Layout-Driven Hot-Carrier Degradation Minimization Using Logic Restructuring Techniques

Chih-Wei (Jim) Chang, Kai Wang, Malgorzata Marek-Sadowska
Department of Electrical and Computer Engineering,
University of California, Santa Barbara, CA 93106, USA

Abstract

The rapid advances in semiconductor manufacturing technology have created tough reliability problems. Failure mechanisms such as hot-carrier effect, dielectric breakdown, electrostatic discharge and electromigration have posed tremendous threats to the long-term reliability of VLSI circuits. As a result, designers not only need analysis tools to locate the problem, but also design-for-reliability tools to correct it. However, these problems often surface when the physical layout is done and relatively few logic changes can be made. In this paper, we target the performance optimization issues in the context of hot-carrier induced degradation. A layout driven approach combining rewiring, discrete gate resizing, and pin reordering is proposed. Experimental results show that rewiring-based incremental logic restructuring is a very powerful technique in post-layout design for reliability

1. Introduction

Design trends, such as device miniaturization, system-on-a-chip integration (SOC), and higher operating frequencies, increase concerns about circuit reliability. Hot-carrier effect (HCE) is one of the major failure mechanisms affecting long-term reliability. As the device dimensions shrink to the deep sub-micron ranges, the electric field in the transistor channel increases significantly. Electrons and holes travelling in the channel may gain high enough kinetic energy be injected into the gate oxide and cause permanent changes in the oxide-interface charge distribution. In an NMOS transistor, HCE leads to trans conductance degradation, shift in threshold voltage, and decrease in the drain current driving capability. The performance degradation of particular devices leads to the overall circuit performance degradation. The transistor degradation behavior is a function of time, the amount of transitions, its fanin driving capability, and geometric dimensions. The effects accumulate as the device is in operation. As a result, circuits age over time.

Hot-carrier effects have been studied extensively in the past few decades[7][13][19]. Efficient techniques for accurate transistor-level reliability simulations are implemented in both academic[12] and commercial tools[17]. However, transistor-level simulations of large industrial circuits are computationally too expensive to be feasible. [8] proposed a probabilistic approach to estimate the degradation effect on timing. Recently, a ratio-based gate-level degradation model was proposed in [20] as a higher level abstraction. Each cell from the technology library is pre-characterized for its degradation behavior under various stress conditions. For an excellent review, refer to [7] for more information.

Design for reliability (DfR) techniques considering hot-carrier effects fall into two categories. One category includes such techniques as transistor reordering and resizing[4], technology mapping[2] and technology-independent factorization[11] to minimize the maximum hot-carrier degradation effects among all transistors in the circuit. That is, each transistor in the circuit is labelled with a relative degradation factor and the optimization goal is to minimize the maximum of these factors. This goal targets improvement of the Mean-Time-To-Failure (MTTF) under the assumption that if any device in the circuit fails, the whole circuit fails. The other category of techniques includes the method proposed by Li et al which performs input pin reordering and gate resizing [9] to minimize the performance degradation impact on the entire circuit. The idea is that not all devices in the circuit are of equal importance as far as overall performance is concerned. Devices not on the critical paths can potentially tolerate more degradation without affecting the overall performance.

However, all these techniques operate at the gate/transistor level without knowing the physical layout information, which has tremendous impact on device degradation. For example, input slew rate to the transistor and effective output switching are identified as the most important factors[6][9] determining device degradation. Due to the resistive behavior of deep sub-micron interconnects, slew rates estimation at the gate level is very inaccurate without knowing the placement and routing information. Also, because of the underlying Boolean functionality, some gates experience more switching than the others. Without changing the logic structure of the circuit, switching activity cannot be controlled for optimization purposes.

Rewiring is a technique which tries to reconnect wires in a network without changing the overall functionality. Previous works include Redundancy Addition and Removal (RAMBO)[5], Set of Pairs of Functions to be Distinguished (SPFD)[18] and functional symmetry based rewiring[1]. These techniques have the advantage that only local modifications of the netlists are made - some wires are added and some are removed. These techniques are especially suited for post-layout optimization since the logic can be changed while the placement solution can be minimally perturbed.

