Controlling NBTI Degradation during Static Burn-in Testing

Ashutosh Chakraborty
ECE Department
The University of Texas at Austin
Austin, TX 78703, USA
ashutosh@cerc.utexas.edu

David Z. Pan
ECE Department
The University of Texas at Austin
Austin, TX 78703, USA
dpan@ece.utexas.edu

ABSTRACT

Negative Bias Temperature Instability (NBTI) has emerged as the dominant PMOS device failure mechanism in the nanometer VLSI era. The extent of NBTI degradation of a PMOS device increases dramatically at elevated operating temperature and supply voltage. Unfortunately, both these conditions are concurrently experienced by a VLSI chip during the process of burn-in testing. Our analysis shows that even with a short burn-in duration of 10 hours, the degradation accumulated can be as much as 60% of the NBTI degradation experienced over 10 years of use at nominal conditions. *Static* burn-in testing in particular is observed to cause most NBTI degradation due to absence of relaxation phase unlike the case for *dynamic* burn-in testing. The delay of benchmark circuits is observed to increase by over 10% due to static burn-in testing. We propose the first technique to reduce the NBTI degradation during static burn-in test by finding the minimum NBTI induced delay degradation vector (MDDV) based on timing criticality and threshold voltage change (ΔV_{TH}) sensitivity of the cells. Further, only a subset of the input pins need to be controlled for NBTI reduction, thus our technique allows other objectives (such as leakage reduction) to be considered simultaneously. Experimental results show that the NBTI induced critical path delay degradation can be reduced by more than 50% using our proposed technique.

1. INTRODUCTION

Negative Bias Temperature Instability (NBTI) has emerged as the primary PMOS failure mechanism since the advanced sub-65nm VLSI technology [1]. Several technology trends have caused NBTI degradation to occur at even higher rates in newer technology nodes. These trends include reduction in gate oxide thickness which increases the electric field across the gate oxide, and increased use of nitrides to reduce Boron penetration [2]. Under these circumstances, NBTI dictates the lifetime of the devices instead of other reliability issues such as hot carrier injection, time dependent dielectric breakdown etc. As feature sizes shrink further, NBTI effects will worsen exponentially due to higher operating temperatures caused by increased power density. Due to NBTI, the threshold voltage (V_{TH}) of each PMOS device under negative gate-to-source bias, increases slowly with time. This phenomenon has a strong temperature dependence and worsens at elevated temperature. This gradual increase in V_{TH} of a PMOS device leads to reduced drain current causing larger delay of the cell leading to violation of timing constraints. Over a period of 10 years, the V_{TH} of the PMOS device can increase by up to 50mV [3] causing timing violation and functional failures.

NBTI effect has been known to the design community for several decades [2] but only in the recent technology nodes it has proved to be the reliability limiter of a VLSI product. A similar but opposite effect exists for NMOS devices called positive bias temperature instability (PBTI) in which the V_{TH} of NMOS gate increases when the device is positively biased. In current technology node, the degradation due to NBTI is more severe than PBTI. The reason of V_{TH} increase due to NBTI is believed to be the accumulation of positively charged interface traps at the Si–SiO$_2$ boundary when the PMOS device is negatively biased (i.e. $V_{GS} \leq 0$). When the bias conditioned on the PMOS device is reversed, *some* of the interface traps can recombine leading to *partial* recovery of the V_{TH} degradation. In a typical VLSI product, each device goes over a sequence of positive and negative bias conditions leading to alternate stress and recovery of V_{TH} degradation.

The NBTI effect increases dramatically at higher operating temperature and supply voltage. In the whole manufacturing flow, burn-in testing is a unique stage when the VLSI product is simultaneously subjected to both these detrimental conditions. Thus, burn-in testing has potential to cause significant NBTI degradation (i.e. V_{TH} increase) of the devices even if the duration of burn-in testing is short (of the orders of tens of hours). In this paper, we tackle the problem of quantifying and reducing the impact of burn-in testing on the NBTI degradation of a circuit. We identify that *static* burn-in testing in particular causes severe NBTI degradation of the chip since due to the static nature of the test (i.e. no switching), there are no recovery phases that can reduce the degradation. To the best of our knowledge, there is no prior work to reduce this detrimental effect. We propose finding a partially specified input vector to be used during burn-in testing with the aim of re-distributing the NBTI degradation based on timing criticality as well as NBTI sensitivity of different cells in the circuit. Using the proposed method, the degradation in the critical path delay of benchmark circuits could be reduced by as much as 53%.

