A Fanout Optimization Algorithm Based on the Effort Delay Model

Peyman Rezvani and Massoud Pedram

***Abstract*—This paper presents a Logical Effort-based fanout OPtimizer for ARea and Delay (LEOPARD), which relies on the availability of a (near) continuous size buffer library. Based on the concept of logical effort in very large scale integrated circuits, the proposed algorithm attempts to minimize the total buffer area under the required time and input capacitance constraints by constructing the fanout tree topology and assigning the buffer sizes. More precisely, the proposed algorithm produces the optimum fanout tree solution if the fanout tree topology is restricted to a chain of buffers. For the case where a discrete size library of buffers is available, this paper also presents a postprocessing (buffer merging) step that transforms the continuous buffer-sizing solution to a discrete one while minimizing the round-off error. Experimental results show that compared with previous approaches, both for continuous and discrete buffer libraries, LEOPARD achieves a significant reduction in the total buffer area subject to the required time constraints.**

***Index Terms*—Buffer insertion, fanout optimization, gate sizing, logic design, logical effort.**

I. Introduction

Quite often in a very large scale integrated (VLSI) design, a signal needs to be distributed to several destinations under a required timing constraint at each destination. Furthermore, in practice, there may also be a limitation on the load that can be driven by the source signal. Fanout optimization is the problem of finding a buffer-tree topology and sizing the buffers in this topology so as to satisfy the constraints. Since these buffers must be picked from the sizes that are available in a given cell library, the more realistic problem is to find the optimum sizes for the buffers from the set of sizes available in the library. This problem has been proved to be NP-complete [1]. While several approaches exist for tackling the fanout optimization problem using simplified delay models [9], [10], new techniques [12] have also been proposed which use more accurate delay models or even taking interconnect delay into account [11]. More recently, however, researchers [3] have started to use continuous, as opposed to discrete, size libraries, in the sense that the optimum fanout tree is calculated with the assumption that buffers are available in all sizes. This greatly simplifies the problem and allows the application of more powerful optimization techniques. At the same time, the number of discrete sizes for inverters in a typical application-specified integrated circuit (ASIC) library has increased to the extent that a "near-continuous inverter sizing" model has become a valid and fairly accurate model.

In [2], the authors simplified the fanout optimization problem by restricting the search space to a subset of trees and showed that the results still compare very favorably with the algorithms that consider a larger set of topologies. The authors used a dynamic programming approach to implicitly enumerate the set of so-called LT-trees and find the optimal LT-tree topology and sizing. An LT-tree is either a 2-level buffer type or a chain of buffers with intermediate fanouts to sinks that ends up to sinksor to a 2-level tree. Reference [3] also restricted the search space to a certain class of trees, called fanout-free trees, and showed that there still exists an optimal solution in this search space under a gain-based delay model. Fanout-free trees are trees in which a buffer can drive at most one other buffer.

In this paper, an algorithm is presented that finds the fanout tree topology and sizes of the buffers on the tree by decomposing the whole problem into subproblems and solving each subproblem separately for each sink. The solutions to the subproblems are then merged to form the solution to the whole problem. Our derivation relies on the notions of logical and electrical effort first proposed in [4].

Sutherland and Sproull [4] minimized the delay along any single path by assigning equal delay budgets to each stage on that path. While this approach was proven to minimize the delay, it did not necessarily result in an optimal solution in terms of the total buffer area. Kung [3], on the other hand, solved the fanout-optimization problem to minimize the input capacitance seen at the source gate subject to timing constraints for the sinks and without any consideration of the buffer area. In contrast, the approach presented in this paper minimizes the total buffer area subject to capacitance constraint for the driver. This is an important distinction because it allows one to tradeoff the propagation delay through the source driver and through the rest of the buffer tree to reduce the total buffer area without too high of an increase in the overall delay.

The remainder of this paper is organized as follows. In Section II, the effort delay model that is used throughout this paper is explained. Section III explains the details of the algorithm. In Section IV, experimental results are shown, and in Section V, we conclude the paper.

II. Delay Model

The delay model used in this paper is based on the concept of logical and electrical efforts presented in [4]. The effort-based model is basically a reformulation of the conventional *RC* model of CMOS gate delay.