In this paper, we propose a layout-driven hot-carrier degradation minimization approach which combines the functional symmetry based rewiring technique with traditional gate resizing and pin reordering techniques. The whole optimization process operates on the placed (routed) netlist. The philosophy behind this setting is that interconnect information is very crucial in accurately determining the hot-carrier degradation on each device. At the post-layout level, large scale logic restructuring, such as re-factorization and re- mapping is not desirable since circuit elements have already been placed. We adopt the functional symmetry based rewiring technique

DAC 2001, June 18-22, 2001, Las Vegas, Nevada, USA.

proposed in [1] to restructure the logic while keeping the existing placement solution minimally perturbed.

2. Preliminaries

In this section, we first review the ratio-based degradation model from [16][20] followed by the probabilistic switching estimation technique from [10] and the functional symmetry based rewiring technique of [1].

Let T_{fresh} and T_{aged} be the fresh and aged pin-to-pin signal delays. α is the aged-to-fresh signal delay ratio which characterizes the overall degradation of all transistors in the gate due to hot-carrier effect. These variables are defined for each transition type (rise or fall) of each signal path of the logic gates. Equation 1 shows the relationship between T_{fresh}, T_{aged} and α.

$$T_{aged} = \alpha T_{fresh} \qquad (1)$$

α can be characterized by the following equation:

$$\alpha = \left(\sum_{i=1\ldots n} \alpha_i \right) - n + 1 \qquad (2)$$

where *n* is the number of transistors in series and α is the aged-to-fresh delay ratio when only pin *i* is under stress. It is defined as follows:

$$\alpha_i = \Psi(T_{slew}, C_L, N_{sw}) \qquad (3)$$

In this equation, T_{slew} is the slew rate of the input pin. C_L is the load capacitance of the gate output. N_{sw} is the number of effective switching of the input pin. By "effective" we mean that the input pin switching leads to an output pin switching. α can be viewed as a degradation factor of the i-th transistor in series in the conducting path. Conceptually, slower slew rates put transistors in undesirable bias conditions for a longer periods of time, larger load capacitances stress transistor longer during charging and discharging, and more effective switching stresses the transistors more often. These all lead to more transistor wear-out. Function Ψ is determined in the process of transistor level simulations and the results are used to build a three-dimensional table for later reference.

Equation 2 might be worth further explanation. Essentially, it decouples/simplifies the effect of degradation on the conducting path into individual contributions from transistors along the path. For example, consider the high-to-low delay of a two-input NAND gate in Fig. 1. As a first-order simplification, transistors in series are regarded as resistors in series. When only M1 switches in the whole lifetime, R1 degrades to $\beta_1 R1$ with $\beta_1 > 1$ and the delay degradation ratio α_1 can be written as:

$$\alpha_1 = \frac{T_{aged(M1)}}{T_{fresh}} = \frac{\beta_1 R1 + R2}{R1 + R2}$$

Fig. 1(c) shows the effect. Similarly, if only transistor M2 switches over the whole lifetime, α_2 can be written as:.

$$\alpha_2 = \frac{T_{aged(M2)}}{T_{fresh}} = \frac{R1 + \beta_2 R2}{R1 + R2}$$

Now, after characterizing degradation of each of the individual transistors, the effect of both degraded transistors M1 and M2

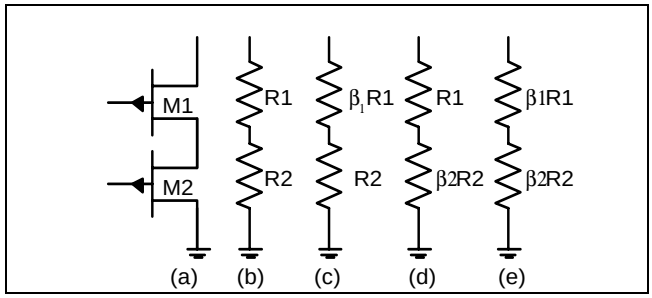

Fig. 1: Ratio derivation[20]

along the conducting path can be added (see Fig. 1(e)) and the resulting α is as follows:

$$\alpha = \frac{T_{aged}}{T_{fresh}} = \frac{\beta_1 R1 + \beta_2 R2}{R1 + R2} = \alpha_1 + \alpha_2 - 1$$

This equation is for the case when *n* equals two. When the number of transistors in series is *n*, this equation can be generalized yielding Equation 2. Based on this extended pin-to-pin delay model, full chip timing/reliability simulation is demonstrated to be 2 to 4 orders of magnitude faster[20] while accuracy is within 1% of the transistor-level counterpart.