The rest of the paper is organized as follows. In the next section, we will present the background of NBTI effect, burn-in testing, and NBTI degradation during burn-in testing. Section 3 describes our proposed methodology for reducing NBTI degradation during static burn-in testing by partial input vector assignment. Experimental setup and benchmarks are presented in Section 4. Section 5 describes the results achieved by our technique in terms of reduction in NBTI induced degradation. Section 6 concludes our paper with directions for future research.

2. BACKGROUND

2.1 Negative Bias Temperature Instability

The increase in magnitude of the threshold voltage (V_{TH}) of a PMOS device under negative bias is due to the generation of interface traps at the oxide and semiconductor interface. These interface traps are the result of breaking of weak Si–H bonds which are formed due to crystal mismatch at the channel-gate interface [4]. Figure 1 from [5] illustrates the complete process of interface trap generation. The breaking of Si–H bonds at the Si–SiO$_2$ interface creates Si$^+$ (interface traps) and H atom. The presence of Si$^+$ at the surface requires larger gate voltage for channel inversion. This is the reason V$_{TH}$ of a PMOS device increases due to NBTI. During

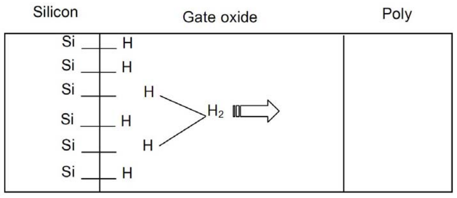

Figure 1: Illustration of NBTI phenomenon. Breaking of Si–H bonds creates Si$^+$ interstitial causing V$_{TH}$ increase. Picture from [2].

the normal operation of a product, every PMOS device in a VLSI chip undergoes both negative and positive gate-to-source bias condition. During the time a PMOS is under negative gate-to-source bias, we refer the PMOS device as being in *stress* phase. On the other hand, while the PMOS device is under positive gate-to-source bias, we refer to it as being in *recovery* phase. For example, in a chain of inverters that is being fed with logic low at the input of the chain, the PMOS gates in the inverters at odd position from the input are under *stress* phase whereas PMOS gates at even positions are undergoing *recovery* phase. For a larger circuit, depending on the statistics of the input vector applied, each PMOS device experiences stress and recovery phases of different durations.

NBTI and its effects have been considerably researched in recent years. These work include lifetime prediction of a chip considering AC stress, NBTI aware timing analysis, reliability of memories due to PMOS Vth degradation etc. The power-law dependence of NBTI on time [6] has been reported by most of the existing research. A predictive NBTI model based on physical understanding and published experimental data was presented in [7]. [8] demonstrated analytical iterative equations for computation of NBTI impact for arbitrary waveforms. Numerical methods to solve the exact reaction/diffusion (R/D) physics based model of NBTI has been proposed in [9]. The work in [1] proposed a tight upper-bound of NBTI degradation for long term computation. In [10], the impact of NBTI on circuit timing has been tackled by making logic synthesis aware of NBTI. More recently, the impact of NBTI on clock tree has been researched during design time [11] and runtime [12] which aim at reducing the mismatch between the NBTI degradation or delay change to improve integrity of the clock tree.

In a recent work [1], authors have derived a tight analytical upper bound on the V$_{TH}$ degradation of a PMOS device, ΔV_{TH} as

$$\Delta V_{TH} \quad = \quad \left((V - V_{TH})^2 \times t \times e^{\frac{-2E_n}{kT}} \right)^{\frac{1}{6}} \qquad (1)$$

where V is the supply voltage, t is the duration of operation of the device, E$_n$ is the activation energy for NBTI effect ($\tilde{0}$.50 eV [2] for H$_2$ diffusion), k is Boltzmann constant with value of 8.617×10^{-5} eVK^{-1}, and T is the operating temperature of the device in Kelvin. From Eqn 1, we can identify that NBTI induced V$_{TH}$ degradation is exponentially dependent on temperature of operation. Increasing supply voltage can also substantially increase V$_{TH}$ degradation.

