Gain-Based Technology Mapping for Minimum Runtime Leakage under Input Vector Uncertainty

Ashish Kumar Singh, Murari Mani, Ruchir Puri*, and Michael Orshansky
Department of Electrical and Computer Engineering, University of Texas at Austin
*IBM T.J. Watson Research Center, Yorktown Heights, NY

ABSTRACT

The gain-based technology mapping paradigm has been successfully employed for finding minimum delay and minimum area mappings. However, existing gain-based technology mappers fail to find circuits with minimal leakage power. In this paper, we introduce algorithms and modeling strategies that enable efficient gain-based technology mapping for minimum leakage power. The proposed algorithm is probability-aware and can rigorously take into account input state probability distribution to generate a circuit mapping with minimum leakage at a given percentile. Minimizing leakage at high percentiles is essential for minimizing peak leakage, which strongly influences the cooling limits and packaging costs.

The algorithms have been tested on the ISCAS85 benchmark suite. Results indicate that the mappings produced by the new algorithm consume, on average 14% lesser leakage power at the 99% percentile with 1% delay penalty when compared with the approaches used in previous gain-based mappers [2]. Also, compared to a dominant-state mapper, our approach produces mappings with 15% lesser mean value of leakage. The new algorithm also reduces leakage at high quantiles by 12.8% on average, compared to a dominant state leakage minimizing mapper and the maximum savings can be as high as 21.49% across the benchmarks. Compared to the bin based mapper [10], the runtime of the algorithm is 15X faster.

Categories and Subject Descriptors

B.6.3 [**Design Aids**]: *Automatic synthesis, Optimization*

General Terms: Algorithms, Performance.

Keywords: Leakage, Technology Mapping, Logical Effort.

1. INTRODUCTION

Reducing the leakage power at all levels of design is a known need. The explosive growth of static (leakage) power consumption is a crucial problem: at the 65nm node, leakage power accounts for 45% of total power of the circuit [11]. There is significant potential for leakage reduction in synthesis by employing technology-dependant optimization. Since leakage is highly dependent on the library gates, technology mapping (TM) which binds a circuit to a set of specific gates is an especially potent technique.

Technology mapping for minimum delay and area are well developed in the form of load-independent models [3][15][16], load-binning techniques [10] and gain-based strategies [4][2]. Gain-based TM was introduced in [4] and has been essential for solving the problem of timing closure [12]. Gain-based techniques are attractive because of their superior computational efficiency for rich libraries. Technology mapping for leakage has been investigated for the first time in [7]. However, the solution of [7] relies on a heuristic treatment of load dependency, which leads to sub-optimal results in the case of a continuously sized gate library. Furthermore, it is also likely to incur a high run-time penalty because of the need to discretize gates in such a library.

The problem of TM for leakage in a gain based setting has not been studied previously. The failure of existing gain-based algorithms which are based on implicit area minimization to find a mapping with minimum leakage necessitates the development of such a technique.

The first novel contribution of this paper is the development of the technology mapping algorithm for leakage minimization using an efficient gain-based setting. The second essential contribution is that the algorithm is formulated to find a mapping that minimizes leakage at an arbitrary quantile of its distribution. The leakage of a gate is highly sensitive to the value of the input vector: for single gates the difference between the maximum and minimum leakage can be up to 10X [13]. For larger circuits, the range is smaller but still remains substantial. While average (mean) power determines battery life, maximum (peak) power directly influences cooling

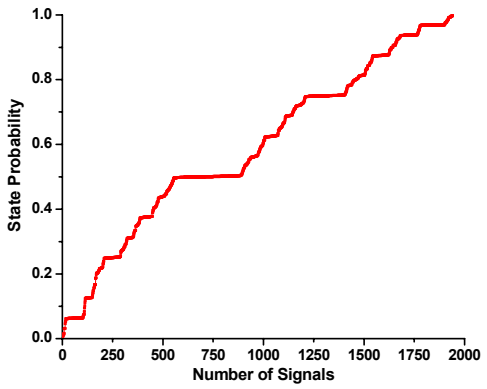

Figure 1. The distribution of state probabilities of the signals of the combinational circuit c3540. The distribution appears to be fairly uniform, except for a cluster around 0.5. The state probability of any primary output being 1 is set to be 0.5.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
DAC 2006, July 24–28, 2006, San Francisco, California, USA.
Copyright 2006 ACM 1-59593-381-6/06/0007…$5.00.

requirements and packaging costs. The cost of a cooling solution depends on the maximum power, and increases dramatically beyond a certain point: a 15% increase in power, can lead to a 3.5X increase in the cost of a cooling solution [14].