Using the same terminology as in [4], the delay of a gate is defined to be

$$d = \tau(p + gh) \tag{1}$$

where τ is a time unit that characterizes the semiconductor process being used. It is only used to convert the unitless part of $(p + gh)$ to a time unit. For simplicity, τ is not considered from now on. Parameter p is the parasitic delay of the gate. The major contribution to the parasitic delay is the capacitance of the source/drain regions of the transistors that drive the output. Throughout this paper p_{inv} is used as the parasitic delay for an inverter. Parameter g is called the *logical effort* of the gate and depends only on the topology of the gate and the ability to produce output current. The logical effort for an inverter is assumed to be 1 and, for other gates, calculated based on their internal topologies. The logical effort of a logic gate tells how much worse it is at producing output current than is an inverter, given that each of its inputs may have only the same input capacitance as the inverter. Parameter h (specified for each input pin of the gate) is called the *electrical effort* (also called gain) of the gate and is defined to be the ratio of the capacitive load driven by the gate to the input capacitance at the corresponding input pin. The electrical effort describes how the electrical environment of the logic gate affects performance and how the size of the transistors in the gate determines its load-driving capability.

The important point is that p and g are independent of the size of the gate, and the only factor that is affected by sizing is the electrical effort h. Reference [4] shows how p and g are independent of sizing by doing the reformulation to define the four factors τ, p, g, and h in terms of the resistance and capacitance of a minimum size inverter and a template gate representing the topology of the gate. For details, refer to [4].

Manuscript received October 7, 2003; revised March 16, 2003. A preliminary version of this work appeared in *Proc. Int. Conf. Computer-Aided Design*, San Jose, CA, pp. 516–519, Nov. 1999. This paper was recommended by Associate Editor M. D. F. Wong.

The authors are with the Department of Electrical Engineering-Systems, University of Southern California, Los Angeles, CA 90089 (e-mail: peyman@usc.edu; pedram@usc.edu).

Digital Object Identifier 10.1109/TCAD.2003.819423

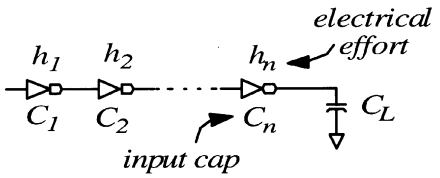

Fig. 1. Buffer chain.

III. Algorithm

In this section, the fanout optimization problem is stated as two separate problems, and each one is solved separately.

One-sink fanout optimization (1FO) problem: Given the source of a signal Q with maximum driving capability C_{in} and a sink S with capacitive load C_L, required polarity P, and required arrival time T_R, find the optimum number of buffers for a buffer chain and the appropriate sizing for them to minimize the total buffer area such that the delay from Q to S is less than or equal to T_R, the required polarity P is achieved, and the capacitive load imposed on Q is no more than C_{in}.

Multiple-sink fanout optimization (mFO) problem: Given the source of a signal Q with maximum driving capability C_{in} along with a set of m sinks S_i each of which is assigned a triplet (C_{L_i}, T_{R_i}, P_i) where C_{L_i} is the capacitive load, T_{R_i} is the required arrival time, and P_i is the required polarity for the sink S_i, find a fanout tree of buffers and the appropriate sizing for them to minimize the total buffer area such that the timing constraint and the polarity required at each sink is satisfied and the capacitive load imposed on Q is no more than C_{in}.

Note that the only difference between the two problems is the number of sinks to be driven. Area, the objective function in both of these problems, is considered to be the summation of input capacitances of all the buffers, which is reasonable with the assumption of continuous sizing for the gates.

The rest of this section is organized as follows. The 1FO problem is solved in Section III-A, and in Section III-B, the mFO problem is solved based on the solution derived for the 1FO problem.

A. Buffer Chain

For the 1FO problem, the solution is a chain of buffers between the source and the sink (Fig. 1). The variables of the problem are defined to be the number of buffers n and the electrical efforts of these buffers $h_1, h_2, \ldots, h_n$.

Since the logical effort for an inverter is 1, the delay through the buffer chain can be expressed in terms of n and h_is as follows:

$$\text{delay} = np_{\text{inv}} + \sum_{i=1}^{n} h_i. \tag{2}$$

The overall area, which is calculated as the summation of the input capacitances of all buffers on the buffer chain, may subsequently be expressed as

$$\text{area} = \sum_{i=1}^{n} C_i = \sum_{i=1}^{n} \frac{C_L}{\prod_{j=i}^{n} h_j}. \tag{3}$$

The goal would be to find n and all h_is to minimize area while both timing and input capacitance constraints are satisfied. That is

$$\begin{cases} \text{Min} & \text{area} \\ \text{st}: & \text{delay} \le T_R \\ & C_1 \le C_{\text{in}}. \end{cases}$$

Theorem 1: In the 1FO problem, delay through the optimum buffer chain is exactly equal to the specified required time T_R, i.e., delay $= T_R$.

