RC24324 (W0707-163) July 31, 2007
Electrical Engineering

IBM Research Report

Timing-aware Power Minimization via Extended Timing Graph Methods

Haifeng Qian, Emrah Acar
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

Research Division
Almaden - Austin - Beijing - Cambridge - Haifa - India - T. J. Watson - Tokyo - Zurich

Timing-aware Power Minimization via Extended Timing Graph Methods

Haifeng Qian and Emrah Acar

***Abstract*— Power is an increasingly important performance metric, and must be considered during various design stages. With the advancement of multiple threshold devices, leakage power can be better controlled, utilizing fast and high-leakage devices just for critical paths, while low-leakage devices are used for non-critical parts to minimize power. In this paper, a practical timing graph-based algorithm is proposed to perform concurrent discrete optimization (V_t-assignment, device width biasing, device length biasing, etc) to minimize the power consumption, especially leakage, of a circuit subject to timing performance constraints. Our algorithm honors important constraints that are common to an industrial design methodology, including hierarchy, structural connectivity and layout-related rules. We demonstrate the performance of the algorithm in an industrial design automation platform consisting of an incremental transistor-level timing analysis engine and optimization environment.**

***Index Terms*—Circuit tuning, discrete optimization, power minimization, threshold voltage.**

I. Introduction

ALTHOUGH timing performance is still the primary target for IC manufacturers, power is recognized as the real performance limiter with current integration technologies. As devices shrink, they run faster at the expense of excessive leakage current. This poses itself as a tax on the performance, as the leakage power is becoming the dominant portion of the total power of a circuit [8]. Hence power optimization must be performed at various stages of the design flow.

We present a circuit-level optimization procedure, specifically by manipulating threshold voltage assignments, transistor widths, gate channel lengths and other parameter modifications that allow a design meet power and timing requirements. This is different than continuous optimization, where some design parameters can be chosen freely on an analog scale.

Manuscript received February 2, 2007.

Haifeng Qian and Emrah Acar are with IBM Research, Yorktown Heights, NY 10598 USA (phone: 914-945-2102; e-mail: qianhaifeng@us.ibm.com, emrah@us.ibm.com).

Circuit optimization research used to be concerned with automatic tuning the circuit solely for timing performance, and the transistor width used to be the primary variable to be continuously tuned [3][4][5][13]. Established transistor sizing methodologies can be roughly divided into two categories. The first is sensitivity-based discrete heuristics represented by TILOS [5][13]. The transistor sizing operation is discretized, for example, a downsizing or upsizing operation can be defined as scaling the width of a transistor by a constant factor; the solution is optimal under certain simplistic delay model, but suboptimal in general. The second is continuous nonlinear optimization, followed by snapping transistor sizes to the technology-imposed values [3]. Because its solution in the continuous domain is optimal and transistor widths are near-continuous variables, the quality is typically superior to discrete heuristics, at the cost of high runtime.

With the advent of multiple-V_t devices, voltage islands and multiple oxide thickness, the list of candidate variables is now expanded by a number of discrete operations, e.g. [6], which often offer better power-delay tradeoffs than transistor sizing. Most prominently, because sub-threshold leakage current is an exponential function of V_t [17], changing certain gates to higher V_t has the potential to achieve more power savings than gate sizing alone with the same timing budget. Recently gate-length is also suggested as a new design parameter, where it can be selected from a variety of alternatives as opposed to a single design length [18].

Since the optimal solution is most likely achieved by the simultaneous consideration of all design options, several methods have been proposed to incorporate multiple operations [10][11][12][14][15][16].

It is not trivial to incorporate discrete variables into a continuous sizing methodology such as [3], and the cost is often suboptimality and significant runtime increase. Hence most of these multi-variable methods are heuristics, and are often based on sensitivities defined similar to [5].

Here in this work, the proposed method is also a sensitivity-based one that performs *discrete optimization*. We discretize the operations just like [5], however the sensitivity is defined in an inverse way: we start with a

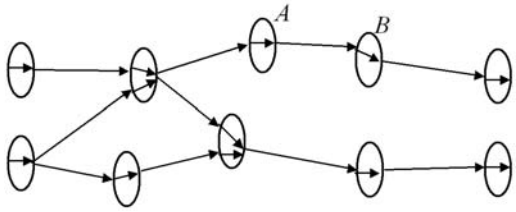

Fig. 1. CCC timing graph.

design that meets the timing requirement, and minimize its power without creating a timing violation. To reduce the greediness and avoid the lack of global view in [5][17], a graph-based iterative approach is used to look for an optimal set of transistors to modify in each iteration. This is similar in principle to [2] and [16] (V_t assignment only), but there are differences that make our method more likely to find a near-optimal solution and with less computational complexity. More importantly, the proposed approach is a general framework of truly simultaneous considering multiple discrete design choices: the set of chosen changes in each iteration is a mix of, for example, sizing operations and V_t assignments.

This technique is general enough to support a variety of design options, e.g., oxide thickness assignment and Vdd selection. Furthermore, emphasis is placed on handling constraints in industrial designs, which may originate from high-level issues, structural integrity of the design data, or layout related issues.

II. Problem Statement

In this section, we define the power optimization problem, and discuss various practical constraints.