Both average and maximum runtime leakage power can be reduced by exploiting leakage state dependency, unequal state probabilities and leakage variances. It was observed in [8] that the assignment of transistors in a gate to higher Vth taking into account the probability of the gate being in a specific state can result in 62% reduction of power. Accordingly, for gates with a skewed state probability, only transistors leaking in that state will be set to high Vth, thereby minimizing leakage with minimal delay penalty. While the fundamental observation in [8] is correct, it is based on the simplifying assumption that the state probabilities are skewed to either 0 or 1, which enables a rather straightforward algorithm for runtime leakage minimization. Extensive experiments on general combinational circuits do not validate this assumption (Figure 1), thus requiring the development of a general algorithmic approach to input state probability-aware optimization. Such an approach is developed in this paper in the context of TM.

The paper is organized as follows. In Section 2, the reasons for the insufficiency of the existing gain-based strategies for finding a minimum leakage mapping are discussed. Section 3 introduces the new leakage gate models and describes the algorithm flow. Finally, the results are analyzed in Section 4.

2. Challenge of Gain-Based TM for Leakage

Gain-based TM (GB-TM) algorithms utilize logic-effort based delay models [1], in which delay is composed of two parts, a parasitic delay p and an effort delay which is directly proportional to the gain $g = C_{out} / C_{in}$. Here, C_{out} is the output load and C_{in} is the input capacitance. For a fixed value of gain, gate delays are fixed, and the minimum delay mapping can be easily performed. Several gain-based TM algorithms fundamentally rely on the monotonic relationship between area and delay which is enforced implicitly via the notion of gain. A simplified approach to target area-delay tradeoff using GB-TM is presented in [2]. A *pivotal gate* is selected and is assigned some *global gain value*. Then, the gains of all the other library gates are adjusted so that they have the same effort delay as the pivotal gate. Because of the monotonic one-to-one relationship between delay and area through gain, higher gain leads to a mapping with higher delay and lower area. Optimal area-delay Pareto curves are obtained by sweeping across the range of global gain values.

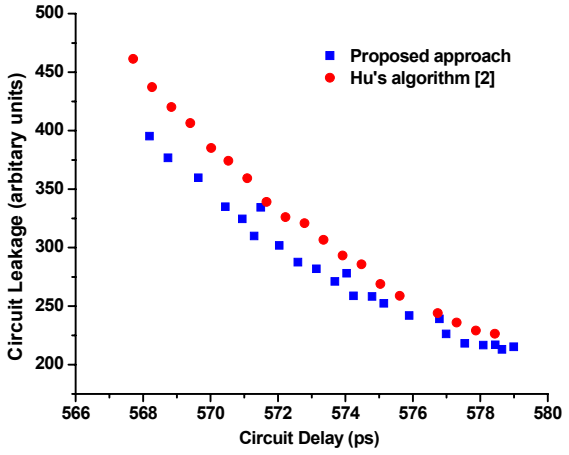

Figure 2. The algorithm in [2] is unable to find the leakage-optimal mappings even with a very fine-grained sweep of the global gain (2.11 to 2.30 in steps of 0.01) for C1908. Leakage savings of 4.8% to 9.6% are obtained using our approach with no delay penalty.