Najm introduced the notion of equilibrium probability and transition density for power estimation[10]. The equilibrium probability of a signal *x*, denoted P(*x*), is the fraction of time *x* is evaluated to logic 1. The transition density of *x*, denoted D(*x*), is the average number of transitions per unit time. Under spatial and temporal independence assumption, an efficient algorithm was introduced to propagate the density values from the primary inputs throughout the circuit. To see how the propagation algorithm works, recall the concept of Boolean difference: if *f* is a Boolean function that depends on *x*, then the Boolean difference of *f* with respect to *x* is defined as:

$$\frac{\partial f}{\partial x} = f|_{x=0} \oplus f|_{x=1} = f_{\bar{x}} \oplus f_x$$

Here, $\oplus$ represents the Boolean exclusive-or function. The Boolean difference is the XOR of the positive and negative cofactors with respect to *x*. Essentially, $\frac{\partial f}{\partial x}$ is the condition that if there is a transition on *x*, there is a corresponding transition on *f*. For example, let *f* be a two-input AND gate. i.e. $f = x_1 \cdot x_2$. The Boolean difference of *f* with respect to x_1 is $\frac{\partial f}{\partial x_1} = 0 \oplus x_2 = x_2$. So, when $x_2 = 1$, any transition at x_1 will cause a corresponding transition at *f*.

It is shown that under the spatial independence assumption, the transition density at the output of a function *f* can be calculated by the following equation:

$$D(f) = \sum_{i=1}^{n} P\left(\frac{\partial f}{\partial x_i}\right) D(x_i) \qquad (4)$$

Let *f*: $\{0, 1\}^n \rightarrow \{0, 1\}$ be a single-output completely specified Boolean function defined on the input set X = $\{x_0, x_1, \ldots, x_{n-1}\}$. Four cofactors can be defined with respect to two variables x_i, x_j

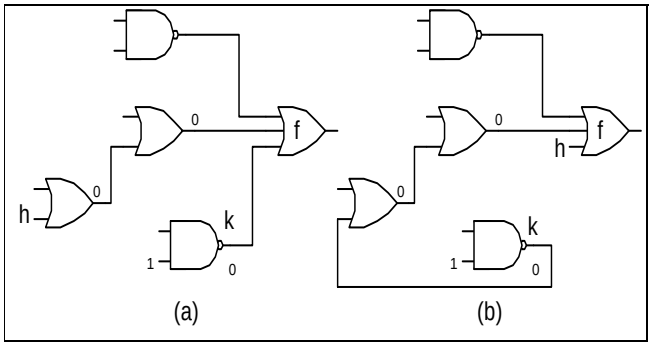

Fig. 2: h and k are swappable

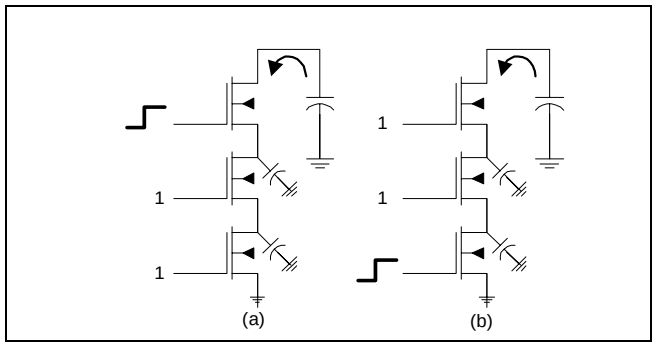

Fig. 3: Pin position affects delay and reliability

$\in$ X. i.e. $f_{\overline{xi}\,\overline{xj}}$, $f_{\overline{xi}\,xj}$, $f_{xi\,\overline{xj}}$, and $f_{xi\,xj}$. *Symmetry* is defined as the equivalence between any of the two cofactors. Among them, two types of symmetries are of special interest and are stated below:

Definition 1: x_i and x_j are *non-equivalence symmetric* (NES) in f(X) if and only if $f_{\overline{xi}\,xj} = f_{xi\,\overline{xj}}$. That is, the exchange of x_i and x_j does not change f. i.e. $f(..., x_i,..., x_j,...) = f(..., x_j,..., x_i,...)$.