A. Definitions and Assumptions

We assume that a static timing analysis engine is available. We will assume it supports transistor-level netlists by characterizing individual devices in clusters with proper logic functions. Figure 1 shows a timing graph used in the timing engine, where the oval nodes correspond to channel-connected components (CCC) (i.e., sets of transistors that are source-drain connected, and roughly correspond to gates), and the directed edges correspond to timing propagation segments, each of which represents the delay of a CCC or the interconnect delay of a net.

The delay and slew values are obtained by simulation in our implementation environment, but they can also be computed by analytical equations and/or table lookup, and the proposed algorithm in this paper is independent of this choice of technique. Directed paths in the timing graph represent data paths within one clock cycle. In other words, the start and end nodes are primary inputs/outputs and latches; in the case of designs using transparent latches, the start and end nodes are those with the same clock phase. Timing analysis provides the arrival time (AT) at each node in the graph, and given a timing target, the required arrival time (RAT) at each node; the slack at a node is defined as RAT-AT, and the slack at the output pin of a CCC is referred to as the timing slack of this CCC.

For simplicity of presentation, in Sections II and III, we will work with two types of variables: transistor width and V_t -level. The optimization variables are represented by the following vector

$$\mathbf{x} = [w_1, v_1, w_2, v_2 \ldots w_N, v_N]^T$$

where w's are continuous transistor width variables, and v's are discrete variables representing choices of design parameters in an countable feasible set. For simplicity v's can be thought of V_t -levels for devices. The operation can be performed in three granularity levels:

- CCC level: All transistors in a CCC must be scaled together, and must share the same discrete parameter (e.g. V_t). Then N is the number of CCCs in the circuit.
- Ptree/Ntree level: PMOS transistors in a CCC must be scaled together and must have the same discrete parameter, while NMOS transistors in a CCC must be scaled together and must share the same discrete parameter. For each CCC, there is a (w,v) pair for the PMOS transistors and another pair for the NMOS transistors. Hence N is twice the number of CCCs in the circuit.
- Transistor level: Every single transistor can be modified independently, and needs a (w,v) pair. Hence N is the total number of transistors in the circuit.

In this paper, we only consider application of power-saving techniques. Namely, we will select transistor width downsizing and high-V_t insertion. We define a move as a single change in an entry of the vector **x**. If the move affects a single design object, we refer to as a single move, and if it impacts multiple objects due to a design/netlist constraint, it will be referred to as a group move. In our experiments, we used 65nm SOI technology with such opportunities to mitigate leakage power.

We assume that there is a delay engine available to compute ΔD, the delay impact of any single move. Because at any of three granularity levels, a single move can only modify one CCC, ΔD is defined as the internal delay increase of this CCC, plus the delay increase of the net it drives. Since we deal with moves that don't affect input device capacitances significantly, the delay impact on the previous stage is very small and can be neglected. However if really desired, the impact can also be calculated and modeled in ΔD at a cost of extra runtime. ΔD can be obtained by simulation, analytical equations or table lookup, and again the proposed algorithm is independent of

this choice of technique. The only assumption about ΔD is that it is nonnegative for any move. Note that ΔD is computed assuming that only one entry of $\mathbf{x}$ is modified, and it is a function of both the move and the current status $\mathbf{x}$, and hence must be recomputed for a different $\mathbf{x}$. However, it could be computed incrementally when the specific CCC is modified, and can be tagged along with the delay calculation in an incremental static timing analysis engine.

We assume that there is an analysis engine available to compute $P(\mathbf{x})$, the power consumption of the design specified by $\mathbf{x}$. Depending on the application, $P(\mathbf{x})$ can be leakage alone, or the sum of leakage, dynamic, and short-circuit power, or more generally, a weighted sum of all three power components. The proposed algorithm is independent of this choice of object function. The power impact of a single move can be defined as $\Delta P = P(\mathbf{x})-P(\mathbf{x'})$, where $\mathbf{x'}$ differs from $\mathbf{x}$ by only one entry.

Like delay, power consumption can be computed by various methods, and the proposed algorithm is independent of this choice of technique. Like ΔD, the power impact ΔP can be also calculated incrementally.

B. Problem Statement

Let $S(\mathbf{x})$ be the minimum of all CCC timing slacks in the design solution specified by $\mathbf{x}$. Let S_{target} be a given constant that represents the desired timing target. The power optimization problem can be stated as follows.

Given an initial solution $\mathbf{x_1}$ *such that* $S(\mathbf{x_1}) > S_{\text{target}}$, *given the timing analysis and power estimation engines that can compute* ΔD *and* ΔP *of any single move for any solution vector, find* $\mathbf{x_2}$ *such that:*

- $P(\mathbf{x_2})$ *is minimized.*
- *The w entries of* $\mathbf{x_2}$ *are equal or less than the corresponding entries of* $\mathbf{x_1}$, *the v entries of* $\mathbf{x_2}$ *are equal or greater than the corresponding entries of* $\mathbf{x_1}$.
- $S(\mathbf{x_1}) > S_{\text{target}}$.

In the above problem statement, we only assume the availability of ΔD and ΔP of single moves, and do not make any assumption about the joint effect of multiple moves. For example, if a move is applied to CCC-A and another move is applied to CCC-B in Figure 1, the resulting slacks of both CCCs may or may not be the original slacks deducted by $\Delta D_A + \Delta D_B$, depending on the accuracy of the timer engine. Such an additive assumption is often needed by existing methods such as [11][12], but our method does not need it.