Proof: According to (3), area is a monotonically decreasing function of all h_is $(i = 1, \ldots, n)$. In other words, increasing any h_i will always result in a buffer chain with smaller area. The delay, on the other hand, is an increasing function of all h_is according to (2). This means that by increasing any arbitrary h_i, area can be decreased and delay can be increased up to the point that delay becomes no larger than the given constraint T_R; therefore, the optimum buffer chain has delay $= T_R$. ■

Lemma 1: In the 1FO problem, for a fixed number of buffers n in the chain, the optimum buffer chain has $\sum h_i$ equal to a constant $T_R - np_{\text{inv}}$.

Proof: According to Theorem 1 and (2)

$$np_{\text{inv}} + \sum_{i=1}^{n} h_i = T_R.$$

The first term on the left hand side np_{inv} is constant for a given n. Therefore, $\sum h_i$ for the optimum buffer chain with n buffers is also constant and equal to

$$\sum_{i=1}^{n} h_i = T_R - np_{\text{inv}}. \tag{4}$$

Hence, the claim is proved. ■

To find the optimum number of buffers n the maximum input capacitance constraint $C_1 \le C_{\text{in}}$ is used, where C_1 is the input capacitance of the first buffer in the chain being driven by the source signal and C_{in} is the given constraint on the input capacitance.

The input capacitance for the first buffer is computed as follows:

$$C_1 = \frac{C_L}{\prod h_i}. \tag{5}$$

Let the electrical effort of the chain be defined as the product of electrical efforts of all the buffers, and let it be shown by H. Using the above equation, the input capacitance constraint can be restated as follows:

$$H = \prod h_i = \frac{C_L}{C_1} \ge \frac{C_L}{C_{\text{in}}}. \tag{6}$$

Theorem 2: In the 1FO problem, for a fixed number of buffers n in the chain, the electrical effort of the buffer chain H achieves its maximum value when all h_is are equal.

Proof: According to Lemma 1, the summation of all h_is is constant for any given number of buffers. Since the product of some variables with a constant summation is maximum when all those variables are equal, all h_is have to be equal to maximize H. ■

The electrical effort of each buffer for the buffer chain that maximizes H, according to Theorem 2 and (4), would then be

$$\hat{h}_i = \hat{h} = \frac{T_R - np_{\text{inv}}}{n} \quad \forall i = 1, \ldots, n. \tag{7}$$

So, the maximum of H, named $\overline{H}$ as a function of n would be

$$\overline{H} = \left(\frac{T_R - np_{\text{inv}}}{n}\right)^n. \tag{8}$$

$\overline{H}$ is drawn in Fig. 2 for $T_R = 14$ and $p_{\text{inv}} = 0.6$.

According to Theorem 2, there is a maximum value that H can achieve for any given buffer count. Therefore, the only buffer counts that are feasible are those for which the maximum value that H achieves is not less than the ratio C_L/C_{in} (6) and those correspond to the buffer counts between the points of intersection of $\overline{H}$ and line C_L/C_{in} (Fig. 2). As an example, for Case I in Fig. 2, there is no feasible solution because there are no intersection points and $\overline{H}$ lies below C_L/C_{in} for all buffer counts. For Case III, on the other hand,

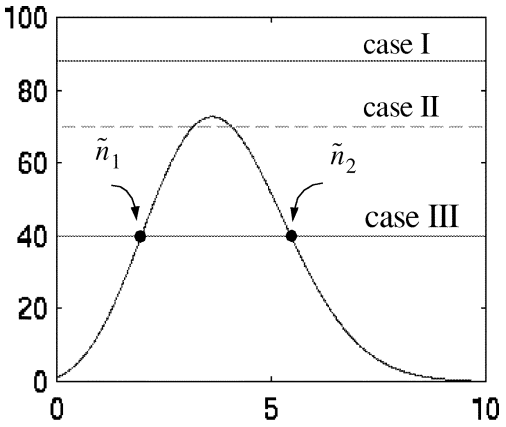

Fig. 2. Plot of $\overline{H} = \mathrm{Max}(\prod h_i)$ versus n.

algorithm OptN (C_{in}, C_L, T_R, p)

1. **begin**
2. $(\tilde{n}_1, \tilde{n}_2) = solve\left(\left(\frac{T_R - np_{inv}}{n}\right)^n = \frac{C_L}{C_{in}}\right)$;
3. $n_1 = \lceil \tilde{n}_1 \rceil$ or $\lceil \tilde{n}_1 \rceil + 1$ depending on polarity p;
4. n_2= **min** $(\lfloor \tilde{n}_2 \rfloor, \lfloor T_R/p_{inv} \rfloor)$;
5. **for** $n = n_1, \ldots, n_2$ **step** 2
6. $$\{h\} = \begin{cases} Min & area_n \\ st: & delay_n \le T_R \\ & C_1 \le C_{in} \end{cases}$$
7. **return** (best n, best {h});
8. **end**

Fig. 3. Algorithm $OptN$.

there are two points of intersection $\tilde{n}_1$ and $\tilde{n}_2$; therefore, the only feasible buffer counts are between $\tilde{n}_1$ and $\tilde{n}_2$.

