Gate Replacement Techniques for Simultaneous Leakage and Aging Optimization

Yu Wang[1], Xiaoming Chen[1], Wenping Wang[2], Yu Cao[2], Yuan Xie[3], Huazhong Yang[1]
[1]Dept. of E.E., TNList, Tsinghua Univ., Beijing, China
[2]Dept. of E.E., Arizona State Univ., USA, [3]Dept. of CSE, Pennsylvania State Univ., USA
[1] Email: yu-wang@mail.tsinghua.edu.cn

***Abstract*—[1]As technology scales, the aging effect caused by Negative Bias Temperature Instability (NBTI) has become a major reliability concern for circuit designers. On the other hand, reducing leakage power remains to be one of the design goals. Because both NBTI-induced circuit degradation and standby leakage power have a strong dependency on the input vectors, Input Vector Control (IVC) technique may be adopted to mitigate leakage and NBTI. However, IVC technique is in-effective for larger circuits. Therefore, in this paper, we propose two fast gate replacement algorithms together with optimal input vector selection to simultaneously mitigate leakage power and NBTI induced circuit degradation: Direct Gate Replacement (DGR) algorithm and Divide and Conquer Based Gate Replacement (DCBGR) algorithm. Our experimental results on 20 benchmark circuits at 65nm technology node reveal that: 1) Both DGR and DCBGR algorithms outperform pure IVC about on average 20% for three different object functions: leakage power reduction only, NBTI mitigation only, and leakage/NBTI co-optimization. 2) The DCBGR algorithm leads to better optimization results and save on average 100X runtime compared with the DGR algorithm.**

I. Introduction

As technology scales, Negative Bias Temperature Instability (NBTI) is emerging as one of the major reliability degradation mechanisms [1]. NBTI occurs when PMOS transistors are negatively biased (i.e., $V_{gs} = -V_{dd}$) at elevated temperature, causing a shift in threshold voltages. Over a long period of time, such V_{th} shifts can potentially cause a significant increase in the delay of PMOS devices [2], and result in about 10-20% degradation in circuit speed, thus may lead to a functional failure [3]. The impact of NBTI on circuit performance has become a key issue with technology scaling [4]. Consequently, it is important to model, analyze, and mitigate the impact of the NBTI effect on the circuit performance.

Based on the various circuit level NBTI degradation analysis models [5]–[7], previous works estimated the NBTI induced life-time degradation with the assumption that the circuits operate all the time. However, in practical not every application requires the underlying hardware to operate at the highest performance level all the time. Modules in which the computation is burst are often idle. There are periods during which the PMOS transistors are under static stress condition. Many PMOS transistors affected by NBTI can be found in both combinational and storage blocks when the gate inputs are set to "0" during the standby time, leading to a larger degradation. Consequently, it is important to accurately estimate the NBTI-induced degradation at the standby time in order to safely guard-band the circuit performance, and to find design techniques to mitigate such degradation.

Input Vector Control (IVC) is a well-studied technique for leakage power reduction [8] at the standby time. Since NBTI also depends on the input patterns of PMOS devices, IVC can be used to mitigate the NBTI effect during the standby mode. Fig. 1 shows the relation between the circuit leakage power and the circuit delay degradation caused by NBTI under different input vectors. We can see that given the required constraint for both leakage and delay degradation (the shadow region in Fig. 1), a set of input vectors can be preselected and applied to the entire circuit at the standby mode, such that both the total leakage power and delay degradation are minimized. In this example, less than 1% of sampled input patterns provides the minimum of both circuit degradation and the leakage.

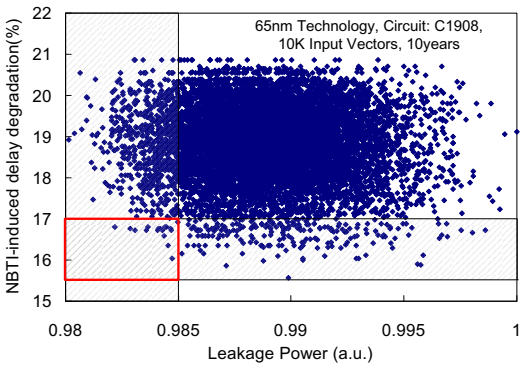

Fig. 1. Leakage power versus delay degradation for different input vectors.

Wang *et al.* [9] proposed a method to select the best input vectors from the minimum leakage vector set. However, the best input vectors for minimum leakage power may not be the best input vectors to minimize NBTI-induced circuit degradation and they didn't consider the difference of NBTI effects during active and standby time, the results claimed only 3% circuit degradation saving at the 90nm technology node. Jaume *et al.* [10] used different input vectors to change the zero-probability of internal PMOS transistors, so that the PMOS transistors' degradation was evenly distributed. The effect of this technique on an adder is evaluated, however, detailed research for random logic is needed.

Although pure IVC techniques have been evaluated for mitigating NBTI, they are not very effective when the circuit becomes larger. How to efficiently find the optimal results for leakage and NBTI induced circuit degradation remains a problem. There is no literature about simultaneously NBTI and leakage mitigation through **Internal Node Control (INC)** [11]–[13] which is proved to be more effective to reduce leakage power than pure IVC.