C. Additional Constraints

Many research efforts including ones referenced above have targeted at leakage power optimization. However most of the previous works assume that applicable design changes are for individual design objects (gates, transistors, NMOS/PMOS stacks) and can be applied without restrictions. However, these assumptions are often not valid in real-life design environments.

A typical macro design for a high-performance integrated circuit consists of logic synthesis, layout design, electrical and physical verification on multiple databases with parallel representations of the design. It is not easy to modify the netlist locally without propagating the changes to other representations. And if the original design is in a hierarchical data model, the modified design often needs to maintain a similar hierarchy, which maps any local change to multiple changes throughout the design. In other words, there are limitations from the design methodology, which require multiple objects to be considered together. We will denote this requirement as *grouping*. For instance, if the hierarchy of the original design is desired to be kept the same, all instances of a particular cell in the design need to be grouped together: if any instance of a particular cell is to be modified, its other instances should follow the same modification. Hence, in evaluating the impact of modifying an instance, we need to analyze the joint effects of changing all the instances of the same type used in the netlist. Similar limitations can be layout-based. For example, two cells need to be in the same V_t class if they are spatially close, and their V_t moves are grouped together. The concept of grouping may also be used to represent other classes of constraints.

Another constraint is that certain circuit elements that are not intended for any modifications: typically clock circuitry, clock gating, and power gating modules. This requires an additional *no-touch* designation for such elements. If a no-touch-designated object is also inside a group, we will propagate the no-touch designation to all other items in the group. In other words, these constraints can be used in association rules.

In summary, the following constraints are not included in the problem statement in the previous subsection, but are present in a custom-design environment and need to be handled.

- Slew constraint: timing constraints are also defined on slews at the output pin(s) of each CCC. The circuit optimization procedure should not cause any slew violation.
- No-touch designation: design changes are prohibited for certain CCCs.
- Grouping: certain objects must be identically modified for reasons not limited to preserve higher level representations, hierarchy etc. Hence these objects share the same $\mathbf{w}$ and $\mathbf{v}$.

Next, we first describe a graph-based algorithm for the

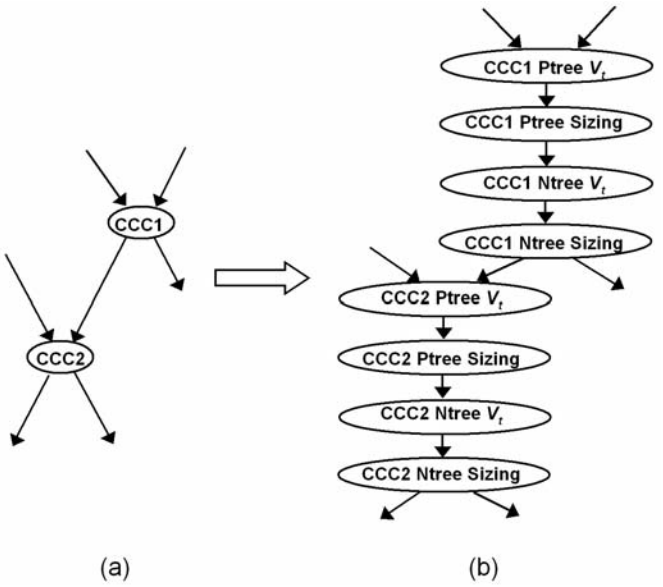

Fig. 2. Conversion from a timing graph to a move graph.

problem stated, and then how to incorporate these constraints.

III. Proposed Method

The proposed algorithm is an iterative procedure, where in each iteration, a set of moves are chosen and applied to the design, such that the power saving is maximized in a certain way, and that no two modified objects share a data path in a given iteration. The description is divided into three subsections: the first converts a timing graph into a move graph, the second formulates one iteration of the procedure as a weighted maximum independent set (WMIS) problem, and the third incorporates specific design constraints.

As the first step of the proposed method, we construct a directed graph where each node corresponds to a single move defined in Section 2, based on the original CCC timing graph and available design operations.

Let us use the example in Figure 2 to explain this graph conversion. Figure 2(a) shows two CCCs in the timing graph, one driving the other. Note that the graph is already different from Figure 1: timing propagation edges inside CCCs are dropped. The remaining edges carry connectivity information, i.e., if there exists a directed path between two CCCs, then they must share a data path in the design, and it is possible that the ΔD of a move on one of the two can affect the timing slack of the other.

Figure 2(b) shows the corresponding structure in the converted move graph. In this example, the algorithm works in the Ptree/Ntree mode, and there are four possible operations on a CCC: changing all PMOS transistors to higher V_t, changing all NMOS transistors to higher V_t, downsizing all PMOS transistors with a constant factor, downsizing all NMOS transistors with a constant factor. Each node in the timing graph is converted to a string of nodes in the move graph, where each move node represents a single change on the associated CCC, and hence has its own ΔD and ΔP pair.

Obviously, if the algorithm works in the CCC mode, each CCC in the timing graph would be converted into a list of two nodes, the sizing move and the V_t move; if the algorithm works in the transistor mode, a CCC with m transistors would be converted into a string of $2m$ nodes: two for each transistor. In fact, the above conversion can be more flexible.