With these observations, algorithm $OptN$ in Fig. 3 is proposed for finding the optimum number of buffers and their sizes.

To find the optimum number of buffers, the line C_L/C_{in} is intersected with the graph $\overline{H}$ (line 2 of Fig. 3 and Case III in Fig. 2) which results in $\tilde{n}_1$ and $\tilde{n}_2$. Note that

$$\lim_{n \to 0} \overline{H} = 1. \tag{9}$$

Therefore, there always exists an $\tilde{n}_1$ unless the line C_L/C_{in} is passing below unity, which means that C_L is less than or equal to C_{in}, in which case, no buffers need to be used at all. On the other hand, there exists an upper bound on the number of buffers because of the intrinsic buffer delay. According to (4), for the electrical efforts of buffers to have a meaningful physical interpretation, $T_R - np_{\mathrm{inv}}$ has to be positive, which means (line 4 of Fig. 3)

$$n \le \frac{T_R}{p_{\mathrm{inv}}}. \tag{10}$$

In short, the buffer count is limited by $\tilde{n}_1$ on one side and by $\tilde{n}_2$ and T_R/p_{inv} on the other side. Therefore, the optimum buffer count, n, lies between n_1 and n_2 (lines 3 and 4 of Fig. 3).

There is a possibility that the line C_L/C_{in} could intersect the graph where there is no integer n between the points of intersection to satisfy the polarity constraint. This only happens when the line crosses the $\overline{H}$ curve very close to the peak of the graph (Case II in Fig. 2). In lines 5 and 6, the optimum sizing for the buffers on the chain is found by solving a convex optimization problem as follows:

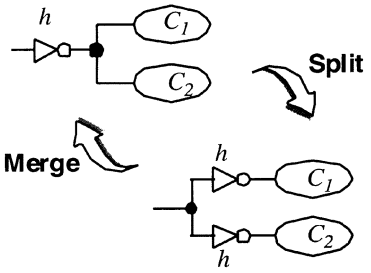

Fig. 4. Split/merge transformations.

$$\begin{cases} \mathrm{Min} & \frac{C_L}{h_n} + \frac{C_L}{h_n h_{n-1}} + \ldots + \frac{C_L}{h_n h_{n-1} \ldots h_1} \\ \mathrm{st}: & h_1 + \ldots + h_n \le T_R - np_{\mathrm{inv}} \\ & h_1 \ldots h_n \ge \frac{C_L}{C_{\mathrm{in}}}. \end{cases} \tag{11}$$

This is a minimization of a posynomial function with posynomial inequality constraints that can be easily solved in polynomial time [6]. Finally, among all of the solutions, the one with the minimum area is selected as the optimum solution.

It is interesting to note that by taking the derivative of $\overline{H}$ and setting it equal to zero, its maximum value is found to be at

$$\hat{n} = T_R \times \lambda(p_{\mathrm{inv}}) \tag{12}$$

where

$$\lambda(p_{\mathrm{inv}}) = \frac{\mathrm{Lambert}\left(\frac{p_{\mathrm{inv}}}{e}\right)}{p_{\mathrm{inv}}\left(\mathrm{Lambert}\left(\frac{p_{\mathrm{inv}}}{e}\right) + 1\right)}. \tag{13}$$

The function Lambert(ω) is the solution to the nonlinear equation $xe^x = \omega$. For further information about Lambert function, refer to [5]. As p_{inv} tends toward zero

$$\lim_{p_{\mathrm{inv}} \to 0} \lambda(p_{\mathrm{inv}}) = \frac{1}{e} \tag{14}$$

and this corresponds to allocating the well-known electrical effort of e to each buffer with the assumption of $p_{\mathrm{inv}} = 0$.

Theorem 3: Algorithm $OptN$ finds the optimum solution for the 1FO problem.

Proof: Since all of the feasible solutions are explicitly considered, the algorithm is guaranteed to find the optimum solution. ■

B. Buffer Tree

In this section, the more general case of the fanout-optimization problem is considered, where the source signal is driving more than one sink.

Reference [3] introduced two transformations that can be performed on a fanout tree, namely *merging* and *splitting* (Fig. 4). It is shown here that these transformations maintain the same area, delay, and capacitance.

Theorem 4: The split/merge transformations applied to a fanout tree preserve the input capacitance (thus, area) and the delay.