However, this technique fails in finding a minimum leakage mapping. The basic reason is that while there exists a one-to-one mapping between gain and area, the mapping between gain and leakage is one-to-many. For instance, all assignments of high/low threshold voltages to the transistors in a gate will be characterized by the same logical effort and gain. Consequently, a strategy that minimizes delay for a fixed gain in a rich library with mixed-Vth gates would always lead to a mapping with all gates set to low Vth and a mapping containing mixed-Vth gates would not be picked, even if its delay is only marginally larger. Because of the exponential delay-leakage tradeoff via Vth, the resulting leakage for a gate may be orders of magnitude higher than the minimum possible to achieve the delay target. Sweeping the global gain value in small steps does not help, since the min-delay and max-leakage solution is always identified. We have experimentally confirmed this hypothesis by finding circuit mappings that are uniformly better in leakage (Figure 2).

A solution to this problem within the gain-based TM framework is to propagate a set of competitive delay-leakage pairs for each gain value. Performing this efficiently without the knowledge of loads to match is a challenge. While relying on gain-based delay models eliminates the problem of sizing in delay estimation, to enable accurate estimation, we propose a linear model of leakage in terms of the output capacitance of a match. This result is a key facilitator in capturing the dependence of leakage on the output capacitance without resorting to a binning based approach. Instead, we propagate the leakage/load slope, and use this value of the slope to compare the quality of two Pareto points in terms of leakage. Thus, during the dynamic programming flow of the gain-based TM, the mappings matched to any fanin-cone will have leakage directly proportional to the output capacitance. We have developed a scheme that allows us to compute the leakage slope of a fanin cone as a linear function of the root gate leakage slope and the leakage slopes of the fanin mapping of the root gate.

3. Gain-Based TM for Leakage

In this Section, we develop the necessary models and algorithms for gain-based TM. Two algorithms are presented. The first algorithm minimizes leakage by taking the leakage cost to be the leakage of the dominant state. The second algorithm extends this approach to minimizing leakage at a given percentile value.

3.1 Cell Modeling

A continuous library is assumed and a logic effort model of delay [1] is adopted. In the gain based delay model, the delay of each gate is characterized as $d = g_i h_i + p_i$, where $g_i = C_{out} / C_{in}$ is the electrical effort, h_i is the logical effort and p_i is the parasitic delay. Logical effort theory [1] states that the delay of a path is optimized if the effort delay $g_i h_i$ is balanced along the path. One way to achieve this is to define *a global gain value G* for the inverter or some other fixed library cell which we refer to as *pivotal gate* and then determine the gain value of all the cells according to:

$$g = \frac{G\, h_{inv}}{e}$$

Leakage estimation during technology mapping is difficult because cell widths, and thus leakage, are not known until the full circuit is

mapped. A novel contribution of this paper is the extension of the logical effort based model to leakage. The library consists of N cells. Let the delay, capacitance, dominant leakage, width and gain of gate i be denoted as t_i, $C_{in,i}$, $leak_i$,W_i, g_i respectively. The cells in the library were characterized using SPICE to obtain the following information for cell i:

$$t_i = h_i g_i + p_i = h_i \frac{C_{out}}{C_{in}} + p_i \ , \ C_{in,i} = r_i W_i \ , \ leak_i = q_i W_i \ .$$

After the global gain G is decided, the delay of each cell is fixed and the delay of cell i is given by $T_i^G = G\,h_{inv} + p_i$. If the output capacitance is C_{out} , the leakage of cell i can be estimated in terms of the output capacitance as:

$$T_i^G = \frac{C_{out}}{C_{in}} h_i + p_i, \ \frac{T_i^G - p_i}{h_i} = \frac{C_{out}}{C_{in}}$$

$$C_{in} = C_{out} \left(\frac{h_i}{T_i^G - p_i} \right) W_i = C_{out} \left(\frac{h_i}{r_i (T_i^G - p_i)} \right)$$

$$leak_i = C_{out} \left(\frac{q_i h_i}{r_i (T_i^G - p_i)} \right) = L_i C_{out}$$

Note that now leakage has been expressed as a linear function of output capacitance, a mapping algorithm can be developed which takes the aforementioned library characteristics as input parameters.

3.2 Using Dominant State Leakage as Cost

The initial phase of the gain based algorithm consists of steps which are identical to the binning based approach [10]. Technology independent optimization is followed by conversion of the Boolean netlist to a canonical DAG subject graph *G(V, E)* consisting of only two primitive gates e.g. NAND and INVERTER. Afterwards, all possible matches rooted at each of the node of the subject graph are identified. Since we use a true DAG covering algorithm, matches may include multi-fanout vertices internally.