In this paper, we propose two fast gate replacement algorithms which simultaneously mitigate the leakage power and NBTI induced circuit degradation. The contributions of this paper can be summarized in the following aspects:

1) The gate replacement techniques are for the first time used for NBTI mitigation. Based on the basic gate replacement technique for

[1]This work was supported by National Natural Science Foundation of China (No. 60870001, No.90207002) and TNList Cross-discipline Foundation. Yu Cao's work was partially supported by GSRA/SRC. Yuan Xie's work was supported in part by grants from NSF 0643902, 0702617, and a SRC grant.

978-3-9810801-5-5/DATE09 © 2009 EDAA

NBTI and leakage reduction, we first propose a Direct Gate Replacement (DGR) algorithm and then propose a Divide and Conquer Based Gate Replacement (DCBGR) algorithm to further improve the NBTI/leakage reduction achievement and the optimization speed.

2) The complexity of DGR algorithm is $O(n^2)$ in the worst case and $O(n)$ on average; while the complexity of DCBGR is $O(n)$. Therefore, our algorithms will serve well when circuit scale becomes larger.

3) Our experimental results show that: for larger circuits, IVC technique is less effective, while INC through gate replacement technique is more effective for both NBTI and leakage mitigation.

4) Although the gate replacement technique is compatible with standard cell design flow, the area penalty remains a problem. Our DCBGR results for leakage only and NBTI only show that the area penalty for leakage reduction is larger: on average 13.26%, while the area penalty for NBTI mitigation is smaller: on average 3.53%.

II. Preliminaries

A. Degradation Model under NBTI Effect

Depending on the bias condition of PMOS transistor, NBTI has two phases: stress phase and recovery phase. In the stress phase ($V_{gs} = 0$), the holes in the channel weaken the Si-H bonds, which results in the generation of the positive interface charges and hydrogen species, correspondingly, threshold voltage (V_{th}) of the PMOS transistors increases. During the recovery phase ($V_g = V_{DD}$), the interface traps can be annealed by the hydrogen species and thus, V_{th} degradation (ΔV_{th}) is partially recovered. If a PMOS device is always under stress condition, it is referred as *static* NBTI. Otherwise, both stress and recovery exist during active circuit operation, it is described as *dynamic* NBTI.

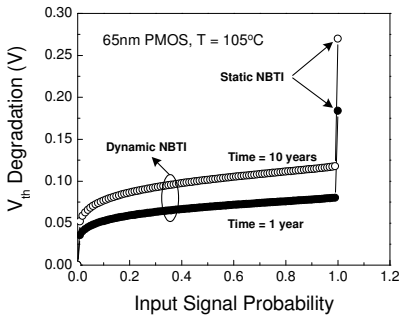

Fig. 2. Static and dynamic NBTI degradation for different input signal probabilities.

Based on the reaction-diffusion mechanism, real time NBTI model is developed in [14], [15]. For dynamic NBTI, there is a sudden change at the beginning of the recovery phase, which has a significant impact on the estimation of NBTI degradation. A long-term prediction model is derived for both static and dynamic NBTI in [15]. Fig. 2 shows ΔV_{th} prediction by using the proposed model. The big difference between the static and dynamic NBTI, has also been observed in silicon data [16], [17]. Therefore, the simple static analysis may cause an extremely pessimistic estimation of NBTI-induced degradation and consequently, results in over-margining in design stage. On the contrary, only dynamic NBTI model for the total lifetime without considering the static NBTI effect during the standby time may lead to an underestimation of NBTI-induced performance degradation. In this paper, we use dynamic NBTI model in the active time and static NBTI model in the standby time.

The delay difference due to ΔV_{th} is given by [9], [18]:

$$\Delta d(v) = \alpha \Delta V_{th} / (V_{gs} - V_{th}) \times d(v) \quad (1)$$

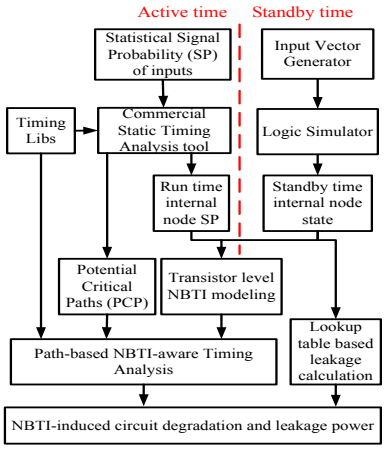

Fig. 3. The NBTI and leakage co-simulation flow.

where $d(v)$ is the original delay of gate v which can be extracted from the commercial STA tools. There could be several ΔV_{th} of different PMOS's in one gate. In such cases, we just select the largest one to calculate the gate delay degradation, which is the worst case delay degradation.

B. NBTI/leakage co-simulation flow

Fig.3 shows our NBTI/leakage co-simulation flow. For a given circuit, commercial static timing analysis tool is firstly used to generate the Potential Critical Paths (PCPs) using standard timing libraries. When the circuit is in the active mode, statistical information for input Signal Probability (SP) is used to generate the internal node SP. When circuit is in the standby mode, logic simulator is used to generate the voltage level of each internal node. The active time internal node SP and the standby time internal node states are used to estimate the NBTI-induced V_{th} degradation through transistor level NBTI modeling. The leakage power is estimated based on the input vector aware leakage lookup tables. Based on the V_{th} degradation estimation and the original timing libraries, a fast path-based NBTI-aware timing analysis is performed. We modify the input vector generation module to implement our gate replacement algorithms.