Definition 2: x_i and x_j are *equivalence symmetric* (ES) in f(X) if and only if $f_{\overline{xi}\,\overline{xj}} = f_{xi\,xj}$. That is, the exchange of $\overline{x}_i$ and $\overline{x}_j$ does not change f. i.e. $f(..., x_i,..., x_j,...) = f(..., \overline{x}_j,..., \overline{x}_i,...)$.

We now discuss the concept of *implication supergate* and its relationship to functional symmetry. Implication supergate was introduced in [15] for Automatic Test Pattern Generation (ATPG) applications. It was later generalized, used to identify easily detectable functional symmetries in a Boolean network and applied for post-placement delay optimization[1]. *Generalized implication supergate* is a set of connected gates that functionally behave like a big AND, OR, or XOR gate. For example, the circuit in Fig. 2(a) consists of five gates but they behave functionally the same as a big OR gate rooted at *f* with some input phase inversions. To extract all generalized implication supergates from a given netlist, we start from the primary outputs and process each gate in a reverse topological order. At each primary output, depending on its gate type, either direct backward implication or XOR propagation is attempted[1]. Multiple-fanout nodes, or nodes where backward propagation stops, are treated as new implication supergate roots and the propagation process continues. This procedure stops when all primary inputs are reached. After the extraction, the network is uniquely partitioned into AND, OR, and XOR supergates with inverters and buffers at their pins.

The most important result developed in [1] is that wires that are covered by the same generalized implication supergate are either non-equivalent or equivalent symmetric and can be swapped without changing the overall functionality of the circuit. This is illustrated in Fig. 2. Since pins *h* and *k* are covered by the same implication supergate rooted at *f*, they are functionally symmetric and can be swapped. The swap results in netlist shown in Fig. 2(b) which is functionally equivalent to the one in Fig. 2(a). Wire swapping has two major effects on circuit performance. First, it may reduce the number of levels the critical path has to travel. For example, if the critical path extends from *k* to *h* in Fig. 2(b), then it is beneficial to swap the wires to achieve the result in Fig. 2(a). Second, the interconnect loading can dramatically be reduced on the critical path and hence improve the slew rate. Swapping also changes the switching activity of the gates associated with the swapping as well as gates in their fanout cones.

3. Performance Optimization for Hot Carrier Effect

In this section, we first discuss the motivation of this work followed by the problem formulation and the proposed algorithm.

3.1 Motivation

Even though large-scale fast yet accurate reliability simulation is feasible, there are no systematic ways to correct the timing degradation found. In [6], design for reliability considerations at circuit level are given as a set of guidelines. In our paper, we move one step further by considering the design for reliability issues at the logic level guided by accurate layout information. Our main argument is that only circuit level consideration is not adequate. From Equation 3, it is clear that degradation is affected by three variables. Namely, T_{slew}, C_L, and N_{sw}. In a digital circuit, these three parameters could potentially vary dramatically among different gates. In other words, the stress is uneven. Some transistors wear out faster than the others. For example, because of the underlying Boolean logic, some gates in the circuit switch much more frequently than the others and some gates have larger loads than the others. Circuit-level techniques simply cannot address all these phenomena.

We now discuss the effect of gate sizing, pin reordering, and rewiring on HCE and circuit performance. To improve T_{slew} of an input pin, the driver of the pin can be resized up to have better driving capability to improve the slew rate. However, a larger driver causes larger loading C_L on the preceding stage, which needs to be taken into account for trade-off. Rewiring can be used to either reduce the loading of the driver by minimizing the interconnect loading, or simply to replace the driver by another driver which has better driving capability or is physically closer to the sink pin to reduce the interconnect length. The effective switching N_{sw} can be changed only by rewiring the netlist.