Proof: The proof for split transformation is as follows. Suppose the electrical effort of the original buffer before splitting is h. Thus, the delay through the buffer for both of the branches is $h + p_{\mathrm{inv}}$, and the input capacitance is $(C_1 + C_2)/h$, which is also the area of the buffer. After splitting the original buffer to two buffers with equal electrical efforts of h, the delay for both branches would still be $h + p_{\mathrm{inv}}$ and the input capacitance would be $C_1/h + C_2 + h$, thus, the same input capacitance and, hence, the same area. For merge transformation, one can easily verify the same provided that the electrical efforts of the buffers to be merged are equal. ■

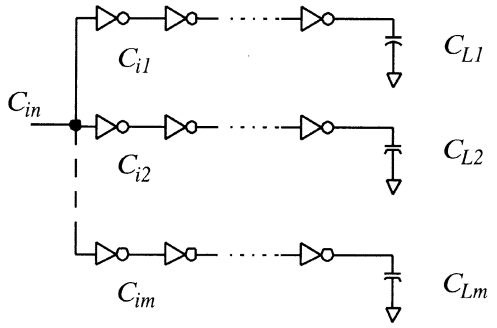

Fig. 5. Input capacitance allocation for a fanout-free buffer tree.

Therefore, if T^* is the optimal fanout tree with the proper sizing of buffers, it can be split to a *fanout-free tree* consisting of a set of buffer chains $\overline{T}$, which has the same area as T^*, according to Theorem 4, and also satisfies the timing and input capacitance constraints (Fig. 5). First, $\overline{T}$ will be found by using the optimal algorithm presented in Section III-A. The method used to transform $\overline{T}$ into T^* will be discussed later.

The 1FO problem was stated such that the maximum input capacitance allowed was given. Therefore, before the mFO problem can be broken down into 1FO problems, different portions of C_{in} need to be allocated to each branch (Fig. 5).

Input capacitance allocation (ICA) problem: Given a number of sinks, each with a required time, capacitive load, and required polarity, and a total budget on input capacitance $\mathrm{C_{in}}$, allocate portions of $\mathrm{C_{in}}$ to each branch, such that the total area is minimized while the given constraints for all sinks are satisfied.

In this section, it is first proven that the ICA problem is NP-complete and then a heuristic is proposed for solving this problem.

Intuitively speaking, the input capacitance allocation problem is similar to Knapsack problem, where objects of the Knapsack problem correspond to the capacitance budgets of each branch and the total capacitance is limited by the input capacitance constraint C_{in}, which corresponds to the Knapsack volume.

Before it can be formally proven that this problem is NP-complete, the behavior of area must be studied as a function of input capacitance for each branch. The valid range for the buffer count on branch i is [1, $\lfloor T_{R_i}/p_{\text{inv}} \rfloor$], according to (10). For each buffer count n, in this range, there exists a maximum electrical effort for the buffer chain, according to (8). Therefore, because of the capacitance constraint in (6), there exists a minimum required input capacitance as follows:

$$\underline{C_i} = \frac{C_{L_i}}{\left(\frac{T_{R_i} - n p_{\text{inv}}}{n}\right)^n} \tag{15}$$

where the denominator is the maximum value that can be achieved by $\prod h$, according to (8). On the other hand, there exists a maximum beneficial input capacitance, $\overline{C}_i$, for each buffer count which means that allocating an input capacitance larger than $\overline{C}_i$ will not improve area any further. This value can be calculated using the same optimization problem as in (11) but with dropping the capacitance constraint. That is

$$\{h\} = \begin{cases} \text{Min} & \text{area}_n \\ \text{st}: & \text{delay}_n \le T_R \end{cases}$$

and then calculating $\overline{C}_i$ as follows:

$$\overline{C}_i = \frac{C_{L_i}}{\prod h}.$$

Obviously, any input capacitance larger than $\overline{C}_i$ will not improve area any further because allocating $\overline{C}_i$ already results in the same solution as when the capacitance constraint is dropped.

Now that there exists a range for input capacitance for each buffer count, it can be proven that area is a decreasing function of input capacitance in this range.

Theorem 5: For a fixed number of buffers in a buffer chain, the area cost is a decreasing function of input capacitance for $\underline{C}_i \le C_{\text{in}} \le \overline{C}_i$.