Dynamic programming is applied in the next phase. It recursively propagates the leakage-delay cost trade-off Pareto pairs for the partial mappings generated at fanin cones of the intermediate nodes of *G*. For a pattern *m* matched at the node *v[i]*, we use the merge operation on all possible combinations of Pareto points at the input pins of *m*. Afterwards, the add operation is applied to combine the costs of resultant mappings to *m*. Pruning is subsequently employed to eliminate sub-optimal Pareto points. If the number of Pareto points in this step still exceeds a predefined threshold, we apply alpha-approximate pruning [9], which results in an exponential decrease in the number of points as a function of the percentage of sub-optimality introduced. In the final stage, we select a Pareto point which has the lowest leakage cost under the target delay from each of the primary outputs. Backward propagation is then performed by visiting the subject graph in a pre-order traversal starting from the library pattern associated with the selected Pareto points at the primary outputs and propagating the slack available to the fanins of the associated pattern.

We show that the leakage of the partial mapping at the fanin cone of any node is a linear function of the output capacitance. Each Pareto point consists of a (d,l) tuple. The element d denotes the delay of the partial mapping and l is the slope of the leakage with respect to the output capacitance. Thus if the output capacitance is $C_{out,}$ then the leakage value of the partial mapping corresponding to the Pareto point is lC_{out} . Note that C_{out} is a quantity that is unknown in the gain based framework until we have fully mapped the circuit. However for pruning purposes it suffices to have only the slope information. We illustrate the pseudo-code for Pareto point merging and adding for dominant leakage based GB-TM in Figure 3. Let M be a pattern match rooted at a subject graph vertex $v[k]$. Let M correspond to the library cell c and let *T1* and *T2* denote the partial mapping at the fanins of M corresponding to the Pareto point (d_1,l_1) and (d_2,l_2). A new partial mapping with the delay d and leakage slope l is then obtained by taking *T1* and *T2* and adding it to the match M. The computation of these values is described below:

$$d = \max(d_1,d_2) + T_c^G$$

The leakage of this combined partial mapping is estimated by observing that the input capacitance of M is C_{out} / g_c . Thus the total leakage is:

$$leak = l_1 \frac{C_{out}}{g_c} + l_2 \frac{C_{out}}{g_c} + C_{out} L_c = \left(\frac{l_1 + l_2}{g_c} + L_c \right) C_{out}$$

Thus, the slope l can be written as:

$$l = \frac{l_1 + l_2}{g_c} + L_c$$

The merging and adding operations can be generalized to the case when the match M has n inputs and corresponding fanin graphs $t_1,\ldots,t_n$ have associated Pareto tuples $(d_1,l_1),\ldots,(d_n,l_n)$ respectively. The delay, leakage-slope costs (d,l) for the partial mapping obtained by combining $t_1,\ldots,t_n$ and the match M can be obtained as:

$$d = \max(d_1,\ldots,d_n) + T_c^G$$
$$l = \frac{l_1 + \ldots + l_n}{g_c} + L_c \qquad (1)$$

Merge_Add (M,c)

1. **Define** $S = \bigcup S_i$ where S_i is the Pareto set at the i-th input pin of M
2. Sort the delay components of all the Pareto points in S. Let the delays be $d_1,\ldots,d_j$
3. For each $d_k \ (1 \le k \le j)$
   a. Find in each S_i the Pareto point P_i with the lowest leakage slope value under the restriction that the delay component is smaller than d_k . If no such d_k exists then set P_i to be NULL.
   b. Combine the picked up Pareto points P_i 's with the match M using (1) to get the Pareto point Q_i
   c. $Q_k.pin[w] = v_w$ for all $(1 \le w \le n)$; $Q_k.leakage = l$; $Q_k.delay = d$; $Q_k.pattern = c$
4. **Return** $(Q_1,\ldots,Q_j)$ which is the set of all the final Pareto points generated.