III. Gate Replacement (GR) Technique

The gate replacement technique is to replace a gate $G(\overrightarrow{x})$ by another library gate $G(\overrightarrow{x}, sleep)$ [12], where $\overrightarrow{x}$ is the input vector of gate G, $sleep$ is the sleep signal of the circuit, such that:

1) $G(\overrightarrow{x}, 0) = G(\overrightarrow{x})$, when the circuit is active ($sleep = 0$);

2) $G(\overrightarrow{x}, 1)$ has smaller leakage power or can serve as an INC point to mitigate NBTI effect when the circuit is standby ($sleep = 1$).

1) Gate replacement for NBTI: The NBTI effect on a PMOS transistor depends on the stress condition: V_{gs} and stress time, which are both related to the input state of a gate. Consequently, all 1's will be the best input pattern with the smallest NBTI-induced degradation for all gate types. Fig. 4 is an example that shows how to mitigate NBTI-induced degradation by gate replacement. The NAND2 gate $G2$'s delay will be larger if $G1$'s output is 0 at the circuit standby time. Through gate replacement technique, we replace $G1$ by an NAND3 gate so that the output is changed to 1. Hence the NBTI effect on $G2$ is mitigated during the standby time.

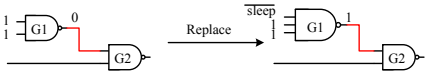

Fig. 4. A gate replacement example for NBTI mitigation.

2) Gate replacement for leakage: We call a gate at its WLS (worst leakage state) [12] when its input vector leads to the largest leakage power. Fig. 5 shows how to replace an NAND2 gate to reduce its leakage power. The NAND2 gate is in WLS with leakage power 454.71nW, when its input is 11. We replace it with an NAND3 gate, of which the leakage power is 249.1nW during the standby time. Then we can save up to 45.2% of the leakage power.

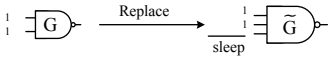

Fig. 5. A gate replacement example for leakage reduction.

3) Different input vector dependency of NBTI and leakage: All 1's will be the best input pattern with the smallest NBTI-induced degradation for all gate types. Meanwhile, leakage power varies among different input vectors. We simulate all the cells (NAND/AND, NOR/OR, INV, BUF) in the library, and find out that the best case input patterns to mitigate the leakage for NAND/AND/INV gates are all 0's at the inputs, while for NOR/OR/BUF gates are all 1's at the inputs. Therefore, although NBTI and leakage both depend on the input patterns, we can see the discrepancy: for NAND/AND/INV gates, the input pattern for least leakage will lead to worst NBTI-induced delay degradation; on the contrary, for NOR/OR/BUF gates, the input pattern for least leakage will lead to best case NBTI-induced delay degradation. Consequently, if we use pure IVC technique, the best input vector for leakage may lead to worse NBTI induced degradation, and vice versa. Therefore, we have to get a thorough control of internal node state through INC techniques, such as gate replacement technique, so that the internal node state can be carefully chosen to meet both leakage power and lifetime requirements.

4) Overhead analysis of gate replacement: Gate replacement will introduce delay and area overhead; however, these overhead can be controlled by adding delay and area constraints during the optimization algorithm, or transistor re-sizing. In this paper, the delay constraint is set to be less than 5% of the original delay at time 0 after gate replacement. From our experimental results, although delay requirement at time 0 is relaxed, we will get a better circuit delay after 10 years. For power overhead, the dynamic power overhead is trivial, because the sleep signal remains constant at both active and standby mode; the leakage power overhead during circuit active mode caused by the leakage difference of different gate types can be neglected if the standby time is long enough.

IV. Gate Replacement Algorithms

In this section, we propose our two fast gate replacement algorithms: Direct Gate Replacement algorithm and Divide and Conquer Based Gate Replacement algorithm.

A. Direct Gate Replacement (DGR) algorithm

Similar to the previous gate replacement algorithm [12], there are also two key steps for the Direct Gate Replacement: 1) Get the optimal input vector for circuits; 2) Gate replacement based on the optimal input vector. We follow the two steps and amend the previous algorithm to further consider NBTI induced circuit degradation together with leakage power.

1) Get the optimal input vector: An optimal input vector is chosen from 10K random input vector search. Since we are considering NBTI effect and leakage power together, the object function is as follows:

$$F(D_{circuit}, P_{leakage}) = A \times D_{circuit} + B \times P_{leakage} \quad (2)$$

where $D_{circuit}$ is the circuit delay after 10 years; the $P_{leakage}$ is the circuit leakage power at time 0. A and B are two weight constants for circuit designers to balance the leakage power requirement and circuit lifetime requirement. The best leakage and circuit delay results of random search are used as our reference.

2) Direct Gate Replacement based on the optimal input vector: In the DGR algorithm, we first arrange all the gates in the circuit into a topological order. The topological order guarantees that when we find a gate at its WLS, all its predecessors have already been considered. Then all the gates are evaluated one by one according to this order. The detailed algorithm is shown in Fig. 6. Firstly, all the critical paths are investigated to mitigate the NBTI effect, and then we evaluate the gates in the circuits to further reduce the leakage power.