2.2 Burn-in Testing

Due to increasing defect density and variation during manufacturing process, testing of the fabricated products has becomes very difficult for sub-65nm VLSI technology node. The failure rate of a typical product is known to be bath-tub shaped [13] and is characterized by three phases. The first phase is the infant mortality period where the products with inherent defects or manufacturing aberration quickly fail. This is followed by relatively flat second phase with low failure rate which depicts the normal operation of the product. The third phase occurs near the end of expected lifetime of a product where the wear-out mechanisms cause the product to malfunction or fail altogether. Of particular concerns are the products that belong to the first type where devices appear to operate correctly after fabrication but will fail shortly after being deployed in the field. VLSI testing deals with this problem by performing (aging) accelerated tests. By operating the product under voltage, thermal, or mechanical stress, the devices in it can be artificially aged much quicker than the real duration of operation of testing. Burn-in testing is one such test to screen and eliminate marginal devices and is performed at significantly increased supply voltage and temperature. The aim of burn-in test is to expedite the onset of infant mortality by accelerating the aging of product. The acceleration in aging during burn-in testing is the product of acceleration due to higher supply voltage and the acceleration due to higher operating temperature [14]. A functional test done after burn-in testing can catch most of the failures due to ion contamination and resistive interconnects. Recent work in the area of burn-in testing has focused on reduction of burn-in time, and burn-in recipe modification [14] by choosing appropriate values of elevated supply voltage or temperature.

There are two types of burn-in tests: a) static burn-in and b) dynamic burn-in. Static burn-in is performed by operating the product to be tested at elevated voltage and temperature but with a fixed vector applied to its input thus no switching occurs in the circuit. On the other hand, during dynamic burn-in, a series of vectors are applied to the circuit causing switching of various circuit nodes. In a typical burn-in recipe, both static and dynamic burn-in testing is performed. Due to its constant bias nature, static burn-in causes more degradation than dynamic burn-in testing. Through HSpice simulations, we observed 4X larger V$_{TH}$ degradation during static burn-in compared to dynamic burn-in, in line with the trend in [2].

2.3 NBTI During Static Burn-in Testing

One may wrongly assume that since static burn-in testing is performed for a short duration, its impact on V$_{TH}$ increase may be insignificant. Due to the exponential (polynomial) dependence of NBTI on temperature (voltage), we expect significant V$_{TH}$ increase during burn-in test. To quantify, we simulated the V$_{TH}$ increase of two 45nm PMOS devices under regular vs burn-in testing conditions: the first device was simulated at supply voltage of 0.9V, temperature of 65^{o}C operating for 10 years. The second device was simulated at a supply voltage of 1.25V, temperature of 125^{o}C operating for *only* 10 hours. These set of conditions correspond to aging acceleration factor of approximately 1000. The resultant increase in V$_{TH}$ of the second device was as much as 61% of that of first device. This experiment shows that in just 10 hours of static burn-in, the device accumulates more than 60% of the degradation it would accumulate in 10 years operating in field. Therefore, the NBTI degradation of a chip due to burn-in testing cannot be neglected even if the burn-in duration is short. To the best of our knowledge, there has been no prior work to estimate and reduce the NBTI degradation caused by static or dynamic burn-in testing. Our primary contributions in this work are:

1. We quantify the threshold voltage (V$_{TH}$) degradation due to NBTI during static burn-in testing and show that severe degradation can result.

2. To reduce the NBTI impact of static burn-in we propose use of minimum NBTI induced delay degrading vector (MDDV)

during static burn-in.

3. We formulate the problem of finding MDDV as a zero-one linear program (ZOLP) that automatically considers timing criticality and NBTI sensitivity of cells in the circuit.

3. REDUCING NBTI IN STATIC BURN-IN

During static burn-in testing, a static (i.e. non changing) vector is applied at circuit input which biases the circuit in a fixed configuration under high operating temperature and voltage. Due to the presence of sequential elements (flops or latches), it is not enough to only determine the logic values at each primary input of the circuit since this will bias only those gates which lie on paths from primary inputs to first set of sequential gates. Therefore, the input vector assignment should assign logic values to the output of the sequential elements apart from the circuit primary inputs. For testability reasons, all the sequential elements in the circuit are chained together using scan chain methodology through which the required input vector assignment, even deep inside the circuit, can be performed. We assume that the design independent environmental parameters of burn-in testing recipe are at their optimal value depending on analysis of failure rate decrease and thus cannot be modified. These parameters include the burn-in duration, voltage and temperature whose values are determined after failure rate analysis as well as aging acceleration requirements.

3.1 Minimum Delay Degrading Vector (MDDV)

The choice of the input vector applied during the static burn-in test presents itself as a strong design dependent optimization knob. This is due to different delay sensitivity of various gates to V_{TH} degradation and logic value propagation due to functionality implemented by the gates. A good input vector during burn-in testing can distribute the V_{TH} degradation among the gates such that the degradation of some figure of merit of the design is minimized. Since NBTI mainly impacts performance (and signal integrity indirectly), we define the critical path delay of the circuit as the metric to optimize. We call this vector as Minimum Delay Degrading Vector (MDDV) and formally define the problem as:

DEFINITION 1. *For a given burn-in recipe (duration, voltage, and temperature), gate level netlist of the product, and timing characterization of standard cells, find the (possibly partial) input vector which when applied during static burn-in, minimizes the increase in critical path delay of the product.*