Figure 3. Algorithm to reduce complexity of the pruning operation from $O(p_1 \ldots p_n)$ to $O(p_1 + \ldots + p_n)$.

In general, we will have multiple Pareto points at the fanins of a match. The operations of merging and pruning can now be

generalized to sets of Pareto points. Let match *M* have *n* inputs which are fanouts of subject graph nodes $v_1,...,v_n$. Each of these v_is have an associated Pareto set $S_1,...,S_n$ which has been assumed to be already computed since we proceed in a topologically sorted order in the DP framework. A naïve way to merge all the Pareto points would be to choose all possible combinations of *n* Pareto points from each of $S_1,...,S_n$ and merge them using (1). However, this has the complexity of $O(p_1...p_n)$ where $p_i = |S_i|$. We can reduce the complexity to $O(p_1 + ... + p_n)$ by employing the approach presented in Figure 3.

For a fixed *M* the Pareto points returned by the Merge_Add function will be co-optimal. However the presence of multiple matches at any subject node *v[i]* will give rise to a combined set of Pareto points, some of which may be redundant. Hence we need a pruning strategy to remove them. Let the Pareto points generated be $(d_1,l_1),...(d_n,l_n)$. A simple strategy is to do pair wise comparison between each pairs (d_i,l_i) and (d_j,l_j) and prune the first point if it has both higher delay and leakage slope. This has the complexity of $O(n^2)$ which can be reduced to $O(n \log(n))$ by using the following approach (Figure 4):

Prune ($(d_1,l_1),...(d_n,l_n)$)

1. Sort the original Pareto set with respect to the delay components in a decreasing order of delay to obtain $(d_1',l_1'),...,(d_n',l_n')$.
2. **Initialize**:
   a. ***After_prune_set*** = $\{(d_n',l_n')\}$
   b. **current** = *(n-1)*
3. Let (d_k',l_k') be the Pareto point with the lowest index *k* in ***After_prune_set***
4. if ($l_{current}' \geq l_k'$)
   prune $(d_{current}',l_{current}')$
   else
   insert $(d_{current}',l_{current}')$ in ***After_prune_set***
5. **current** = **current** – 1
6. if (**current** not equal to 0) goto 3.
7. Return ***After_prune_set***

Figure 4. Algorithm to reduce complexity of pruning Pareto points generated after Merge_Add operation from $O(n^2)$ to $O(n \log(n))$.

The Pareto points are propagated during the post-order traversal shown in Figure 5. We introduce a dummy sink gate with delay *0* and leakage slope *0* which has its inputs as all the primary outputs of the circuit.

Based on the delay constraint, the mapped circuit is recovered during a pre-order traversal of the subject graph. During the pre-order traversal, the arrival time constraints are propagated backwards starting from primary outputs. The library patterns are bound to specific subject graph nodes so that we cover the full subject graph. We always choose Pareto points which satisfy the arrival time constraints with lowest value of leakage slope. For example suppose we encounter a Pareto point *M* which corresponds to pattern *c* which has delay *d*. Further suppose that *M* has an arrival time constraint *a*. The arrival time constraints at *M*'s fanins become *a-d*. We call *a-d* as the arrival time imposed by *M*.

At each fanin of *c*, we pick up a Pareto point with lowest leakage slope which satisfies the arrival time *constraint imposed by M*. In case of multiple primary outputs the quality of the mapped circuit depends on the order in which we process the primary outputs. It is usually most beneficial to perform pre-order traversal in the order of the most critical (in terms of available slack) to least critical primary output [7]. In case of multiple primary outputs we may run into a scenario where during the post order traversal some fanin of the pattern associated with the current Pareto point *M* may be already mapped. Suppose one such fanin cone is rooted at subject graph node *v*. In this case, we check if the currently mapped Pareto point *P* at *v* meets the arrival time constraint imposed by *M*. If this occurs, we don't change the mapped pattern assigned at that fanin. Otherwise, we replace *P* by the best leakage slope Pareto point *Q* rooted at *v* which meets the new arrival time constraint. This is accomplished in two steps: removal of the previous pattern *p* bound at *v* and recursive update of fanin cones of *p*. Potentially the mappings at the fanin cones of *p* can be improved by the removal of the arrival time imposed by *P*. Recursive propagation of the arrival time constraint is then performed at the fanin cones of the children of the patterns associated with *Q*.