Proof: Increasing input capacitance C_{in} for a branch will decrease the ratio C_L/C_{in} in the capacitive constraint of the optimization problem in (11). Therefore, there either exists a better solution with smaller area or, if not, the same solution with the same area is still achievable. Hence, increasing input capacitance will not increase area and, therefore, area is a decreasing function of input capacitance and claim is proven. ■

Area versus input capacitance for some buffer count will, therefore, look something like the graph shown in Fig. 6(a). As shown in Fig. 6(a), no feasible solution exists for input capacitances smaller than $\underline{C}_i$ and the area stays the same for input capacitances larger than $\overline{C}_i$. Different buffer counts in the range [1, $\lfloor T_{R_i}/p_{\text{inv}} \rfloor$] result in the graphs shown in Fig. 6(b). The minimum area over all buffer counts will, therefore, look like the graph shown in Fig. 6(c). This piecewise nature of area versus input capacitance, which is due to different buffer counts, causes the ICA problem to be NP-complete.

Theorem 6: ICA problem is NP-complete.

Proof: To perform the proof, the 0-1 Knapsack problem will be reduced to the ICA problem. In the conventional version of the Knapsack problem, each item has a size and a value and the objective is to maximize the total value. In the ICA problem, however, the objective is to minimize area. Therefore, we will consider the negative of area, rather than the area itself, so as to make the problem a maximization problem rather than a minimization one [Fig. 7(a)].

The value versus size curve for some item of 0-1 Knapsack problem is shown in Fig. 7(b). The point about this graph is that it is not a continuous one. For sizes below s_i, the value is zero, and for sizes greater than s_i, the value is v_i. Assuming δ to be the accuracy of the machine, the graph can be modified to the one shown in Fig. 7(c) to make it a continuous one. Note that the graph may have any arbitrary behavior in the range between s_i and $s_i + \delta$. This new graph is a special case of the graph shown in Fig. 7(a), in which the curve has become linear. Since the 0-1 knapsack problem is NP-complete, the ICA problem is NP-hard as well, otherwise one could formulate the 0-1 Knapsack problem as an ICA problem and solve it in polynomial time. Note that the NP-hardness of ICA is because of the piecewise nature of the area versus input capacitance curve and, that, in turn, is because area is represented by different functions for different buffer counts. Now that it has been proven that ICA is NP-hard, it must be shown that the decision version of ICA can be tested in polynomial time. This is obviously true because one can easily add up the input capacitances of each branch and compare it with the input capacitance budget C_{in}. This can be done in linear time, meaning ICA is in NP, and since it was proven that ICA is NP-hard, therefore, the ICA problem is NP-complete. ■

After proving that ICA is an NP-complete problem, this section proceeds by proposing a heuristic method for allocating input capacitances to each branch.

Let m denote the number of sinks and, thus, the number of branches. Consider the kth branch $(1 \le k \le m)$ and $\overline{H}_k$, the maximum of electrical effort of the kth branch, has its minimal value of 1 at $n_k = 0$ (lim. $\overline{H}$ when n tends toward 0). On the other hand, $\overline{H}_k$ cannot be any larger than $\mu(T_{R_k}, p_{inv})$, the value of $\overline{H}_k(n_k)$ when n_k is calculated from (12). According to (5), the maximum value of $\overline{H}_k$ corresponds to the minimum value of C_{ik}. Therefore, the minimum acceptable input capacitance would be

$$\underline{C}_k = \frac{C_{Lk}}{\mu\left(T_{R_k}, p_{\text{inv}}\right)}. \tag{16}$$