Any design change on a CCC can become a move node and join the string, as long as it is a discrete operation: for example, choices among multiple supply voltages, choices among multiple oxide thickness, and so on, can all be incorporated. To achieve a faster convergence, the string can also include nodes that represent combined moves: for example, a node that represents higher V_t selection plus downsizing of PMOS transistors can be added to Figure 2. The size of the move graph increases linearly with the number of candidate design changes to be considered for a CCC, and this makes the proposed algorithm scalable.

It is easy to verify that the following property is maintained during the graph conversion: if there exists a directed path between two CCCs in the timing graph, there also exists a directed path between any pair of their move nodes. This ensures the correctness of the proposed heuristic, and will be explained in the next subsection.

Finally, the graph conversion only needs to be performed once during the initialization of the algorithm, and the move graph can be used repeatedly, while ΔP and ΔD values of a move node should be updated when the design is modified.

If a move becomes infeasible (for example, a V_t move becomes infeasible when a transistor is already using the highest threshold voltage), the ΔP and ΔD values of that node are simply set to zero so that it will not be selected by the procedure described in the next subsection.

A. Weighted Maximum Independent Set

Let us start by citing definitions from the graph theory. The *maximum independent set* (MIS) of a graph is defined as the largest set of nodes such that no two nodes in this set are adjacent.

The *weighted maximum independent set* (WMIS) of a node-weighted graph is defined as the set of nodes with the maximum weight sum such that no two nodes in this set are adjacent. The *transitive closure* of an directed graph G_1 is another directed graph G_2 constructed by adding directed edges to G_1, such that there exists an edge from node a to node b in G_2, if and only if there exists an directed path from a to b in G_1. The *transitive reduction* of G_1 is defined as the smallest directed graph such that its transitive closure is also G_2. For example, Figure 3 is the transitive closure of Figure 1. (ignoring the edges inside CCCs), and Figure 1 is the *transitive reduction* of Figure 3. In general, however, if G_2 is the transitive closure of G_1, G_1 is not necessarily the transitive reduction of G_2. In that case, we will refer to G_1 as a *relaxed transitive reduction* of G_2 in the following discussion.

Now suppose the move graph from the previous subsection is graph G_1, and we construct its transitive closure G_2 and then find the WMIS of G_2, based on the following node weight definition:

$$\text{node weight} = \begin{cases} \dfrac{\Delta P}{\Delta D} & \text{if} \quad \Delta D < S_{\text{CCC}} - S_{\text{target}} \\ 0 & \text{if} \quad \Delta D \geq S_{\text{CCC}} - S_{\text{target}} \end{cases}$$

where ΔD, ΔP are as defined for this move node, S_{target} is as defined in the problem statement, and S_{CCC} is measured for the move node in the timing graph. Such weight definition for a move node is a sensitivity function similar to those in [1][5][14] and zero if the move is infeasible under the timing constraint. Intuitively, this serves as the merit function of a node, and the proposed algorithm tries to maximize the total value of the chosen moves.

The WMIS of the transitive closure graph G_2 therefore corresponds to the node set with the maximum weight sum in the move graph G_1 such that there does not exist a directed path between any two nodes in this set. From the previous subsection, if there is no path between two move nodes, the two affected CCCs do not share a data path in the design. This means that if the design changes represented by the WMIS are applied on the design, each modified CCC is independent from others in terms of timing, and is only affected by its own ΔD. According to the node weight definition, any move with a positive weight is feasible, and therefore applying these changes will not cause a timing violation. The above forms the basic framework of the proposed algorithm.

Now the power optimization problem can be converted into an iterative process, where each iteration finds the WMIS of the transitive closure of the move graph and applies the design changes specified by the WMIS.

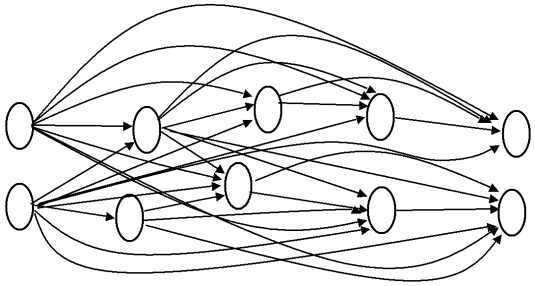

Fig. 3. The transitive closure of the timing graph in Figure 1.

This usage of WMIS is similar to the methodology in [2] for dual-supply voltages optimization, but the following differentiates our method from theirs. For a large timing graph, finding its transitive closure graph can be expensive in terms of both time and space complexity. Fortunately, for acyclic graphs, there exists a correspondence between the WMIS of the transitive closure and the *min flow* of the relaxed transitive reduction graphs. The *min flow* problem considers a directed graph with a set of source nodes and sink nodes; for a non-source non-sink node, the sum of its in-flows must be equal to its out-flows; each node has a non-negative minimum capacity, and the problem is to find the minimum flow that can be routed from source nodes to sink nodes such that the flow through each node is at least its assigned minimum capacity. The min flow problem can be mapped to the commonly known max flow problem, and hence be solved in polynomial time [7]. If the minimum capacity of any node is assigned to be equal to the node weight, for an acyclic directed graph G_1 and its transitive closure G_2, the sum of node weights of the WMIS of G_2 is equal to the min flow of G_2 [9]; then because of the fact that the min flow *of* G_2 *is* equal to the min flow of any of its relaxed transitive reduction graphs (proof omitted), the min flow of G_1 is also equal to the sum of node weights of the WMIS of G_2. Therefore, unlike [2], we do not build the transitive closure G_2 explicitly, and only work on the original relaxed transitive reduction G_1 itself: find its min flow first, and then techniques are available to identify the WMIS of the transitive closure [9]. Another similar methodology can be found in [16] for V_t assignment only, which also modifies a set of independent CCCs in each iteration. But instead of WMIS, [16] used a heuristic based on topological levels of gates. Since our method works on a much larger search space, it is more likely to approach the optimal solution.