Forward_Pass (Subject Graph $G(V,E)$)

1. Visit the subject graph nodes in topological order
2. For current node *v[i]*.
3. For all match m rooted at *v[i]* and associated library gate pattern *c*
   a. **Set** S_m = **Merge_Add**(m,c)
   b. **Set** $S = \cup S_m$ " $match\ m$
   c. **Set** S' = **Prune**(S)
   d. $v[i].P = S'$

Figure 5. Algorithm for post-order traversal of subject graph.

3.3 TM for Quantile Leakage Optimization

The uncertainty in the input vector state is captured by specifying the probabilities of each of its primary input being 1. Thus leakage becomes a random variable due to the state dependency. The proposed technique for gain-based technology mapping minimizes the value of leakage power at a certain quantile a .

$$\min q_a(leakage)$$
$$d_{circuit} \leq d_{given}$$

In order to map a circuit such that leakage at any quantile value is minimized (due to input state uncertainty), we would like to characterize the leakage of library gates for each input vector instead of the worst case leakage. Let the library gate has *i* have x_i inputs. Thus, it has $y_i = 2^{x_i}$ input states. In an input state *e*, let the leakage be characterized as $leak_i^e = q_i^e W_i$ where, W_i is the width of gate *i*.

We evaluate input state probability at each of the intermediate signals of the subject graph by random simulation. The random simulation is done by generating the primary inputs at each iteration of the Monte Carlo simulator based on the primary input signal probabilities. We associate a function with the leakage random variable *L* that penalizes for high leakage variance defined as: $Penalty(L) = E[L] + l\sqrt{Var(L)}$.

Table 1: Comparison between the penalty function based technology mapper and the dominant state leakage based technology mapper. Leakage is measured at the 99th quantile.

Benchmark	Size	Max leakage improvement (%)	Average leakage improvement (%)	Run Time for Gain-based Mapping (s)	Run time for Bin-based Mapping (s)
9symml	311	31	23.28	10	75
C1908	1167	17.80	12.14	60	802
C7552	4609	22.48	9.52	200	5006
C5315	3448	11.44	6.05	110	1210
C3540	1942	9.32	6.65	100	1007
C1355	706	70.12	40.60	40	362
Apex6	886	10.04	7.89	30	210
Alu2	565	12.26	6.41	20	194
C880	521	9.59	7.27	20	190
C499	711	20.98	8.92	40	434
Average	1486	21.49	12.8	63	949

The *penalty parameter* l allows for tuning of the penalty accorded to the variance. In general we run through various sweeps of the algorithm for different value of l . Similar to the dominant case leakage, it can be shown that this penalty function is also a linear function of the output capacitance. We illustrate its computation assuming that the gate i is facing output capacitance C_{out} . Let the input states be $e_1,...,e_n$ with probabilities $p_1,...,p_n$. According to the equations in Section 3.1 derived for the leakage in terms of the output capacitance, we deduce that leakage in the state e_j is given by:

$$leak_i^{e_j} = C_{out}\left(\frac{q_i^{e_j} h_i}{r_i(T_i^G - p_i)}\right) = C_{out} L_i^{e_j}$$

Hence, the penalty function can be computed as:

$$C_{out}\left(\sum_j p_j L_i^{e_j} + l\sqrt{\sum_j p_j (L_i^{e_j})^2 - \left(\sum_j p_j L_i^{e_j}\right)^2}\right)$$
$$= C_{out} L_{penalty,i}^{p_1,...,p_n,l}$$

Thus, the penalty function is directly proportional to the output capacitance and is only dependent on the state probabilities $p_1,...,p_n$. We use as our cost metric for leakage, the slope of this penalty with respect to output capacitance. A Pareto point consists of delay and the leakage penalty slope $(d, l_{penalty})$. Note that unlike the case of worst case leakage, $l_{penalty}$ for a cell is also dependent on the input state probability of the matched pattern. Thus, this value cannot be pre-computed for library gates and must be computed during the post order traversal during the forward phase of algorithm based on the actual state probability information of the library pattern. The simulation for a subject graph can be done before the run of mapping algorithm since the signal state probabilities are independent of a specific technology binding of the subject graph.