The idea of using MDDV is similar to using minimum leakage power vector (MLV) which is defined as the input vector which causes the least leakage power in the circuit during stand-by periods. Some of the representative work discussing exact and heuristic solutions for MLV are [15, 16]. Finding MLV is an NP complete problem: exact solutions are typically SAT based and the heuristics proposed are mainly randomized search based. However, the problem of determining MDDV is very different from determining minimum leakage vector in many aspects:

- *Cost Function*: The optimization objective of MLV problem is simply the algebraic sum of all leakage power values of individual gates. However, the optimization objective of MDDV is to reduce delay through the circuit which depends on relative delay among competing paths thus more difficult to optimize.

- *Solution Type*: In MLV, the complete (i.e. all bits of) the input vector must be computed for obtaining the least leakage. However, due to the presence of non timing critical cells, it is possible to partially determine some bits of MDDV vector which feed timing critical cells. This allows surgical analysis of the circuit using high complexity formulations such as zero-one linear program (ZOLP).

- *Scope*: Since the input vector is only partially determined in MDDV, other optimization objectives can be co-optimized to assign logic values to the rest of the input pins.

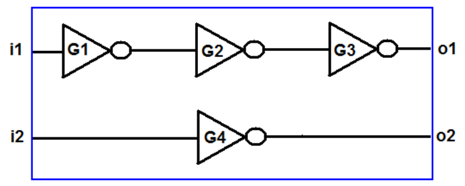

Figure 2: Illustration demonstrating partial determination of input vector for the purpose of NBTI induced delay degradation reduction. Non timing critical inputs are ignored if their delay can never increase enough to become critical path.

EXAMPLE 1. *Figure 2 shows a toy example circuit depicting the concept of partial vector determination. The circuit has two primary inputs (i1 and i2) and two primary outputs (o1 and o2). Assume that each inverter gate has a nominal delay of 1ps. The critical delay of the circuit is 3ps due to the top path. Now, consider that due to NBTI degradation, each inverter's delay increases by 10%*[1]*. Thus, the degraded delay of the top path is 3.3ps whereas that of the bottom path is 1.1ps. Since the degraded delay of the bottom path is lesser than the original critical path delay of the circuit, we infer that NBTI degradation can never make that path critical, hence we ignore this path and focus on only the top path. For minimum increase in the delay of the top path, the input i1 must be set to logic 1. This finishes the determination of* MDDV *by forcing only partial bits of input vector to desired value (i.e. i1 = 1). The other input (i.e. i2) are don't cares for NBTI degradation analysis and can be set to any value depending on any simultaneous optimization objective such as leakage power.*

3.2 MDDV Determination Algorithm

Algorithm 1 shows the overall flow of our algorithm for finding MDDV. Each step of the flow is described later in this section. The first step in our flow is to determine all probable cells, the logic value of whose pins need to be controlled for reducing NBTI induced delay degradation. These cells comprises of the fanin cone of any timing endpoint (i.e. primary outputs or inputs of sequential cells) that do not meet the timing constraint after NBTI degradation. For each net driven by the output pins of such cells, we assign a binary variable which represents its logic value during static burn-in testing. Depending on the logic implemented by each cell, logic equations among the variables associated with the input and output nets of a cell can be determined. Similarly, timing constraints among the input and output nets can be obtained by assigning arrival time variables which allow critical path optimization with linear number of constraints [17]. With these relations, the problem of determining the minimum NBTI degradation vector MDDV can be cast as a zero-one linear program (ZOLP).

[1]In reality delay increase is not a fixed number for different gates, the assumption of 10% is only for illustration. For the sake of simplicity, in this example rise and fall delay's different dependence is ignored.

Algorithm 1 Compute `MDDV` vector for static burn-in

```
1: Determine post static burn-in timing critical cells
2: Assign binary variable to net connecting critical cells
3: Determine logic implication and timing constraints
4: Cast MDDV finding as ZOLP problem
```

Determining post burn-in critical cells: Algorithm 2 describes method we used to determine post burn-in critical cells. With the understanding that burn-in will degrade the circuit timing to some extent, the algorithm takes an *optional* parameter (default value = 0) called acceptable degradation (AD) which denotes the degradation that is acceptable during burn-in testing (Line 2). The circuit's required timing that is used during timing analysis is incremented by the acceptable delay degradation (Line 3). The goal of providing a realistic number for acceptable degradation is to reduce the computation efforts by removing the paths whose increased delay will be less than AD even if the PMOS devices in it undergoes NBTI degradation. With the updated value of required time, static timing analysis (STA) is performed to identify timing failing paths (Line 5). During this STA run, we load the degraded timing library which has delay numbers corresponding to behavior of a cell after a particular threshold voltage increase has occurred. From the results of STA, all the endpoints which do not meet the timing are identified. For each of these endpoints, we find the cells lying in their fanin cone. All such cells are marked as critical cells.