The above discussion assumes the move graph to be acyclic. This is true for the majority of designs where the CCC timing graph is acyclic to start with. However the CCC timing graph of a design is not necessarily acyclic in

general, and for those timing graphs that are not, they need to be made acyclic by ignoring certain timing propagation segments. Typically only a few edges need to be dropped, and to minimize the incurred accuracy impact, the max-slack edge of each cycle is chosen in our implementation.

Admittedly, this iterative algorithm based on finding WMIS in each iteration is a sequence of local optimizations that does not guarantee optimality of the overall solution to the optimization problem. However, it has the capability to handle multiple design options and various practical constraints, and we will validate its performance on real-life designs in the next section.

B. Handling Constraints

No-touch constraints are handled by forcing zero sensitivity, i.e., zero weight on the corresponding graph nodes. Slew constraints and the grouping constraints need a closer look.

A grouping constraint dictates that the group members must be identically modified. For example, in Figure 2, if the PMOS transistors in CCC1 and the PMOS transistors in CCC2 are grouped, the "CCC1 Ptree V_t" move and the "CCC1 Ptree V_t" move must happen together, and the "CCC1 Ptree Sizing" move and the "CCC1 Ptree Sizing" move must happen together. To honor such constraints, a preprocessing step is added to merge the nodes in the move graph. This is done by merging them into a single node, which represents a group of moves, and is referred to as a *group node.* Such merger operations convert the original move graph that contains only single move nodes, to a new move graph that contains the remaining independent single moves nodes plus the newly created group nodes, and this new mixed graph will be used in the iterative algorithm in the previous subsection. During the merging operations, the following three issues must be resolved:

- If a member in a group is designated as no-touch, all other members in the group will be designated as no-touch. Hence we process all group members as described earlier.
- After the merger operations, the resulting new graph must remain acyclic, and the correlation among the move nodes must be maintained. In other words, if there exists a directed path from move node *a* to move node *b* in the original move graph before mergers, there must still exist a directed path from the representative of *a*, either *a* itself or a group node that contains *a*, to the representative of *b*. Figure 4 and 5 illustrate the merger operations that satisfy these requirements: Figure 4 shows the simple case of merging two nodes that do not share a path, and Figure 5 shows merging nodes A and B when B is inside the fan-out cone of A. Nodes C and D in Figure 5 represent the first set of nodes that are outside the fan-out cone of A encountered during an upstream traversal from B; new edges are added from them to the merged node AB; a blank node replaces node B after the merger, to maintain connectivity among B's neighboring nodes. The merging operation is similar for the case when B is in the fan-in cone of A. A group with more than two nodes can be merged by repeating the above operations. The worst-case complexity of merging all groups is quadratic with respect to the number of nodes in the move graph, and this is a one-time overhead at the initialization stage of the algorithm.

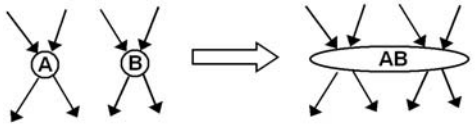

Fig. 4. Merging two independent nodes.

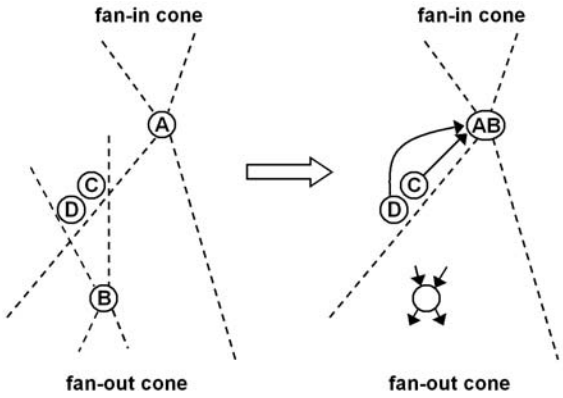

Fig. 5. Merging two dependent nodes.

- There are many ways to define the weight, i.e., sensitivity, of a group node. In our implementation, we still use $\Delta P_{\text{group}}/\Delta D_{\text{group}}$. The ΔP_{group} of a group node is defined as the sum of ΔP values of the original single move nodes. In order to evaluate ΔD_{group}, suppose the original move nodes are *1,2,…k,* we construct a new directed graph *G* composed of these nodes, such that there exists a directed path from node *i* to node *j* in *G* if and only if there exists a directed path from *i* to *j* in the original move graph. *G* is not unique, and is acyclic if the original move graph is acyclic. For a directed path *Y* in *G*, define its length as $L_Y = \sum_{i \in Y} \Delta D_i$. For node *j*, define $\Delta D_j = \max_{Y \text{ passses } j} L_Y$ and then for the group node define $\Delta D_{\text{group}} = \max_j \Delta D_j$. The group node is considered infeasible and has zero sensitivity if for any node *j*, ΔD_j is greater than S_{CCC}-S_{target}. Such sensitivity calculation and feasibility check can

be done in linear runtime with respect to the group size.