The equations for adding and merging can be written as follows similar to (1):

$$d = \max(d_1,...,d_n) + T_c^G$$
$$l_{penalty} = \frac{l_{penalty,1} + ... + l_{penalty,n}}{g_i} + L_{penalty,c}^{p_1,...,p_n,l}$$

The algorithm flow is similar to the case of dominant state GB-TM described in the previous Subsection except that we replace the leakage slope of the dominant leakage state by the leakage slope of the penalty function.

4. Experiments and Results

The library consists of 9 cells with different functionalities and 60 structurally different cells. For each cell we consider all possible input states and identify the dominant leakage states. For each of these dominant leakage states, we select the leaking transistors and set them all to a high Vth. Thus, for each library gate we have several mixed dual-Vth versions. Having such a configuration space of mixed dual-Vth gates allows us to reduce leakage significantly without having to pay a high delay penalty [8]. We always have two versions for each gate in which all transistors are either set to low Vth or high Vth. In all our experiments, we have set the probability of any primary output being 1 to be 0.5.

The previous work in [2] uses gain-based TM for the minimization of area under delay constraint. The technique presented is also applicable to leakage minimization under delay constraint. The area-delay trade-off in [2] is generated by obtaining minimum delay mappings by fixing various global gain values. As we sweep across the gain in increasing order we obtain successively circuit configurations with higher delay and lower area. The mappings corresponding to the least delay for each value of global gain result in a monotonic curve of area as a function of delay. An analogous technique also provides a trade-off between leakage and delay. In our experiments, the global gain *G* was varied from 1.111 to 1.120 in steps of 0.001 and the pivotal gate was chosen to be low threshold voltage NAND4 gate. These choices are arbitrary and others could have been set-up as well. In Figure 6, we show the percentage improvement in leakage with respect to the Pareto points generated by the mapper of [2]. The average leakage gain is 14% and the leakage gain can increase to as much as 20% for

some benchmarks. Improvement is achieved at the cost of a minor (1%) penalty in delay with respect to mappings produced by [2].

In the next experiment, we contrast the dominant leakage versus the quantile GB-TM algorithms wherein both algorithms are implemented in our gain-based framework and perform simultaneous propagation of delay-leakage tradeoff Pareto points is performed in both cases. The results we obtain indicate that the

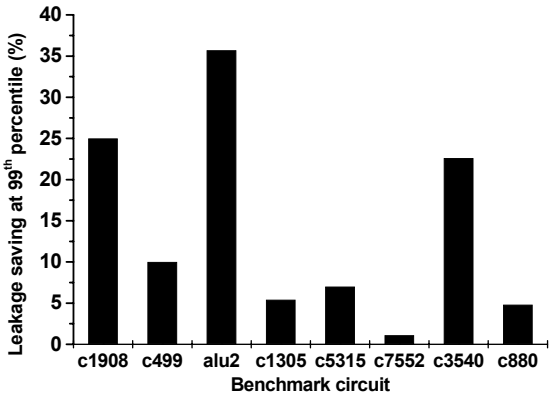

Figure 6. Comparison of leakage savings at 99th percentile enabled by our approach with the algorithm presented in [2]. We are able to achieve 14% savings in leakage on average.

dominant-state leakage minimization gives sub-optimal results compared to the penalty function based approach. This can be attributed to the ability of the latter approach to account for the impact of signal state probabilities on total circuit leakage. The comparison results are reported in Table 1. For each benchmark, we used 10 different global gain values of 1.75, 1.85, 2, 2.1, 2.25, 2.35, 2.5, 2.75, 3.25 and 3.5. Across the benchmarks, we obtain average savings of 12.8% and maximum savings of 21.49% at 99% quantile leakage. Our algorithm has extremely small run-time and provides a 15X speed-up over a bin based algorithm utilizing a discretized library. The run-time reported in the table is cumulative run-time for the 10 gain values and for a single value of penalty factor l . We made seven choices 0.1, 0.2, 0.3, 1, 2, 3 and 0 for the penalty factor.

