the unmodified directed acyclic circuit. Because the circuit is not tree-structured, in the backward pass phase of the algorithm, conflicts will occur. 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.

VI.C. Dynamic Programming

The optimal dynamic programming 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 a topological order (line 1). For each gate, each possible input combination is examined (line 2). The output value is computed and, based on 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). 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

TABLE III
HEURISTIC RESULTS FOR ISCAS85 CIRCUITS

Circuit	Optimal Delay			Heuristic	% Worse	% Worse	Time	Total	INC	Original	INC	%
	LB	Mid	UB	Delay	than Mid	than UB	(s)	Gates	Gates	Trans.	Trans.	Increase
c432	1690.7	1691.3	1691.8	1700.8	0.56	0.53	1.3	159	11	636	22	3.5
c499	1626.7	1628.1	1629.6	1628.0	-0.01	-0.10	6.7	526	18	1836	36	2.0
c880	1911.4	1912.7	1914.0	1914.0	0.07	0.00	2.7	336	11	1306	22	1.7
c1355	1534.6	1538.1	1541.5	1547.4	0.60	0.38	5.2	480	12	1840	24	1.3
c1908	2171.3	2173.5	2175.7	2175.7	0.10	0.00	3.2	363	8	1322	16	1.2
c2670	1627.5	1628.3	1629.1	1629.1	0.05	0.00	11.9	592	13	2302	26	1.1
c3540	2595.7	2597.0	2598.4	2595.7	-0.05	-0.10	27.0	725	29	2966	58	2.0
c5315	2435.6	2435.7	2435.8	2435.8	0.00	0.00	41.0	1452	12	5650	24	0.4

combination for its child. The corresponding input combination is selected for the gate (line 11).

VI.D. Runtime

The heuristic requires time linear in the number of gates. Partitioning is performed with a single topological traversal. The dynamic programming 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, although the overall algorithm iterates multiple times, empirical results show that it converges rapidly and the number of iterations can be limited to a small number (15 in our reported results).

VII. HEURISTIC EXPERIMENTAL RESULTS

We implemented the proposed heuristic in Python and tested it on the same benchmarks and with the same 7 gate library as in Section V. Note that the runtime is independent of the number of cells in the library. All tests were done on a 2.5 GHz AMD Athlon XP computer with 2 GB of memory. The runtime for each benchmark is shown in the "Time" column of Table III.

The results are shown in Table III. For each benchmark, the lower, midpoint, and upper bounds from the optimal solutions are given. The heuristic solutions are first compared with the midpoint, because this was used in Section V to compute the average improvement. In all cases, the heuristic solutions are quite good, with an average 0.17% degradation from the optimal midpoint. For benchmarks c499 and c3540, the heuristic solution is actually better than the optimal midpoints, although obviously not better than the lower bound. Benchmark c1355 shows the worst degradation with a 0.60% increase in the critical path delay.

We also compare the heuristic solutions to the optimal upper bound because it is entirely possible that for many of these benchmarks, the upper bound actually is optimal. In all cases except two, the heuristic solutions are at least as good as the optimal upper bound. Benchmarks c432 and c1355 show the only degradation, with 0.53% and 0.38% increases in critical path delay, respectively.

The last five columns of Table III show the impact on circuit area for the solutions produced by the heuristic. The number of gates modified with INC and the percent increase in transistor count needed to implement INC are shown. On average, INC imposes only a 1.6% area overheard, in contrast with the 8–12% overheard required for gate sizing [6]. Note that the estimated area increase is based solely on the increase in transistor count; we did not perform place and route. For designs that are interconnect-dominated, the area impact may be even smaller. Benchmark c432 is the worst with a 3.5% increase. The impact on power consumption will also be small. The average 1.6% increase in transistor count should translate into a similarly-small increase in leakage and switching power consumptions.

VIII. CONCLUSION

We have described internal node control, a technique for minimizing the impact of static NBTI on circuits with standby-equipped functional units. The optimal placement of INC yields an average 26.7% improvement in NBTI-induced delay degradation for the ISCAS85 benchmarks. A linear-time heuristic was shown to give solutions within 0.17% of optimality on average. The area and power consumption overheads are negligible, with a 1.6% increase in transistor count.

REFERENCES

[1] M. Alam and S. Mahapatra, "A comprehensive model of PMOS NBTI degradation," *Microelectronics Reliability*, vol. 45, no. 1, pp. 71–81, Jan. 2005.

[2] B. C. Paul, K. Kang, J. Kufluoglu, M. A. Alam, and K. Roy, "Impact of NBTI on the temporal performance degradation of digital circuits," *IEEE Electron Device Ltrs.*, vol. 26, no. 8, pp. 560–562, Aug. 2005.

[3] F. Brglez and H. Fujiwara, "A neutral netlist of 10 combinational benchmark circuits and a target translator in FORTRAN," in *Prof. Int. Symp. Circuits and Systems*, May 1985, pp. 695–698.

[4] Y. Wang, H. Luo, K. He, R. Luo, H. Yang, and Y. Xie, "Temperature-aware NBTI modeling and the impact of input vector control on performance degradation," in *Proc. Design, Automation & Test in Europe Conf.*, Apr. 2007, pp. 546–551.

[5] J. Abella, X. Vera, and A. Gonzalez, "Penelope: The NBTI-aware processor," in *Proc. Int. Symp. Microarchitecture*, Dec. 2007, pp. 85–95.

[6] R. Vattikonda, W. Wang, and Y. Cao, "Modeling and minimization of PMOS NBTI effect for robust nanometer design," in *Proc. Design Automation Conf.*, Jul. 2006, pp. 1047–1052.

[7] Y. F. Tsai, D. Duarte, N. Vijaykrishnan, and M. Irwin, "Characterization and modeling of run-time techniques for leakage power reduction," *IEEE Trans. VLSI Systems*, vol. 12, no. 11, pp. 1221–1232, Nov. 2004.

[8] A. Abdollahi, F. Fallah, and M. Pedram, "Leakage current reduction in CMOS VLSI circuits by input vector control," *IEEE Trans. VLSI Systems*, vol. 12, no. 2, pp. 140–154, Feb. 2004.

[9] "Predictive technology model," http://www.eas.asu.edu/~ptm.

[10] Y. Cao, T. Sato, D. Sylvester, M. Orshansky, and C. Hu, "New paradigm of predictive MOSFET and interconnect modeling for early circuit design," in *Proc. Custom Integrated Circuits Conf.*, Sep. 2000, pp. 201–204.

[11] S. V. Kumar, C. H. Kim, and S. S. Sapatnekar, "NBTI-aware synthesis of digital circuits," in *Proc. Design Automation Conf.*, Jun. 2007, pp. 370–375.

[12] D. R. Bild, "Static NBTI reduction using internal node control," Master's thesis, Northwestern University, Evanston, IL, Dec. 2008.

[13] Taiwan Semiconductor Manufacturing Company, "TSMC 65 nm standard cell library," 2006, http://www.synopsys.com/.

[14] T. Ralphs and M. Guzelsoy, "The SYMPHONY callable library for mixed integer programming," in *Proc. Conf. INFORMS Computing Society*. Springer, Jan. 2005.

[15] L. Cheng, D. Chen, and M. D. Wong, "A fast simultaneous input vector generation and gate replacement algorithm for leakage power reduction," *ACM Trans. Design Automation in Electronic Systems*, vol. 13, no. 2, Apr. 2008.