Another HCE effect concerning the ordering of transistors has been observed in [6][9]. The top nMOS transistors that are directly connected to the output node have the potential of experiencing the most damage if they switch last. This is because the stress on nMOS transistor is directly related to V_{ds} (drain-to-source voltage difference). Suppose there is an effective transition on pin x_i (other inputs have already arrived). When x_i connects to the output node, V_{ds} is larger than V_{ds} in the case when x_i is closer to the ground. This is due to the charge redistribution on internal nodes. Fig. 3 illustrates this situation. However, conventional timing optimization techniques tend to put the last arriving signal closer to the out-

put node to minimize the overall arrival time. This trade-off also needs to be considered during optimization.

3.2 Problem Formulation

Let $T_{fresh}(D)$ and $T_{aged}(D)$ be the non-degraded and degraded critical path delays of a design *D*. We formulate the following problem:

Fresh Delay Constrained Aged Delay Optimization Problem (FDCADOP):

Instance: We assume that we are given a placed and routed standard-cell design D and a hot-carrier degradation pre-characterized standard cell library L. Let P be the family of the sets of pins that are identified to be functionally symmetric and hence can be swapped without changing the overall functionality of the design D.

Configuration: Each gate $g \in$ D can be resized by a functionally equivalent while electrically different cell from L. Each functionally symmetric pin pair $p \in$ P can be swapped to change the logic structure of D. We consider pin reordering as a special case of functional symmetry.

Optimization: Let D' be the new design after gate resizing and pin swapping from the original design D. The optimization is to find a D' that satisfies the following goals:

$$minimize \ \ T_{aged}(D')$$

s.t.

$$T_{fresh}(D') \le T_{fresh}(D)$$

Essentially, we want to minimize the performance degradation of the aged design by redistributing the stress to logic elements that are not on the timing critical path. However, we are not willing to sacrifice any performance loss of the fresh design. It is to be noted that simply optimizing $T_{fresh}(D')$ (traditional delay optimization goal) does not necessarily lead to an optimized solution of $T_{aged}(D')$. This is because an optimized $T_{fresh}(D')$ might lead to unfavorable stress conditions on the transistors along the critical path. As discussed in the previous section, various trade-offs have to be considered simultaneously, and we need a unified algorithm taking into account both performance requirement and the aging effect to resolve this situation. Our algorithm will be detailed in the next section.

3.3 The Algorithm

To solve the FDCADO problem, we adopt a probabilistic approach based on [10] to estimate the N_{sw} of each pin of the gate. Under the spatial and temporal independence assumption, N_{sw} for a pin x_i of gate *f* under zero-delay model can be expressed as:

$$N_{sw} = D(x_i)P\left(\frac{\partial f}{\partial x_i}\right)T$$

where $D(x_i)$ is the per unit time transitions number of the input pin x_i, $P\left(\frac{\partial f}{\partial x_i}\right)$ is the probability of propagating a transition from pin x_i to the output of gate *g*, and T is the total time period. We assume that half of the transitions N_{sw} are low-to-high and half are high-to-low.

```
function FDCADO (N: placed netlist, L: technology library)
Calculate_Slack();
Extract_Generalized_Implication_Supergate();
update ← all GSG in the circuit;
moves ← ∅;
loop {
  old_cost ← Cost(N);
  foreach GSG ∈ update {
    GSG.move ← ∅;
    GSG.fit ← Fitness(N);
    foreach gate g ∈ L implementing GSG {
      fit ← Fitness(N[GSG ← g]);
      if (fit > GSG.fit) {
        GSG.fit ← fit;
        GSG.move ← g;
      }
    }
    foreach swap s of GSG {
      fit ← Fitness(N[GSG ← GSG(s)];
      if (fit > GSG.fit) {
        GSG.fit ← fit;
        GSG.move ← s;
      }
    }
  }//end of "foreach gsg ∈ update"
  moved ← BestMultipleMove(N);
  update ← GetPerturbedNodes(N, moved);
} until Converge(old_cost, Cost(N), moved)
```