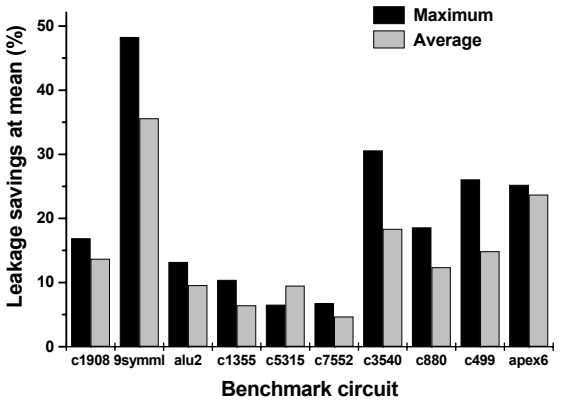

Figure 7. Comparison of mean leakage minimization and dominant leakage minimization. The y-axis gives savings at the mean value of leakage. We obtain on 14.5% average savings 20.5% maximum savings.

Finally, we demonstrate that it is possible to obtain significant reduction in mean leakage using the GB-TM algorithm compared with dominant state leakage minimization (Figure 7). The algorithm minimizes mean leakage if we choose the penalty factor of 0. On comparing the Pareto sets generated by our algorithm for mean against the Pareto point of the dominant leakage minimizing mapper, we obtain on 14.5% average savings 20.5% maximum savings in leakage across the benchmarks.

5. Conclusion

We demonstrated that the leakage of any partial mapping has a linear dependency on the output capacitance, and implemented a gain-based mapper that uses this model. We have also showed the need for probability aware optimization.

6. Acknowledgments

This research was supported in part by SRC, GSRC, NSF, SUN, Intel and the University of Texas.

7. References

[1] I. Sutherland and R. Sproull, "The theory of logical effort: designing for speed on the back of an envelope," *Advanced Research in VLSI*, 1991.

[2] B. Hu *et al*, "Gain-based technology mapping for discrete-size cell libraries," *Proc. of DAC*, 2003.

[3] K. Keutzer, "Technology binding and local optimization by DAG mapping," *Proc. of DAC*, 1987.

[4] J. Grodstein *et al*, "A delay model for logic synthesis of continuously-sized networks," *Proc. of ICCAD*, 1995.

[5] K. Chaudhary and M. Pedram, "A nearly optimal algorithm for Technology Mapping minimizing area under delay constraint," *Proc. of DAC,* 1992.

[6] M. Iyer *et al*, "Wavefront technology mapping," *International Workshop on Logic synthesis*, 1998.

[7] C. Kang and M. Pedram, "Technology mapping for low leakage power and high speed with hot carrier effect consideration," *Proc. of ASPDAC*, 2003.

[8] D. Lee and D. Blaauw, "Static leakage reduction through simultaneous threshold voltage and state assignment,"*Proc. of DAC*, 2003.

[9] S. Roy *et al*, "An alpha-approximate algorithm for delay constrained technology mapping," *Proc. of DAC*, 1999.

[10] R. Rudell, "Logic synthesis for VLSI design," *PhD. Thesis*, University of California, Berkeley, 1989.

[11] S. Borkar *et al*, "Parametric variation and impact on circuits and microarchitecture," *Proc. of DAC,* 2003.

[12] P. Kudva *et al*, "Gain-based logic synthesis," *Proc. of ICCAD tutorial*, 2000.

[13] E. Acar *et al*, "Leakage and leakage sensitivity computation for combinational logic," *Journal of Low Power Electronics*, Volume 1, Number 2, 2005.

[14] S. H. Gunther *et al*, "Managing the impact of increasing microprocessor power consumption," *Intel Technology Journal*, Q1, 2001.

[15] D. Jongeneel *et al*, "Area and search space control for technology mapping," *Proc. of ASP-DAC*, 2000.

[16] Y. Kukimoto *et al*, "Delay-optimal technology mapping by DIG covering," *Proc of DAC*, 1998.