Honoring the slew constraints are done both in the ΔD calculation and in the post-validation of the applied moves. During the calculation of ΔD, we also monitor output slews and penalize any slew violation due to a suggested move by increasing the ΔD by a large number. This would force the algorithm to skip such a move with a low weight. If for some reason, the move is still picked up by the procedure and applied on the design, the post-validation step of static timing analysis based on the modified circuit may demonstrate slew violations. When a slew violation is detected after one iteration, the move that causes this violation is identified based on timing path information, and will be revoked, i.e. it will be reverted back to its original status.

We also generalized our weighting scheme to include slack-criticality of the design [19]. Based on the current timing slacks and the targets we can propagate a criticality metric within the circuit and incorporate it in the weights for each move node. By now, we have collected the necessary pieces of the proposed algorithm, which is summarized in the pseudo-code below:

```
Build acyclic timing graph;
Convert timing graph to move graph;
Merge grouped move nodes into group nodes;
Propagate no-touch designations;
Loop until convergence:
    Update node weight = sensitivity;
    Find WMIS;
    Apply design changes;
    Revoke moves if timing violation;
```

IV. Results

In the experiments, we use a number of circuit macros from a high-performance microprocessor core implemented in a state-of-the-art technology (65nm, SOI). These macros are implemented by mostly regular-V_t devices, while some critical-path elements may have low-V_t devices. The macro sizes range from hundreds of devices to tens of thousands of devices. The timing performance of the macros is satisfactory, but the power performance, especially leakage, needs to be greatly improved to meet the power specifications of the end product.

A. Experimenting Different Modes

Table I shows the first experiment where three circuits are used to evaluate the performance of high-V_t assignment in CCC, Ptree/Ntree and Transistor modes. The initial timing slack is preserved, i.e., the initial and final timing slacks are the same in each run. No grouping constraint is imposed.

The transistor mode always gives the largest leakage saving, at the expense of more runtime: 3-4 times of that of the CCC-level. Transistor mode also impacts the rest of the design tasks (layout generation, transistor back-annotation, library support). The ptree/ntree mode brings intermediate results and could be used whenever the cell libraries can support it. If used in a design flow, each cell need to be provided in not only all-V_t modes but also in hybrid combinations (nfets and pfets may have different V_t 's). Hence this typically results in a larger number of cells needed to be supported by the cell library. We believe that this is more likely to gain more traction in practical design flows compared to fine grain transistor-level optimization requirements. Looking at the CCC mode, we see that algorithm achieve reduction in bulk of the leakage and rather keep the following design automation tasks as is with very little post-optimization impact. The post-optimization tasks often remain the same since we replace a particular CCC with a different version of V_t. This is similar to gate-level optimization in many aspects. For all other experiments, we will use the CCC mode, as it is the most efficient and often preferred due to their simplicity/practicality of post-processing and back-annotation tasks following this optimization.

TABLE I
LEAKAGE REDUCTION BY HIGH-V_T ASSIGNMENT ONLY

Circuit	Initial Leakage	Mode	Final Leakage	Reduction
#1	1.49mW	CCC	0.787mW	47.22%
		Ptree/Ntree	0.484mW	67.54%
		Transistor	0.453mW	69.92%
#2	11.6mW	CCC	9.19mW	20.81%
		Ptree/Ntree	8.67mW	25.30%
		Transistor	8.58mW	26.05%
#3	6.02mW	CCC	5.48mW	8.92%
		Ptree/Ntree	5.46mW	9.25%

B. High-V_t and Width Downsizing with the Grouping Contstraints

Table II shows the second experiment with a set of examples with tight power performance specifications. We compare high-V_t assignment only and simultaneous high-V_t assignment and transistor downsizing, both in CCC mode. Each downsizing move uses a scaling factor of 0.8. No grouping constraint is imposed, except for the results in the last column. By high-V_t assignment only, the dynamic power is also reduced by a few percent, since the capacitance of the higher V_t devices are less than the regular ones. The overall runtime of high-V_t assignment (including the initial timing) is only 2-4 times that of a complete static timing analysis. Such efficiency is achieved by using an incremental timing analysis engine and semi-analytical hardware-based power models that don't require excessive simulation.

The runtime of simultaneous high-V_t assignment and transistor downsizing, however, is 50-200 times that of a complete static timing analysis; on the other hand, more power reduction is achieved with such extra effort. The only exception is benchmark 7, where the total power reduction by high-V_t plus downsizing is slightly less than that by high-V_t assignment alone: in this case, the local optimization of an early iteration leads to a less satisfactory final solution.

it results in smaller widths for transistors located in non-critical paths. Hence the leakage and dynamic power could be reduced without affecting the critical paths and timing requirements. The optimizer operates on delay sensitivities of each CCC with respect to device widths. For each circuit, three different runs are performed: the first with the formal optimization tool that carries out transistor sizing only, the second with the proposed method in high-V_t assignment only, and then a cascade run in which the proposed high-V_t assignment procedure is called after the formal optimization tool. In these three runs, we preserve the minimum slack of the circuit, and honor all the constraints required from hierarchy and layout. The results suggest that transistor-width optimization alone to minimize area is inferior in terms of leakage power to our approach where leakage is targeted for the optimization, and that both methods complement each other and produce the best results when used together.