Fig. 4: Algorithm

Our approach is based on the gate sizing algorithm developed in [3] and illustrated in Fig. 4. The arrival time and required time are first propagated in the circuit and the slack, which is defined as the required time minus the arrival time, is calculated. Essentially, the timing optimization problem is to maximize the minimum slack in the whole circuit. Generalized implication supergate (GSG) extraction[1] is then applied to extract all GSGs from the input netlist. All possible gate resizing and pin swapping choices inside a GSG are considered as the set of possible moves for that GSG. The move which has the best fitness value is selected as the best move of the GSG. The fitness is defined as follows:

$$Fitness = \begin{cases} 0 & \Delta S_{fresh} < 0, \ \Delta S_{aged} < 0 \\ e^{\alpha \Delta S_{fresh} + \beta \Delta S_{aged}} & otherwise \end{cases}$$

where ΔS_{fresh} and ΔS_{aged} are the change of minimum slack as a result of the move in the local neighborhood defined as gates within a user-specified level limit from the source of the move. The neighborhood is an effective way to quantify the benefit of each move[3]. It has been observed that slack change outside the neighborhood tends to be very small and can be neglected for fitness calculation. For each possible move, we propagate the change in slew rate switching activity, as well as arrival/required time in the local neighborhood, to calculate the minimum slack change. The upper part of the fitness function defines the situation when the move degrades both the fresh and aged circuits. This kind of move is not desirable and is assigned zero fitness value, which means it will never be executed. On the other hand, the exponential dependency on $\alpha \Delta S_{fresh} + \beta \Delta S_{aged}$ gives priority to moves

which accelerate both the fresh and aged circuit. α is typically chosen to be much larger than β to penalize the situation when the fresh delay is degraded while the aged delay is improved.

Initially, the *update* list contains all GSGs in the circuit. In the first phase, best move of each GSG is selected based on its fitness value. During the second phase, the best moves of all GSGs in the update list are sorted with respect to their fitness values. A sequence of moves (Best Multiple Move) is then determined and executed based on the sorted list to maximize the overall gain in fitness. GSGs which are neighbors to the executed moves are put into the update list (Get Perturbed Nodes) for the next round of fitness calculation. These two phases iterate until either the iteration limit is exceeded or no more improvements can be made.

It is to be noted that when designing our algorithm, we intentionally make no assumption about the property of the function Ψ in Equation 3. That is, no matter how Ψ is characterized, either by analytical equation, empirical formula, or simply a table-look-up method, our algorithm can still apply. This further demonstrates the robustness of our approach.

4. Experimental Results

A prototype tool has been implemented on top of the SIS 1.3 package[14] and tested on a set of benchmarks from MCNC 91 benchmark suite. All benchmarks are first optimized using SIS script "script.rugged" and timing-driven technology mapped to a 0.35μm commercial standard cell library. The library contains gates with the number of inputs ranging from 2 to 4. Each logic type has 4 different implementations. The mapped netlist is fed to a commercial timing-driven placer. Cell locations are extracted after placement. To model interconnect, we use 2 pF/cm for unit capacitance and 2.4KΩ/cm for unit resistance. All benchmarks runs are performed on an AMD Athlon 650Mhz processor with 256 MB of memory. To characterize the cell library with aging information, we use the transistor level aging simulator BERT[12] together with HSPICE and verified with analytical equations obtained from [7] for an 10-year period. We only characterize the aging effect on NMOS transistor since the degradation of a PMOS transistor is relatively negligible[7] for the technology we are using.

To model the interconnect after placement, a net model is necessary to estimate the delay along the interconnect. Assume all pins have known coordinates after placement. Each net is modeled as a star: the center of the star is the center of gravity of all its terminals. A net is divided into several segments: from source to the star center and from the star center to each sink. Each segment is modeled by lumped RC. We use Elmore delay model for delay calculation. Since the distance from the star center to each sink may vary, each sink may have different delay and slew rates from the source. It is to be noted that our algorithm does not depend on the net model. The optimization can also work on global or detailed routed designs.