Direct Gate Replacement (DGR) algorithm

Input: $\{G_1, \ldots, G_n\}$ all the gates in topological order of the circuit; SLEEP: the sleep signal; $\{x_1, \ldots, x_m\}$:input vectors

Output: a circuit 1) of the same functionality when SLEEP=0 and 2) with less leakage and NBTI-induced degradation when SLEEP=1

```
1 perform NBTI mitigation algorithm                                   //NBTI mitigation Part
2 for i=1 to n do                                                     //Leakage Reduction Part
3   if G_i is at WLS and not visited
4     if G_i is not in critical path then include G_i in selection S
5     else if G_i's output will not be changed after replacement then include G_i in selection S
6     while there is new addition to S
7       for each newly selected gate G in S do
8         temporarily replace G
9         if G's output is changed then
10          include all G's fanout gates in selection S that are unvisited and their
            output will not be changed after replacement
11    calculate leakage change caused by the replacement
12    if there is a leakage reduction then
13      mark all the gates in S as visited
14      make all the replacement above
15    else mark G_i as visited only
16    empty S
17  else mark G_i as visited
18 end
```

Fig. 6. Pseudo code for Direct Gate Replacement algorithm.

NBTI mitigation in the critical paths (Fig. 7): The first line of DGR algorithm Fig. 6 is to perform the NBTI mitigation algorithm shown in Fig. 7. When we consider a gate G_i, the critical fin-in gate G_c on the critical path is first selected (line 2). To mitigate the effect of NBTI in the critical path, the output value of G_c should be set to 1. If the output of G_c is 0 and there is a library gate $G_c^{'}$ that can replace G_c, then we replace G_c with $G_c^{'}$ (line 3-4). After the replacement, if the output is not changed to 1, then we will try to find all the fin-in gates of gate $G_c^{'}$, and replace them according to

NBTI mitigation algorithm

Input: $\{G_1, \ldots, G_n\}$: the gates in topological order of the circuit
(1) whole circuit for DGR; (2) tree circuit for DCBGR

Output: $\{R_1, \ldots, R_n\}$: replace G_i if R_i=true

```
1 for i=1 to n do
2   search the previous critical gate G_c in the critical path of G_i
3     if output(G_c)=0 then
4       Replace(G_c)
5       if output(G_c)=1 then
6         mark the replacement of G_c
7       else
8         search all the fanin gates of G_c and try to replace them according to the type of
          G_c to make output of G_c be 1
9         if output(G_c)=1 then
10          mark the replacement above
11 end
```

Fig. 7. Pseudo code for NBTI mitigation algorithm.

G_c''s type to make the output value of G_c' be 1 (line 7-8).

Leakage power reduction (Fig. 6): After the gate replacement for NBTI mitigation, all the gates are visited by topological order again. We skip the gates that 1) are not in WLS or 2) are in critical paths and their outputs will be changed after replacement or 3) have already been visited, until we find a new gate G_i at its WLS (line 3-5). Then we temporarily replace G_i and keep a set S that includes all the unvisited gates affected by the replacement of G_i. All the gates in S are temporarily replaced (line 6-10). The total leakage change caused by the replacement are calculated (line 11). If there is leakage reduction, all the gates in the set S are marked as replaced and visited(line 12-14). Otherwise we only simply mark G_i as visited (line 15). This algorithm will not be over until all the gates have been visited.

Complexity: The complexity of this algorithm is $O(n^2)$, where n is the total gate number in the circuit.

B. Divide and Conquer Based Gate Replacement(DCBGR) algorithm

Although DGR algorithm described in the previous subsection can achieve better results compared with the results of pure IVC technique, the complexity is $O(n^2)$ which is not scalable when the circuit size becomes larger. On the other hand, since the DGR algorithm is performed based on an initial input vector, the final optimization results may still have a gap with the optimal ones.

We further propose a Divide and Conquer Based Gate Replacement (DCBGR) algorithm based on the improved gate replacement algorithm in [11]: 1) the circuit is divided into several trees; 2)our dynamic programming algorithm is performed on the tree circuits to achieve better results faster; 3) we adjust the dangling nodes in the whole circuit, and continue to perform the algorithm until it converges.

1) Divide the circuit into trees: At the beginning, we divide the circuit into tree circuits by deleting some connections between gates until every gate fans out to at most one gate. For example, if a gate G fans out to k gates $G_1, ..., G_k$, we keep one connection G_i and delete other $k-1$ connections. We keep the connection that has the longest path from G_i to the outputs of the circuit. After deleting the connections, there are many dangling inputs. In this algorithm, all the dangling inputs are always equal to the output of their fan-in gates before deleting the connections.

2) Gate replacement for trees: The detailed algorithm is shown in Fig. 8, where i_j denotes the j^{th} input of G_i; $N(i)$ denotes the input number of G_i; $LK(i,z)$ denotes the minimum total leakage power of the subtree rooted at G_i, when its output is z; $V(i,z)$ denotes the input vector producing $LK(i,z)$; $\overrightarrow{x}$ denotes the input vector of a gate; $\overrightarrow{x_j}$ denotes the j^{th} bit of $\overrightarrow{x}$; $L(i,\overrightarrow{x})$ and $L_R(i,\overrightarrow{x})$ denote the leakage power of G_i and replaced G_i respectively; $Out(i,\overrightarrow{x})$ denotes the output of G_i with its input vector $\overrightarrow{x}$.

Initialization is firstly performed for all the gates from line 1 to 5. Then we perform the NBTI mitigation algorithm described in Fig. 7. We modify the algorithm in [11] to serve as our leakage reduction part.

3) Adjust dangling assignments and perform the algorithm until it converges: When we have got all the inputs of each gate, we assign the dangling inputs again by a reverse topological order. If there are any dangling inputs that have been changed, the algorithm will be repeated from the most anterior gate in the topological order with new dangling input values until the algorithm generates the same input vector or an input vector that has been appeared previously.