TABLE II
POWER REDUCTIONS BY HIGH-V_T, BY SIMULTANEOUS HIGH-V_T AND WIDTH DOWNSIZING AND HIGH-V_T UNDER GROUPING CONSTRAINTS

Circuit	Initial leakage power	Initial dynamic power	After high-V_t		After high-V_t and width downsizing		After high-V_t with grouping
			Leakage reduction	Total power reduction	Leakage reduction	Total power reduction	Leakage reduction
#4	1.06mW	1.24mW	69%	33.9%	69%	60.7%	32.6%
#5	0.20mW	0.25mW	33.8%	16%	42.1%	33.64%	12.3%
#6	0.99mW	1.23mW	40.93%	19.38%	45.6%	37.9%	9.1%
#7	1.84mW	2.23mW	28.35%	13.66%	14.6%	11.1%	10.4%
#8	5.01mW	5.93mW	22.29%	10.81%	31.8%	27.0%	8.2%
#9	4.67mW	5.67mW	15.59%	7.46%	20.9%	16.47%	7.3%
#10	4.93mW	5.96mW	25.74%	12.37%	32.16%	26.37%	9.1%
#11	8.66mW	10.6mW	40.71%	19.46%	35.75%	29.44%	11.5%
#12	3.69mW	4.51mW	22.20%	10.59%	36.47%	29.12%	6.2%
#13	5.17mW	6.30mW	36.77%	17.59%	36.42%	30.06%	20.1%

TABLE III
POWER REDUCTION BY NEAR-OPTIMAL MIN-AREA TRANSISTOR SIZING, HIGH V_T ASSIGNMENT, AND BY THE SEQUENTIAL RUN OF THE TWO

Circuit	Width downsizing		High-V_t		Both high-V_t and widthdownsizing	
	Leakage power reduction	Total power reduction	Leakage power reduction	Total power reduction	Leakage power reduction	Total power reduction
#14	26.87%	25.98%	26.82%	16.05%	41.09%	34.38%
#15	19.57%	20.06%	35.95%	20.24%	45.65%	34.78%
#16	8.88%	8.91%	19.22%	11.64%	25.58%	19.01%
#17	11.05%	10.98%	11.82%	6.64%	22.44%	17.4%
#18	7.84%	6.69%	10.21%	4.96%	14.78%	10.09%

The last column of Table II shows the impact of grouping constraints. The designs typically contain several levels of hierarchy, and grouping constraints are added to guarantee the structural connectivity of the post-optimization circuit. Clearly the additional constraints resulted in an unfavorable power reduction, but these are the results of necessities and requirements of a real-life design flow.

C. High-Vt compared to a Formal Circuit Tuner

In Table III, we use the proposed method in conjunction with a formal circuit optimization tool [3] that yields almost optimal transistor sizes for area minimization under timing and other constraints. The optimizer works on the objective of reducing area based on timing constraints and

D. Joint High-V_t and Device Length Biasing

As the last experiment, we evaluate the performance of the proposed method with multiple types of moves. As mentioned before, high-V_t assignment and longer device length may be used to mitigate leakage of devices located in non-critical paths. To compare the benefits of these techniques when applied individually and jointly, we perform the proposed algorithm with three different schemes with identical timing and other constraints: CCC-level high-V_t assignment only; simultaneous CCC-level high-V_t assignment and CCC-level channel length biasing; CCC-level high-V_t assignment followed by transistor-level channel length biasing. For these experiments, we allowed the device length to increase by 2nm or 4nm from the nominal value.

The second scheme in this experiment models the effect of design-time channel-length optimization, since the

TABLE IV
POWER REDUCTION BY CHANNEL LENGTH BIASING

Circuit	CCC-High-V_t		Joint CCC-level High-V_t and Channel-length		CCC-High-V_t followed by transistor-level Channel-length	
	Leakage power reduction	Total power reduction	Leakage power reduction	Total power reduction	Leakage power reduction	Total power reduction
#19	57.87%	50.28%	64.05%	54.91%	63.46%	54.74%
#20	20.9%	18.24%	25.56%	21.96%	24.74%	21.32%
#21	44.78%	38.94%	52.79%	45.29%	52.82%	45.32%
#22	40.67%	35.06%	46.65%	39.7%	45.03%	39.1%

changes perform on each CCC and can be followed by post-optimization tasks. The third scheme models the effect of post-layout channel-length optimization, since transistor-level optimization results can be applied directly to the final layout at the very end of the design flow.

The results in Table IV suggest that the additional flexibility of channel length biasing results in a 5-8% further leakage reduction compared to only using High-V_t by varying doping concentration. For these testcases, which are based on 65nm technology, the primary reason for these extra power savings is that channel length biasing is a more fine-grain operation than a V_t change, and therefore help to reach better a power-delay trade-off point in the solution space. We expect the device length biasing would yield more benefits for 45nm technology.