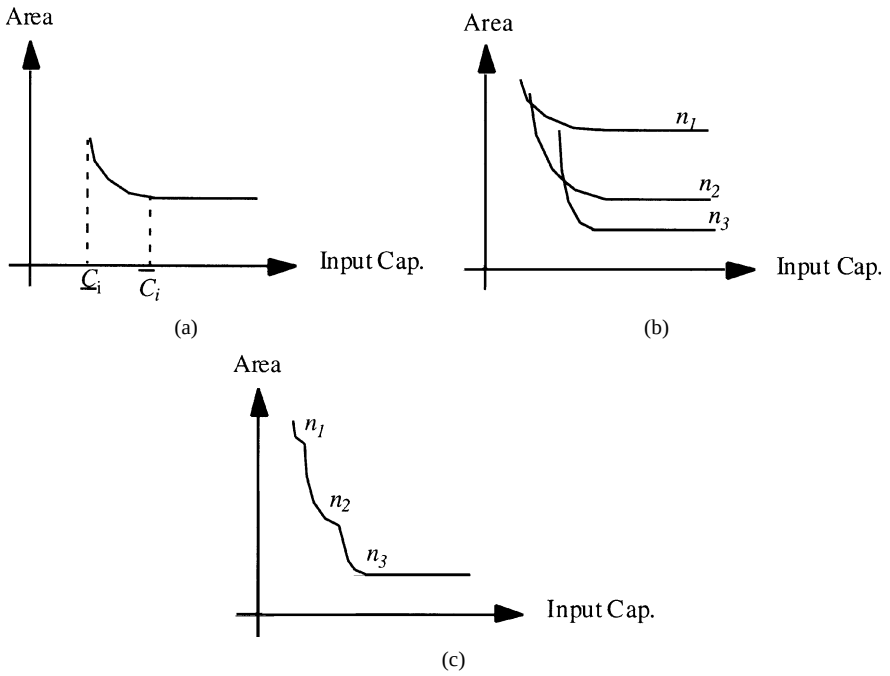

Fig. 6. (a) Area versus input cap for some buffer count n. (b) Area versus input cap for different buffer counts. (c) Minimum area versus input cap.

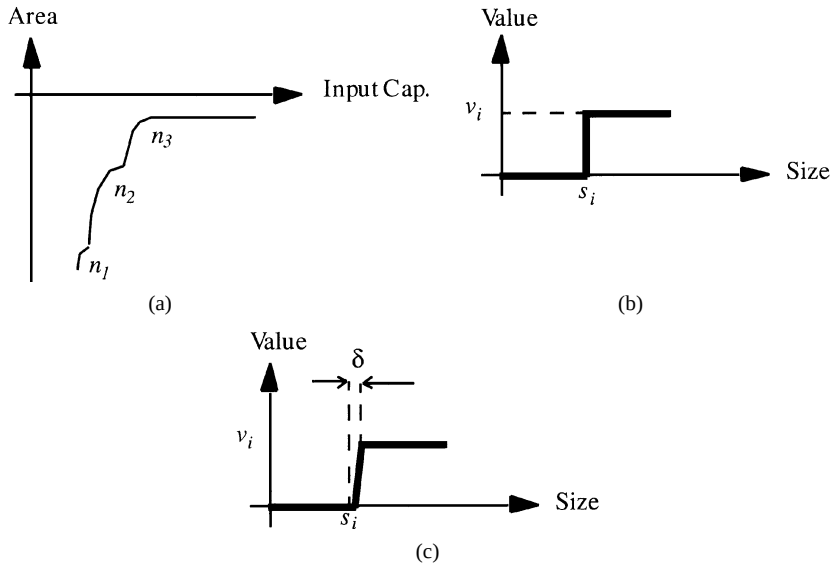

Fig. 7. (a) Area versus input cap. (b) Value versus size for an item of knapsack problem. (c) Modified value versus size graph.

Allocating any capacitance less than $\underline{C}_k$ to any branch will make that branch infeasible. Hence, m new positive variables x_k for $k = 1, \ldots, m$ are introduced such that

$$C_{ik} = \underline{C}_k + x_k. \tag{17}$$

This way, one can be sure that the minimum required capacitance is allocated to each branch. The heuristic is to find x_ks in such a way that their ratio is proportional to the positive slope of $\overline{H}$ graph in Fig. 2. The motivation behind this heuristic is the fact that for two different branches to have the same change in buffer count, the branch with smaller slope would need a smaller change in C_L/C_{in}. When a branch is given a wider range of buffer counts to explore, a better solution will likely be found. For an example, refer to Fig. 8. Branch 1 has a larger slope compared to branch 2; therefore, a larger change in C_L/C_{in} for branch 1 is required to have the same buffer count range as branch 2. Since C_L is given and fixed for each branch, changing C_L/C_{in} corresponds to changing the C_{in} allocated to that branch.

The proposed heuristic is shown in Fig. 9. Line 4 finds x_ks such that the desired ratio between them, as discussed above, is fulfilled.

The slope for each branch is estimated as follows:

$$\text{slope}_k = \frac{y_{\max_k} - y_{\min_k}}{x_{\max_k} - x_{\min_k}} = \frac{\mu\left(T_{R_k}, p_{\text{inv}}\right) - 1}{T_{R_k}\lambda(p_{\text{inv}}) - 0}. \tag{18}$$

After finding the allocated input capacitances, m instances of the 1FO problem will be generated that can be optimally solved by the algorithm presented in Section III-A.