4) Complexity: The complexity of NBTI part is $O(Kn)$ where K is the maximum fan-in number of gates in the circuit before deleting the connections. The complexity of leakage part is $O(n)$ [11].

Divide and Conquer Based on Gate Replacement algorithm for a tree circuit

Input: $\{G_1, ..., G_n\}$ all the gates in topological order of a tree circuit

Output: V_{opt}:the optimal input vector

$\{Rep_1, ..., Rep_n\}$: replace G_i when $Rep(i,z)$=true and its output is z

1 **for** i=1 **to** n **do** *//Initialization*

2 **if** G_i is an input gate **then**

3 $LK(i,z)$=0,$V(i,z)$=z

4 **if** G_i is in critical path **then** Output(G_i)=1

5 **end**

6 **perform NBTI mitigation algorithm** *//NBTI mitigation part*

7 **for** i=1 **to** n **do** *//Leakage Reduction Part*

8 **for** each valid input vector $\vec{x}$ of G_i **do**

9 z=$Out(i,\vec{x})$, $LK(i,z)=\min_{valid\ x}\left\{\sum_{j=1}^{N(i)} LK(i_j,\overrightarrow{x_j})+L(i,\vec{x})\right\}$, $V(i,z)=\bigcup_{j=1}^{N(i)} V(i_j,\overrightarrow{x_j})$

10 **if** G_i is not in critical path or in critical path but its output will not be changed after replacement **then**

11 temporarily replace G_i

12 **if** $L_R(i,\vec{x}) < L(i,\vec{x})$ **then**

13 $LK(i,z)=\min_{valid\ x}\left\{\sum_{j=1}^{N(i)} LK(i_j,\overrightarrow{x_j})+L_R(i,\vec{x})\right\}$, $V(i,z)=\bigcup_{j=1}^{N(i)} V(i_j,\overrightarrow{x_j})$,

14 $Rep(i,z)$=true

15 **end**

16 **end**

17 V_{opt}=$LK(n,0)>LK(n,1)?V(n,1):V(n,0)$

18 calculate Rep in reverse topological order

Fig. 8. Pseudo code for Divide and Conquer Based Gate Replacement algorithm for a tree circuit.

C. C17 circuit as an example of DCBGR algorithm

Fig. 9 shows an example of the DCBGR algorithm for circuit C17. At the beginning, we divide the circuit into trees by deleting connections in Fig. 9(1). $G1$,$G4$, and $G6$ have dangling inputs. If we set the input vector of the circuit to all 0's then the value of these dangling inputs are 011, we set 011 to these dangling inputs as their initial values in Fig. 9(2).

Then we run the algorithm for two different object functions: leakage power reduction only and NBTI mitigation only. The critical paths are marked in red in Fig. 9(5). If NBTI mitigation is considered, the internal node values along these paths should be 1 as more as possible. Then the dynamic algorithm will generate optimal input vectors for different object functions (Fig. 9(3)). We calculate all the logic values in the circuit and assign new dangling inputs in Fig. 9(4). With the new dangling inputs, the algorithm is repeated again until the algorithm converges to optimal input vectors for different object functions as shown in Fig. 9(5). Hence, the optimal input vector for leakage is 00010 while the optimal input vector for NBTI is 11000.

The detailed results are listed in Table I. D_O is the original delay at time 0. D_{nbti} is the circuit delay after 10 years. LK is the leakage power at time 0. The optimal result for NBTI can save the circuit degradation from 8.79% to 1%, since the NBTI effect are eliminated

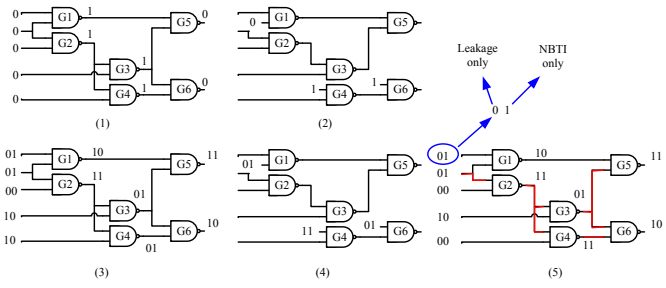

Fig. 9. An example of DCBGR algorithm for NBTI and leakage mitigation.

when the circuit is standby. The optimal result for leakage can reduce 31.1% leakage power compared with the result for NBTI only.

TABLE I
DCBGR RESULTS FOR C17 CIRCUIT.

Object function	$D_O(ns)$	$D_{nbti}(ns)$	Degradation	LK(mW)
Leakage only	0.0796	0.0866	8.79%	1.44E-6
NBTI only	0.0796	0.0804	1%	2.09E-6

V. IMPLEMENTATION AND SIMULATION RESULTS

A. Implementation

We implement our NBTI/Leakage co-simulation flow and the gate replacement algorithms in C++. A commercial static timing analysis tool PrimeTime from Synopsys is used to perform the timing analysis and generate the timing report, as well as the internal node signal probabilities. Benchmark circuits are synthesized using a 65nm library from industry. Some key technology parameters are: V_{dd} = 1.0V; $|V_{th}|$ = 0.20V for both NMOS and PMOS transistors; T_{ox} = 1.2nm. ISCAS85 benchmark and some arithmetic components circuits are used to evaluate our algorithms. The active time temperature T_{active} and standby time temperature $T_{standby}$ are both set to be 378K corresponding to the worst-case NBTI-induced circuit degradation and leakage power. Ratio of active and standby time (RAS) is set to be 1:9. We set input probabilities of all the input nodes to 0.5 for simplicity. The circuit lifetime is set to be 10 years.