Algorithm 2 Find post-burn critical cells

```
1: {// Modify required timing}
2: Input: Acceptable Degradation AD%
3: Increase Required Arrival time by AD%
4: {// Run STA with new library}
5: Load timing information from $V_{TH}$ degraded library
6: Perform STA and find timing violating endpoints (EP)
7: SEL = $\Phi$
8: for all ep in EP do
9:     SEL = SEL $\cup$ cells in fanin cone of ep
10: Return SEL
```

Input-Output logic relation for each cell type: The output net of each critical cell identified in Algorithm 2 is associated with a binary variable which represents its logic value during static burn-in testing. We refer to these nets as critical nets. Depending on functionality of the cell, there exists a logic relation between these variables. Let the input pins of cell x be denoted by IN(x) and the output pin by OU(x). Further, for such a cell, let the relationship between the logic values of these pins be given by

$$OU(x) = F(IN(x)) \tag{2}$$

For example if the binary variable associated with the input and output of an inverter gate is x_a and x_z, then they are related by relation $x_z = 1 - x_a$. Consider an OR gate whose two input pins are associated with binary variables x_a and x_b and the output pin is associated with binary variable x_z. These variables are connected through the relationship in binary form as $x_z = x_a + x_b - x_a x_b$. Similarly, for other cells, the relation binding the input and output binary variables are established.

Generating delay constraints: We associate a real valued variable for each of the critical nets. The value of this variable represents the arrival time of the signal *after NBTI degradation of the circuit*. Similar to method shown in [17] [18], we generate linear constraints relating the signal arrival time at input and output of the cells to minimize the critical path delay of the circuit. Basically each of these constraints ties the signal arrival time of the output of any cell to be larger than or equal to the sum of signal arrival time at any of its input and delay of the timing arc from the corresponding input to the output. Let the delay of the timing arc from input pin *i* to the output pin *o* be D^O_{ij} originally (i.e. without NBTI degradation of the devices in the cell) and D^N_{ij} after NBTI degradation. Further, let the logic value of net driving pin *i* be x_i, then the timing arc delay can be written as

$$D_{ij} = x_i \times D^O_{ij} + (1 - x_i) \times D^N_{ij} \tag{3}$$

that is, when the cell is driven by logic high (i.e. $x_i = 1$) during static burn-in testing and there is no NBTI degradation in its PMOS devices, D_{ij} equals D^O_{ij}. In case of a gate driven by logic low (thus undergoing NBTI degradation), D_{ij} equals D^N_{ij}. Though NBTI impacts only the PMOS devices, it may seem enough to consider only the rise time of a cell. However, the timing arcs from an input pin to output pin can be either negative unate (e.g. NAND gate), positive unate (e.g. AND gate), or even non-unate (e.g. XOR gate). Therefore, both the rise and fall delay of every timing arc needs to be written in the form of Eqn 3.

ZOLP Formulation to determine MDDV: Once the boolean logic relation among the binary variables associated with the input and output pins of each cell is ascertained, we can formulate the problem of finding `MDDV` as a zero-one linear program (ZOLP). Let *EP* be the set of timing endpoints (i.e. D pins of flops or primary outputs) that are driven by the critical cells identified. These are the endpoints whose timing slack is negative and therefore their arrival time needs to be optimized simultaneously. The arrival time of the signal at a node x is denoted as AT(x). Further, let all the cells selected to be critical be represented by *SEL*. The input pins of cell x will be denoted by IN(x) and without loss of generality, we assume that all cells have single output pin denoted by OU(x). The formulation for finding `MDDV` is:

$$\begin{array}{rll}
Minimize : MAX & & \\
Subject\ To : & & \\
AT(i) \leq MAX & \forall i \in \mathbf{EP} & \\
AT(i) + D_{ij} \leq AT(j) & \forall i \in IN(x)\ \ \forall j \in OU(x) & \forall x \in \mathbf{SEL} \\
F(IN(i), OU(i)) = 1 & \forall i \in \mathbf{SEL} & \\
x_i \in \{0,1\} & \forall i \in \mathbf{SEL} &
\end{array}$$