V. CONCLUSION

We proposed a graph-based optimization method to perform power minimization in transistor-level circuit designs. Our approach performs power-reduction actions on the circuit and results in low power circuits with given timing constraints. We also honor constraints that are typical in real-life production lines. This method is implemented in an industrial tool that is incorporated into an industrial design flow and currently yields reliable results.

The proposed method is based on an extended timing to evaluate the power reduction actions. The proposed method evaluates the impact of each action on power and timing, and formulates a maximum independent set problem. Although not performed explicitly, the method considers delay budgeting and evaluates only feasible moves that are allowed in a given path. The iterative nature of the algorithm results in high-quality results.

We emphasized an important aspect of real-life design environments: structural constraints that must be honored for physical and methodological reasons. These issues are solved by the concept of grouping which lets the annotation of the suggested changes with minimum effort. Besides the operations of width sizing, channel length biasing and V_t assignment of which experimental results are presented in this paper, the suggested approach can also be used with other power minimizing actions, including oxide thickness assignment, Vdd island selection, etc. Any discrete actions that impact the design in a local manner can be incorporated in the proposed framework.

REFERENCES

[1] Z. Chen, M. Johnson, L. Wei and K. Roy, "Estimation of standby leakage power in CMOS circuit considering accurate modeling of transistor stacks," *Proceedings of International Symposium on Low Power Electronics and Design*, pp. 239--244, 1998.

[2] C. Chen, A. Srivastava and M. Sarrafzadeh, "On gate level power optimization using dual-supply voltages," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 9, no. 5, pp. 616--629, 2001.

[3] A. R. Conn, I. M. Elfadel, W. W. Molzen, P. R. O'Brien, P. N. Strenski, C. Visweswariah and C. B. Whan, "Gradient-based optimization of custom circuits using a static-timing formulation," *Proceedings of Design Automation Conference*, pp. 452--459, 1999.

[4] J. Darringer etal., "EDA in IBM: Past, Present and Future," *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, vol. 19, no. 12, pp. 1476--1497, 2000.

[5] J. P. Fishburn and A. E. Dunlop, "TILOS: a posynomial programming approach to transistor sizing," *International Conference on Computer-Aided Design Digest of Technical Papers*, pp. 326--328, 1985.

[6] K. Fujii, T. Douseki and M. Harada, "A sub-1V triple-threshold CMOS/SIMOX circuit for active power reduction," *International Solid-State Circuits Conference Digest of Technical Papers*, pp. 190--191, 1998.

[7] A. V. Goldberg and R. E. Tarjan, "A new approach to the maximum flow problem," *Proceedings of Symposium on Theory of Computing*, pp. 136--146, 1986.

[8] *International Technology Roadmap for Semiconductors*, Semiconductor Industry Association, 2004. Available at http://public.itrs.net

[9] D. Kagaris and S. Tragoudas, "Maximum independent sets on transitive graphs and their applications in testing and CAD," *International Conference on Computer-Aided Design Digest of Technical Papers*, pp. 736--740, 1997.

[10] M. Ketkar and S. S. Sapatnekar, "Standby power optimization via transistor sizing and dual threshold voltage assignment," *International Conference on Computer-Aided Design Digest of Technical Papers*, pp. 375--378, 2002.

[11] D. Nguyen, A. Davare, M. Orshansky, D. Chinnery, B. Thompson and K. Keutzer, "Minimization of dynamic and static power through joint assignment of threshold voltages and sizing optimization," *Proceedings of the International Symposium on Low Power Electronics and Design*, pp. 158--163, 2003.

[12] P. Pant, V. K. De and A. Chatterjee, "Simultaneous power supply, threshold voltage, and transistor size optimization for low-power operation of CMOS circuits," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 6, no. 4, pp. 538--545, 1998.

[13] J. M. Shyu, A. Sangiovanni-Vincentelli, J. P. Fishburn and A. E. Dunlop, "Optimization-Based Transistor Sizing," *IEEE Journal of Solid Slate Circuits*, vol. 23, no. 2, pp. 400--409, 1988.

[14] S. Sirichotiyakul, T. Edwards, C. Oh, R. Panda and D. Blaauw, "Duet: an accurate leakage estimation and optimization tool for dual-V_t circuits," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 10, no. 2, pp. 79--90, 2002.

[15] A. Srivastava, D. Sylvester and D. Blaauw, "Power minimization using simultaneous gate sizing, dual-Vdd and dual-Vth assignment," *Proceedings of Design Automation Conference*, pp. 783--786, 2004.

[16] Q. Wang and S. B. K. Vrudhula, "Static power optimization of deep submicron CMOS circuits for dual V_t Technology,"

International Conference on Computer-Aided Design Digest of Technical Papers, pp. 490--496, 1998.

[17] L. Wei, K. Roy and C. Koh, "Power minimization by simultaneous dual-Vth assignment and gate-sizing," *Proceedings of IEEE Custom Integrated Circuits Conference*, pp. 413--416, 2000.

[18] P. Gupta, A.B. Kahng, P. Sharma, and D. Sylvester, "Gate-length biasing for runtime leakage control," *IEEE Transactions on Computer-Aided Design*, pp. 1475-1485, August 2006.

[19] C. L. Berman, D. J. Hathaway, A. S. LaPaugh, L. H. Trevillyan, "Efficient techniques for timing correction," *IEEE International Symposium on Circuits and Systems*, pp. 415-419, 1990.