B. Experimental results

1) Random search: Table II shows the results of random search for all the 20 benchmark circuits. D_O is the original delay at time 0. D_W and D_B are the worst case and best case NBTI induced delay after 10 years. LK_W and LK_B are the worst case and best case leakage power at time 0. These data are generated from 10K input vectors. The difference of NBTI induced delay degradation is on average 6% of the original delay; meanwhile the best leakage power can save on average 8.76% compared with the worst case leakage power. The results of circuits with more than 500 gates show that the IVC is less effective for larger circuits.

2) DGR algorithm: Table III shows the optimization results for leakage only and NBTI only. D_{imp} is delay improvement after 10 years. LK_{imp} is leakage improvement at time 0. These improvements are compared with the best results of Random Search in Table II. Our DGR algorithm can outperform the pure IVC about 10% and 16.56% for leakage only and NBTI only respectively. For larger circuits, we have slightly more leakage saving but potentially larger NBTI effect mitigation because the critical paths may be longer in larger circuits. We also evaluate the runtime and area penalty a_{inc}. The runtime grows fast when the circuit becomes bigger. For C6288, we may need several minutes.

Table IV shows the optimization results for simultaneous leakage and NBTI mitigation. D_{imp} is the delay improvement after 10 years. LK_{imp} is the leakage improvement at time 0. The delay and leakage improvements are compared with the best results of Random Search results: LK_s and D_s, using a weighted object function where leakage power and NBTI mitigation are treated with equivalent importance. Although IVC technique can simultaneously mitigate leakage and NBTI, our DGR performs better: 16.25% more leakage saving and 16.51% more delay compensation. Table IV also shows that the

TABLE II
RANDOM SEARCH RESULTS ($10K$ INPUT VECTORS).

Benchmark Circuits	Gate#	D_O (ns)	D_W (ns)	D_B (ns)	LK_W (mW)	LK_B (mW)
pmult4x4	122	2.522	3.154	3.004	1.17E-04	9.89E-05
c499	182	1.471	1.928	1.849	2.38E-04	2.11E-04
log16	256	1.287	1.801	1.675	2.28E-04	2.04E-04
bkung32	271	2.004	2.518	2.291	3.00E-04	2.65E-04
c432	297	3.965	4.972	4.635	2.55E-04	2.29E-04
array8x8	401	4.967	6.286	5.809	4.18E-04	3.46E-04
pmult8x8	490	4.819	6.026	5.74	4.86E-04	4.11E-04
c880	535	2.689	3.296	3.184	4.16E-04	3.76E-04
log32	640	2.138	3.231	3.047	5.67E-04	5.14E-04
c1355	942	3.089	3.733	3.67	6.60E-04	6.32E-04
c1908	977	3.763	4.548	4.355	6.93E-04	6.67E-04
c2670	1173	3.672	4.609	4.395	8.84E-04	8.49E-04
booth9x9	1206	4.195	5.057	4.883	1.14E-03	1.09E-03
log64	1536	3.862	6.259	5.967	1.36E-03	1.24E-03
c3540	1743	4.784	5.954	5.645	1.27E-03	1.20E-03
pmult16x16	1934	10.101	12.544	12.096	1.95E-03	1.67E-03
c5315	2364	4.924	6.15	5.864	1.82E-03	1.72E-03
c7552	3912	4.984	6.214	5.968	2.91E-03	2.79E-03
c6288	6656	17.94	20.886	20.579	4.69E-03	4.61E-03
pmult32x32	7570	20.921	25.784	25.247	7.65E-03	6.95E-03
average			28.00%	**22.00%**		**8.76%**
Gate#>500			28.23%	**23.30%**		**6.16%**

TABLE III
RESULTS OF DGR ALGORITHM FOR LEAKAGE POWER REDUCTION ONLY AND NBTI MITIGATION ONLY.

Benchmark Circuits	For leakage only			For NBTI only		
	LK_{imp} (%)	Runtime (s)	a_{inc} (%)	D_{imp} (%)	Runtime (s)	a_{inc} (%)
pmult4x4	1.037	1.938	6.81	17.23	0.43	3.546
c499	0.367	3.29	0.242	2.53	0.78	0.161
log16	16.832	5.25	22.2	17.47	1.12	1.316
bkung32	16.501	6.03	12.39	4.87	1.31	0.311
c432	23.638	7.32	25.23	6.16	1.55	5.215
array8x8	0.025	10.75	0.122	10.33	2.24	0.611
pmult8x8	0.0034	16.71	4.44	12.11	3.37	2.375
c880	18.398	19.25	41.6	19.53	4	6.199
log32	17.858	24.75	22.7	16.76	4.87	0.921
c1355	13.073	50.95	59.2	20.986	13.12	6.733
c1908	17.067	55.36	57.5	13.92	12.28	9.538
c2670	14.468	87.73	39.8	9.57	24.56	6.629
booth9x9	14.753	79.57	54.79	29.35	21.6	4.371
log64	18.442	127.6	23.2	19.43	27.1	0.932
c3540	12.496	165.2	46.9	16.35	52.5	7.783
pmult16x16	0.0003	212	4.97	17.8	44.6	2.881
c5315	0.312	315.1	39.8	22.51	109	6.96
c7552	5.678	795	5.8	16.8	409	9.961
c6288	9.049	490.2	13.9	35.19	754	13.249
pmult32x32	0.0234	830.4	0.275	22.29	825	8.318
average	**10.00%**	165.22	24.09%	**16.56%**	115.62	4.90%
Gate#>500	10.26%	269.49	30.73%	20.08%	191.47	6.52%

TABLE IV
RESULTS OF DGR ALGORITHM FOR SIMULTANEOUS LEAKAGE AND NBTI MITIGATION.