By requiring all endpoint delays to be less than the dummy variable *MAX* in the first set of constraints, and minimizing *MAX*, we can effectively obtain the logic value assignment for nets connected to the critical cells so as to obtain the fastest possible circuit. The second set of constraints establishes the actual delay relationship between input and outputs of a cell where the timing arc delay from input pin *i* to output pin *j* is given by Equation 3. The third set of constraints are for relationship between logic value for input and output pins for each critical cell based on Eqn 2. Note that depending on these equation, there will be multiplication of variables in the above formulation. Thus it may seem that the constraints in the above formulation is not linear but quadratic in nature. However, by virtue of the variables being binary, a multiplication of any number of variables can be decomposed in terms of linear equations [12]. For example, consider the term $x_A x_B$ which is multiplication of the binary values x_A and x_B. We can replace the multiplication the variable $x_A x_B$ by a new binary variable M_{AB} and add the following constraints to the above formulation:

$$\begin{array}{rcl}
x_A + x_B & \leq & 1 + M_{AB} \\
(1 - x_A) + (1 - x_B) & \leq & 2 - 2 \times M_{AB}
\end{array}$$

It can be verified that the only condition under which the value of binary variable M_{AB} is 1, is if both x_A and x_B are 1. The above transformation can be repeatedly applied thus allowing conversion of arbitrary multiplication of binary variables to linear equations. Once the above ZOLP is solved, we can obtain self consistent logic values on each pin of the critical cells identified. The solution to the above ZOLP also determines the logic values at the inputs that drive the timing failing paths. In case of sequential circuits, the inputs of the circuit comprises of primary inputs as well as the output pin of sequential elements.

ZOLP Speedup: Depending on the histogram of slack at the timing endpoints of a circuit, there could be a large number of timing violating endpoints for some circuits. Solving one large ZOLP for such circuits can be infeasible due to runtime issues. For such cases, the critical cells can be naturally partitioned into independent components and each such component can be solved independently. The failing cells can be partitioned into independent sets by traversing the fanin-cone of each timing failing endpoint going backwards towards inputs. If during the traversal from an endpoint ep_a, we visit a cell already visited by traversal of endpoint ep_b, the search proceeds after merging the two sets of cells from the traversal of the two endpoints. In our experiments, we *always* identify independent components and solve them individually.

4. EXPERIMENTAL SETUP

Our complete flow was implemented in C++ language with the gate level simulations being done using HSpice [19] tool. All experiments were run on an Intel dual core 2.66 GHz workstation with 4GB main memory running Debian OS. To enable post burn-in static timing analysis (STA), we created a new liberty (.lib) file containing the timing information for the standard cells in degraded state. Static timing analysis to identify timing failing cells was done by Primetime [19] tool. We used opensource Nangate [20] 45nm library for all experiments with nominal PMOS V_{TH} of 350mV. The number of standard cells in this library is 134. The magnitude of V_{TH} degradation in case a PMOS gate undergoes NBTI degradation during the entire duration of static burn-in (i.e. is driven by logic low) was found to be 42mV based on our NBTI simulation. For our implementation we characterized the degraded delay of only those cells which actually appear in the circuit netlist. To solve the ZOLP formulation for finding MDDV, we used the MILP solver Gurobi [21].

Table 1: Benchmarks used in this work. Number of cells, nets, pins, flops, primary inputs and outputs are shown for each benchmark.

Name	# Cells	# Nets	# Pins	# Flops	# Inp	# Out
LDPC	44K	48K	4100	2048	2051	2049
wbConmax	27K	29K	2546	818	1130	1416
B19	87K	99K	77	6118	47	30
ethernet	42K	43K	210	10544	95	115
DES	56K	59K	298	8808	234	64

Table 1 shows the benchmarks used in our work from the Opensource repository [22]. *LDPC* is the decoder unit of Low-Density Parity Checker. *wbConmax* is a wishbone interconnect matrix IP. Benchmark *ethernet* is an ethernet wishbone interface. *DES* is a data encryption standard core. The second column onwards in the table contains the number of cells, nets, number of IO pins, flops, number of primary inputs, and the number of primary outputs in the circuit. For these benchmarks, we solved the MDDV problem to obtain the best partial input vector. The delay degradation when MDDV is applied during static burn-in testing will be compared to that when applying either all 0s or all 1s as the input vector. To substantiate the claim that only some of the bits of the input vector needs to be specified, we also compute the fraction of the input pins whose value is determined by MDDV. Note that the logic value of both circuit *primary inputs* as well as flip flop's output are under purview of input vector determination. In our experiments, the acceptable delay degradation due to NBTI is set to 2% of the critical delay, thus during STA using degraded timing numbers, any path whose delay is more than 102% of critical path delay before burn-in testing was marked as failing timing requirement.