Three algorithms have been implemented to show their relative strength in optimizing the aged circuit under fresh delay constraint: 1) Pin reordering, 2) Gate sizing, and 3) A hybrid approach discussed in the previous section. Experimental results are shown in Table 1. The first column lists the name of each benchmark. The second and third column show the fresh and aged delays of the original circuit after placement. This fresh delay is used as the timing constraint for the aged circuit optimization. Column four is the percentage of performance degradation due to circuit aging. Column five and six are the aged delay after pin reordering and the corresponding degradation percentage as compared to the original fresh delay. Columns seven and eight show the aged delay after gate sizing and the corresponding degradation percentage. Column nine shows the aged delay after our hybrid approach and column ten gives the degradation percentage compared to the original fresh delay. Column eleven, twelve, and thirteen are the CPU time in seconds for pin reordering, gate sizing, and our approach, respectively.

The results clearly show the advantage of considering logic restructuring in combination with traditional techniques. On average, the percentage of degradation can be lowered to be within 1% of the original fresh delay using our technique, while pin reordering and gate sizing result in 8.8% and 4.6% of degradation respectively. The percentage of area change caused by gate sizing is within 3% and is assumed to be handled by an ECO placer.

5. Conclusion and Future Work

In this paper, a timing optimizer targeting directly the circuit aging behavior is proposed. Combining functional symmetry based on rewiring, pin reordering, and gate sizing, our approach shows much better results than the individual traditional approaches. The major difference of our approach is that layout information is taken into account while exploring the logic flexibility without perturbing the existing placement solution too much. On the average, we can minimize the impact of circuit aging to be within one percent of the original design specification.

Our approach can further be improved in several directions, especially on the estimation of switching activities. The probability simulation framework proposed in [8] is directly applicable. Also, the logic restructuring approach used in this work is limited to functional symmetries-based rewiring. Other rewiring techniques such as [5][18] could potentially be used to explore another degree of freedom during optimization. Buffer insertion can also be used to improve the signal slew rate for better degradation control.

Due to the continuing shrinking of device feature sizes, various physical effects pose serious threats to circuit performance and reliability. These effects often emerge after the physical layout is generated and hence relatively few logic changes can be made. We have demonstrated the use of functional symmetry based rewiring to minimize the effect of hot-carrier induced circuit aging at post-layout level. In the deep sub-micron era, the ability to perform incremental logic restructuring is of paramount importance. Techniques presented in this paper serve as an essential step to cope with these physical effects.

Acknowledgments

This work is sponsored in part by Semiconductor Research Corporation under grant 98-DJ-619 and National Science Foundation under grant CCR9. The authors would like to thank Dr. Tan-Li Chou from the Strategic CAD Labs of Intel Corporation for his motivation at the early stage of this work.

References

[1] C.-W. Chang, C.-K. Cheng, P. Suaris, and M. Marek-Sadowska, "Fast Post-placement Rewiring Using Easily Detectable Functional Symmetries", pp. 286-289, Design Automation Conference, 2000

[2] Z. Chen and I. Koren, "Technology Mapping for Hot-Carrier Reliability Enhancement", Proc. of the SPIE - The International Society for Optical Engineering, vol.3216, pp. 42-50, 1997

[3] O. Coudert, "Gate Sizing for Constrained Delay/Power/Area Optimization", in IEEE Trans. on VLSI, pp. 465-472, Dec. 1997