Benchmark Circuits	LK_s (mW)	D_s (ns)	LK_{imp} (%)	D_{imp} (%)	a_{inc} (%)	Runtime (s)
pmult4x4	1.14E-04	3.004	0.198	15.01	4.25	2.08
c499	2.11E-04	1.925	1.78	5.92	0.64	3.52
log16	2.04E-04	1.702	11.31	9.82	14.6	5.49
bkung32	2.69E-04	2.354	13.07	7.23	12.1	6.31
c432	2.38E-04	4.628	23.06	6.428	21.8	7.39
array8x8	4.06E-04	5.832	2.76	7.56	14.6	10.9
pmult8x8	4.71E-04	5.761	0.32	8.49	3.79	16.2
c880	3.80E-04	3.204	23.62	14.48	17.5	12.9
log32	5.19E-04	3.075	15.08	14.24	15.7	24.8
c1355	6.35E-04	3.686	30.63	19.571	25.4	54.5
c1908	6.71E-04	4.371	28.95	19.86	21.2	71.2
c2670	8.57E-04	4.41	22.09	12.69	18.8	97.6
booth9x9	1.09E-03	4.92	28.65	29.2	15.3	90
log64	1.26E-03	5.989	18.39	17.83	17.8	146
c3540	1.21E-03	5.707	21.05	21.25	21.9	259
pmult16x16	1.93E-03	12.14	0.26	18.15	2.61	258
c5315	1.73E-03	5.886	14.52	19.35	20.7	427
c7552	2.79E-03	6.113	21.39	27.54	11.2	1096
c6288	4.62E-03	20.73	46.98	38.53	20.7	1204
pmult32x32	7.55E-03	25.25	0.98	17.1	2.93	1567
average	5.11%	23.11%	**16.25%**	**16.51%**	14.18%	267.99
Gate#>500	3.94%	24.13%	20.75%	21.28%	16.18%	441.26

TABLE V
DCBGR ALGORITHM RESULTS FOR THREE DIFFERENT OBJECT FUNCTIONS.

Benchmark Circuits	For leakage only			For NBTI only			Co-optimization			
	LK_{imp} (%)	Runtime (s)	a_{inc} (%)	D_{imp} (%)	Runtime (s)	a_{inc} (%)	LK_{imp} (%)	D_{imp} (%)	Runtime (s)	a_{inc} (%)
pmult4x4	12.71	0.047	8.94	12.12	0.015	5.11	11.76	10.4	0.031	7.23
c499	14.04	0.031	1.29	17.79	0.031	0.56	13.59	17.0	0.031	1.29
log16	39.91	0.031	0	49.61	0.047	3.04	30.34	17.47	0.047	21.05
bkung32	40.44	0.047	1.87	14.61	0.047	4.92	23.5	14.61	0.062	1.87
c432	33.41	0.047	16.41	39.88	0.047	5.06	4.98	13.82	0.047	16.64
array8x8	3.91	0.047	6.85	25.53	0.047	2.45	2.95	14.33	0.063	6.85
pmult8x8	8.28	0.078	8.36	7.57	0.078	3.61	7.7	5.97	0.078	7.4
c880	43.35	0.079	19.75	22.91	0.079	3.38	15.52	21.24	0.109	21.16
log32	40.34	0.094	0	60.23	0.11	3.42	31.11	19.87	0.141	21.05
c1355	49.81	0.172	23.8	14.24	0.172	3.73	30.26	13.22	0.234	23.8
c1908	48.24	0.172	24.38	13.88	0.203	4.13	26.11	13.88	0.266	25.84
c2670	43.74	0.297	15.96	16.76	0.328	3.4	20.19	10.44	0.422	17.39
booth9x9	40.56	0.312	8.98	9.93	0.359	2.84	35.2	5.58	0.453	8.98
log64	40.86	0.406	0	67.85	0.484	3.67	31.89	21.07	0.594	21.05
c3540	41.67	0.516	18.17	25.49	0.594	3.25	10.24	20.24	0.735	20.98
pmult16x16	5.7	0.656	7.8	11.38	0.718	2.64	5.38	9.46	0.765	7.28
c5315	39.65	1.031	14.68	9.55	1.188	3.28	15.51	9.55	1.453	17.3
c7552	47.77	2.734	18.76	48.6	3.25	4.12	25.49	19.07	4	19.89
c6288	31.26	10.75	19.15	11.04	11.672	6	28.24	10.83	14.172	19.15
pmult32x32	8.85	16.016	7.44	16.03	16.828	1.88	8.71	14.11	18.266	7.17
average	**31.73%**	1.68	11.13%	**23.65%**	1.81	3.52%	**18.93%**	**19.17%**	2.098	14.67%
Gate#$>$500	36.54%	2.76	13.26%	23.95%	2.99	3.53%	22.36%	22.13%	3.46	17.49%

mitigation results for larger circuits are better than the average results, while larger circuits will introduce larger area penalty since more gates are replaced.