5. RESULTS

The results of computation of MDDV is shown in Table 2. For each benchmark, column *OrigDelay* shows the critical path delay (critical delay here onwards) before starting the burn-in testing as reported by STA using the original (i.e. non degraded) library. The degradation due to three input vectors are shown next: a) input vector comprising of all bits at logic low (*All 0* in the Table), b) input vector comprising of all bits at logic high (*All 1* in the Table), and c) the MDDV vector. For each of these three settings of input vectors, the delay achieved after static burn-in testing appears in column *AchvDel* and its degradation w.r.t. the original critical delay is shown in column Δ*Delay*. The column *CPU Time* in the table shows the CPU runtime (in seconds) for preparing and solving the ZOLP formulation for a given benchmark. This runtime includes the time for data reading, STA, critical cell finding, and ZOLP generation and solving. For the case of *All 0* and *All 1*, since we do not need to run any solver, their CPU runtime is zero. The last column *#BinVar* shows the number of binary variables in the generated ZOLP. The time required to characterize the library for a given V_{TH} is not reported since it is a one time effort and the resultant library can be used for many designs.

From Table 2, we observe that the MDDV vector can be computed in reasonable time for all the above circuits. The delay degradation of the circuit after burn-in testing when using the input vector comprising of all bits at logic low (high) is observed to be 7% (7.2%). This corroborates our thesis that burn-in can significantly degrade a circuit due to the NBTI effect. For example, if speed binning is performed on a 2.66 GHz processor after the burn-in conditions simulated in our work, it will run at 2.33 GHz potentially ending up performing at speeds lower than its capability. Intuitively, since NBTI occurs in PMOS devices at negative bias, we expected an all 0 input vector to give much higher degradation than all 1 input vector. However, the degradation due to these two vectors is nearly the same. On further analysis we found that the reason for this observation is that because of logic functions implemented by different gates, an all 0 input vector though stresses the gates close to the launching flops/primary inputs, may induce a logic high value at some other internal node thus reducing its degradation. On application of the NBTI induced minimum delay degradation vector (MDDV), the circuit delay degradation is reduced to only 3.3%. Therefore, using the MDDV vector during static burn-in testing can reduce the delay degradation by more than 2X.

Figure 3 shows the fraction of pins whose value is forced to a particular logic value by the solution of MDDV determination compared to the total number of assignable pins in the circuit. Note that each primary input as well as flop output pin are assignable. From the numbers in this figure, the first observation we make is that the fraction of pins assigned can vary widely among different circuits. Due to the highly convoluted interconnect structure of LDPC decoder, more than 50% input pins get assigned as part of solving MDDV. On the other hand, due to the unique structure of *ethernet* (notice that 25% of cells are sequential elements from Table 2), different cells are not highly connected. The number of input pins that MDDV enforces was found to be just 1.1% for this

Table 2: Comparison of delay degradation using MDDV compared to all 0 and all 1 vector.

		All 0		All One		MDDV			
Benchmark	OrigDel (ns)	AchvDel (ns)	ΔDelay	AchvDel (ns)	ΔDelay	AchvDel (ns)	ΔDelay	CPU Time	# BinVars
LDPC	1.581	1.694	7.14%	1.695	7.26%	1.627	2.90%	635s	4616
wbConmax	1.308	1.411	7.87%	1.398	6.88%	1.347	2.98%	386s	1688
B19	4.646	4.950	6.54%	4.947	7.13%	4.814	3.61%	1024s	6159
ethernet	2.091	2.247	7.46%	2.238	7.04%	2.178	4.16%	158s	380
DES	1.709	1.812	6.02%	1.832	7.25%	1.758	2.86%	327s	1381
Avg:			7.01%		7.10%		3.30%		

benchmark. Overall, on an average only 25% of the input pins are forced by MDDV and the logic value for the rest 75% of the pins can be chosen to optimize a secondary objective during burn-in testing such as leakage power thus reducing power dissipation as well as heating of the chip due to it.

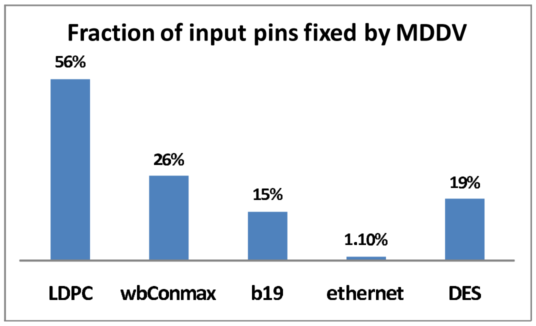

Figure 3: Fraction of input pins whose logic value is fixed by MDDV vector. Rest of the pins are available for optimizing secondary objective.