TABLE 1. Experimental Result

Circuit	T_{fresh} (D)	T_{aged} (D)	%	T_{aged} (D') PR	% PR	T_{aged} (D') GS	% GS	T_{aged} (D') Ours	% Ours	CPU PR	CPU GS	CPU Ours
C432	11.0	12.1	-10.4	12.1	-10.2	11.3	-2.6	11.00	0.0	1.0	1.4	6.3
C499	6.4	7.2	-11.8	7.2	-11.5	6.8	-5.4	6.6	-2.1	3.2	6.4	17.1
C880	10.7	11.3	-6.1	11.1	-4.6	11.0	-2.7	10.7	-0.2	1.5	4.5	6.0
C1355	6.3	7.1	-12.0	7.1	-11.4	6.7	-5.5	6.5	-2.4	3.2	6.8	16.9
C1908	10.4	10.9	-5.1	10.9	-5.1	10.8	-4.4	10.4	0.0	2.6	3.8	15.4
C3540	14.2	16.0	-12.2	15.8	-11.3	15.0	-5.4	14.5	-2.1	13.2	19.0	47.5
C5315	10.2	11.2	-9.8	11.0	-8.4	10.5	-3.2	10.3	-1.5	6.8	12.0	29.1
C6288	40.4	44.7	-10.5	44.5	-9.9	44.2	-9.3	40.5	-0.2	18.2	36.4	104.6
C7552	10.7	11.5	-7.7	11.5	-7.1	11.0	-2.9	10.7	-0.4	7.8	10.4	21.3
average			-9.5		-8.8		-4.6		-1.0			

[4] A. Dasgupta and Ramesh Karri, "Hot-Carrier Reliability Enhancement via Input Reordering and Transistor Sizing", Proc. of Design Automation Conference, pp. 819-824, 1996

[5] L. A. Entrena, K. -T. Cheng, "Combinational and Sequential Logic Optimization by Redundancy Addition and Removal", IEEE Trans. on Computer-Aided Design, 1995, pp. 909-916

[6] Y. Leblebici, "Design Considerations for CMOS Digital Circuits with Improved Hot-Carrier Reliability", IEEE Journal of Solid-State Circuits", vol. 31, No. 7, pp. 1014-1024, 1996

[7] Y. Leblebici and S.-M. Kang "Hot-Carrier Reliability of MOS VLSI circuits", Kluwer Academic Publishers, 1993

[8] P.-C. Li, G. I. Stamoulis, and I. N. Hajj, "A Probabilistic Timing Approach to Hot-Carrier Effect Estimation", IEEE Trans. on Computer-Aided Design, vol. 13, No. 10, Oct. 1994

[9] P.-C. Li, and I. N. Hajj, "Computer-Aided Redesign of VLSI Circuits for Hot-Carrier Reliability", IEEE Trans. on Computer-Aided Design, vol. 15, No. 5, May 1996

[10] F. N. Najm, "Transition Density: A New Measure of Activity in Digital Circuits", IEEE Transactions on Computer-Aided Design, vol. 12, no. 2, Feb, 1993, pp. 310-323

[11] K. Roy and S. Prasad, "Logic Synthesis for Reliability: An Early Start to Controlling Electromigration & Hot-Carrier Effects", IEEE Trans. on Reliability, vol. 44, No. 2, 1995

[12] R. H. Tu et al, "Berkeley Reliability Tools - BERT", IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, vol 12, No. 10. Oct. 1993

[13] C. Hu et al, "Hot-Electron-Induced MOSFET Degradation - Model, Monitor, and Improvement", IEEE. Trans. on Electron Devices, vol. ED-32, No. 2. Feb. 1985

[14] SIS: A System for Sequential Circuit Synthesis", Report M92/41, University of California, Berkeley, May, 1992

[15] K.-H. Tsai, R. Tompson, J. Rajski, and M. Marek-Sadowska, "STAR-ATPG: a high speed test pattern generator for large scan designs", Proceedings of International Test Conference 1999, pp. 1021-1030

[16] L. Wu et al, "Glacier: A Hot Carrier Gate Level Circuit Characterization and Simulation System for VLSI Design", Intl. Symposium on Quality Electronic Design, pp. 73-79, 2000

[17] BTA Technology, http://www.btat.com

[18] S. Yamashita, H. Sawada and A. Nagoya, "SPFD: A new method to express functional flexibility", IEEE Trans. of Computer -Aided Design of Integrated Circuits and Systems, Aug, 2000

[19] P. Yang, and J.-H. Chern, **"**Design for reliability: the major challenge for VLSI", Proceedings of the IEEE, vol.81, (no.5), May 1993. p.730-44

[20] H. Yonezawa et al, "Ratio Based Hot-Carrier Degradation Modeling for Aged Timing Simulation of Millions of Transistors Digital Circuits", Intl. Electron Devices Meeting, pp. 93-96, 1998