3) DCBGR algorithm: Table V shows the optimization results of DCBGR algorithm. All the results are compared with the best optimization results in Table II. The DCBGR results are better than those of DGR algorithm, while the DCBGR algorithm can save on average 100X runtime compared with previous DGR algorithm. For leakage only, DCBGR can achieve on average 31.73% leakage power saving while DGR result is 10%. For NBTI only, DCBGR can compensate on average 23.65% NBTI induced circuit degradation, while DGR result is 16.56%. The best results in Table II are better than the weighted results in Table IV, hence DCBGR algorithm can achieve better results than the DGR algorithm for co-optimization. From Table V, the results of larger circuits are also better than the average level, which is consistent with our previous finding. From the DGR and DCBGR results, the area overhead for leakage reduction is larger than that for NBTI mitigation, since algorithm for leakage will consider all the gates in the circuit while that for NBTI only considers the critical paths.

VI. CONCLUSIONS

Power and reliability become two key design goals with technology scaling down. In this paper, we have proposed two gate replacement algorithms for leakage power and NBTI-induced aging effect mitigation based on our NBTI/Leakage co-simulation platform. Both DGR algorithm and DCBGR algorithm are capable to achieve better results than pure IVC technique. The DCBGR algorithm with a complexity of $O(n)$ is much faster than the DGR algorithm. We also analyze the overhead of gate replacement technique. The area overhead for leakage power reduction is much larger than that of NBTI mitigation. Less than 5% of circuit delay at time 0 caused by gate replacement techniques will lead to about 20% delay degradation saving compared with the pure IVC technique. Furthermore, if more gates in the circuit critical paths can achieve their best leakage power with all 1's as input, the circuit leakage power will be further reduced during the NBTI optimization phase. Hence, for future work, constrained logic synthesis combined with the gate replacement technique may lead to better co-optimization results.

REFERENCES

[1] V. Huard, M. Denais, and C. Parthasarathy, "NBTI degradation: From physical mechanisms to modelling," *Microelectron. Reliab.*, vol. 46, no. 1, pp. 1–23, 2006.

[2] D. K. Schroder and J. A. Babcock, "Negative bias temperature instability: Road to cross in deep submicron silicon semiconductor manufacturing," *Journal of Applied Physics*, vol. 94, no. 1, pp. 1–18, 2003.

[3] S. Borkar, "Electronics beyond nano-scale cmos," in *Proc. DAC*, 2006, pp. 807 – 808.

[4] M. Agarwal, B. C. Paul, Z. Ming, and S. A. M. S. Mitra, "Circuit failure prediction and its application to transistor aging," in *VLSI Test Symposium, 2007. 25th IEEE*, 2007, pp. 277–286.

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

[6] S. V. Kumar, C. H. Kim, and S. S. Sapatnekar, "An Analytical Model for Negative Bias Temperature Instability," in *Proc. IEEE/ACM ICCAD*, 2006.

[7] R. Vattikonda, W. Wang, and Y. Cao, "Modeling and minimization of pmos nbti effect for robust nanometer design," *DAC*, pp. 1047–1052, Jul. 2006.

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

[9] 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. DATE*, 2007, pp. 546–551.

[10] J. Abella, X. Vera, and A. Gonzalez, "Penelope: The nbti-aware processor," in *MICRO 2007*, 2007, pp. 85–96.

[11] L. Cheng, L. Deng, D. Chen, and M. Wong, "A fast simultaneous input vector generation and gate replacement algorithm for leakage power reduction," *Design Automation Conference, 2006 43rd ACM/IEEE*, pp. 117–120, July 2006.

[12] L. Yuan and G. Qu, "A combined gate replacement and input vector control approach for leakage current reduction," *IEEE Trans. on VLSI*, vol. 14, no. 2, pp. 173–182, 2006.

[13] N. Jayakumar and S. Khatri, "An algorithm to minimize leakage through simultaneous input vector control and circuit modification," *DATE '07*, pp. 1–6, April 2007.

[14] W. Wang, V. Reddy, A. Krishnan, R. Vattikonda, S. Krishnan, and Y. Cao, "Compact modeling and simulation of circuit reliability for 65nm cmos technology," *IEEE Transactions on Device and Materials Reliability*, vol. 7, no. 4, pp. 509–517, 2007.

[15] S. Bhardwaj, W. Wenping, R. Vattikonda, A. Y. C. Yu Cao, and S. A. V. S. Vrudhula, "Predictive modeling of the nbti effect for reliable design," in *Conference 2006, IEEE Custom Integrated Circuits*, 2006, pp. 189–192.

[16] V. Huard, C. Parthasarathy, N. Rallet, C. Guerin, M. Mammase, D. Barge, and C. Ouvrard, "New characterization and modeling approach for nbti degradation from transistor to product level," *IEDM 2007.*, pp. 797–800, 10-12 Dec. 2007.

[17] T. Grasser, B. Kaczer, P. Hehenberger, W. Gos, R. O'Connor, H. Reisinger, W. Gustin, and C. Schlunder, "Simultaneous extraction of recoverable and permanent components contributing to bias-temperature instability," *IEDM 2007.*, pp. 801–804, 10-12 Dec. 2007.

[18] B. Paul, K. Kang, H. Kufluoglu, M. Alam, and K. Roy, "Temporal Performance Degradation under NBTI: Estimation and Design for Improved Reliability of Nanoscale Circuits," in *Proc. DATE*, vol. 1, 2006, pp. 1–6.