6. CONCLUSIONS

In this paper, we quantified the negative bias temperature instability (NBTI) induced threshold voltage (V_{TH}) increase during the burn-in accelerated testing process. The problem of reducing the impact on critical path delay increase due to NBTI was targeted by identifying a minimum NBTI delay degradation vector (MDDV). A good MDDV distributes the V_{TH} degradation in an intelligent way considering the timing criticality and NBTI induced V_{TH} increase sensitivity of different cells. Computation of MDDV was formulated as a zero-one linear program (ZOLP). Thanks to non-timing critical paths in the circuit, the MDDV vector requires only a subset of input pins to be at a pre-defined logic values which allows optimization of other objective such as leakage power using other input pins. On a set of benchmarks, our technique reduces the critical path delay degradation by more than 2X while requiring only 25% of the input pins to a fixed logic value. Our future work would be in the area of burn-in recipe (duration, voltage, temperature) optimization for NBTI reduction as well as reducing NBTI degradation during dynamic burn-in testing.

7. REFERENCES

[1] S. Bhardwaj *et al.*, “Predictive modeling of the nbti effect for reliable design,” *IEEE Custom Integrated Circuits*, pp. 189–192, Sept. 2006.
[2] M. Alam *et al.*, “A comprehensive model for pmos nbti degradation: Recent progress,” *Microelectronics Reliability*, vol. 47, no. 6, pp. 853 – 862, 2007. Modelling the Negative Bias Temperature Instability.
[3] Y. Wang *et al.*, “Temperature-aware nbti modeling and the impact of input vector control on performance degradation,” in *DATE 2007*, pp. 546–551.
[4] S. Kumar *et al.*, “Impact of nbti on sram read stability and design for reliability,” *ISQED ’06.*, pp. 6 pp.–, March 2006.
[5] M. A. Alam *et al.*, “A comprehensive model of pmos nbti degradation,” *Microelectronic Reliability*, pp. 71–81, Sept. 2005.
[6] S. Mahapatra *et al.*, “On the generation and recovery of interface traps in mosfets subjected to nbti, fn, and hci stress,” *Electron Devices, IEEE Transactions on*, vol. 53, pp. 1583–1592, July 2006.
[7] R. Vattikonda *et al.*, “Modeling and minimization of pmos nbti effect for robust nanometer design,” in *DAC*, (New York, NY, USA), pp. 1047–1052, ACM, 2006.
[8] S. V. Kumar *et al.*, “An analytical model for negative bias temperature instability,” in *ICCAD ’06*, (New York, NY, USA), pp. 493–496, ACM, 2006.
[9] R. Vattikonda *et al.*, “A new simulation method for nbti analysis in spice environment,” *ISQED*, pp. 41–46, March 2007.
[10] S. Kumar *et al.*, “Nbti-aware synthesis of digital circuits,” in *DAC. 44th ACM/IEEE*, pp. 370–375, June 2007.
[11] A. Chakraborty *et al.*, “Analysis and optimization of nbti induced clock skew in gated clock trees.,” in *DATE*, pp. 296–299, IEEE, 2009.
[12] A. Chakraborty and D. Z. Pan, “Skew management of nbti impacted gated clock trees,” in *ISPD 2010*, pp. 127–133.
[13] G.-A. Klutke, P. C. Kiessler, and M. A. Wortman, “A critical look at the bathtub curve,” *IEEE Transactions on Reliability*, vol. 52, no. 1, pp. 125–129, 2003.
[14] O. Semenov *et al.*, “Burn-in temperature projections for deep sub-micron technologies,” *Test Conference, International*, vol. 0, p. 95, 2003.
[15] K. Gulati *et al.*, “A probabilistic method to determine the minimum leakage vector for combinational designs in the presence of random pvt variations,” *Integr. VLSI J.*, vol. 41, no. 3, pp. 399–412, 2008.
[16] F. Gao and J. P. Hayes, “Exact and heuristic approaches to input vector control for leakage power reduction,” in *IEEE/ACM ICCAD 2004*, pp. 527–532.
[17] M. A. B. Jackson and E. S. Kuh, “Performance-driven placement of cell based ic’s,” in *DAC 1989*, pp. 370–375.
[18] C. J. Alpert, D. P. Mehta, and S. S. Sapatnekar, *Handbook of Algorithms for Physical Design Automation*. Boston, MA, USA: Auerbach Publications, 2008.
[19] Synopsys, *HSpice/Primetime User Guide*. 2007.
[20] “Corporate website.” http://www.nangate.com.
[21] Gurobi Optimization, “Solver gurobi,” 2010.
[22] http://opencores.org.