C. Merging Buffer Chains

So far, a continuous-sized buffer library has been assumed. In reality the ASIC library has a finite (and hopefully large) number of inverter sizes. So the solution needs to be mapped to one consistent with the library. The main problem when rounding the inverter sizes is that it may result in significant errors. To alleviate this problem, the merging

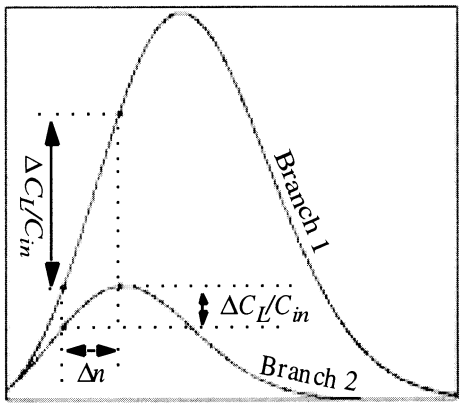

Fig. 8. Different slopes corresponding to different branches.

algorithm InCapAlloc (C_{in}, {C_L}, {T_R})
1. begin
2. **for** k=1,..., m
3. $C_k = \dfrac{C_{L_k}}{\mu(T_{R_k}, p_{inv})}$;
4. {x}=solve:

$$\begin{cases} \sum_{k=1}^{m} (C_k + x_k) = C_{in} \\ \forall k = 2, \ldots, m \\ \dfrac{x_1}{x_k} = \left(\dfrac{\mu(T_{R_1}, p_{inv}) - 1}{\mu(T_{R_k}, p_{inv}) - 1}\right) \times \left(\dfrac{T_{R_k}\lambda(p_{inv})}{T_{R_1}\lambda(p_{inv})}\right) \end{cases}$$

5. **for** k=1,..., m
6. $C_{ik} = C_k + x_k$;
7. **return** ({C_i});
8. end

Fig. 9. Algorithm *InCapAlloc*.

transformation, which is the opposite of the split transformation introduced in Fig. 4, is used.

To show how this works, recall Theorem 4. If the electrical efforts of the buffers on two branches are equal, one can merge them and replace them with a single buffer with the same electrical effort. Note that simply because the electrical efforts of the buffers are the same, one cannot conclude that the buffer sizes are also the same. As shown in Fig. 4, the sizes of each of the buffers before merging are C_1/h and C_2/h, respectively, and the size of the buffer after merging is $(C_1 + C_2)/h$. Therefore, the size of the buffer after merging is equal to the summation of buffer sizes before merging. This fact can be used to reduce the rounding error. As an example, consider a buffer size of 0.35 that has to be mapped to a buffer size of 1 in the ASIC library. Now, if two buffers of size 0.35 could be merged to a single buffer, the size would be 0.7, and rounding to a buffer size of 1 would result in smaller error.

Clearly, one has to be concerned about satisfying the required time and input capacitance constraints when performing this transformation. The merging should be performed in such a way that all timing constraints are satisfied and the area (as well as the input capacitance of the very first stage) is the same. As noted in the proof of Theorem 4, for the merging transformation to produce the exact same area and delay, the electrical efforts of the buffers to be merged must be equal. However, because each branch of the fanout tree is optimized separately with respect to the corresponding sink, the electrical efforts of the buffers may not necessarily be equal. Thus, a constant ε is defined and two buffers are merged if the difference between their electrical efforts is less than or equal to ε percent. In addition, two buffers are merged if the rounding error after merging the two is smaller than the summation of rounding errors of each buffer before the merge operation. Obviously, the efficiency of this approach is dependent on the order in which the buffers are selected to be merged. The approach presented here is to cluster the buffers into groups of nearly equal electrical efforts and check for the merging possibilities inside each group. Merging is performed starting at the source of the signal, and proceeding toward the sinks, while at the same time preserving the area so as not to increase the capacitive load imposed on the previous stage. The pseudocode for a recursive merging algorithm is shown in Fig. 10.

```
algorithm Merge (source)
1. begin
2.    B = all buffers driven by source;
3.    cluster buffers in B based on their electrical efforts;
4.    foreach cluster:
5.        repeat
6.            pick two buffers;
7.            merge if it improves the rounding error;
8.            add merged buffer to the cluster;
9.        until no more merging is possible;
10.   foreach buffer in every cluster:
11.       Merge (buffer);
12. end
```

Fig. 10. Algorithm *merge*.

TABLE I
COMPARISON WITH SUTHERLAND

Circuit	Sutherland		LEOPARD AREA	LEOPARD with 5% slack	
	Delay	Area		Delay	Area
1	6.97	233	232	7.32	183
2	6.86	19	19	7.20	15
3	15.05	458	455	15.80	277
4	12.85	183	182	13.49	123
5	8.13	22	22	8.53	17
6	11.32	143	142	11.89	97
7	6.86	38	38	7.20	30
8	12.20	198	197	12.81	134
9	13.79	245	245	14.48	149
10	8.50	70	69	8.93	54

IV. EXPERIMENTAL RESULTS

Three different sets of experiments were performed. In the first set, the LEOPARD algorithm of Section III was compared with an implementation of the Sutherland algorithm [4], which minimizes delay through a path. The results are reported in Table I.

TABLE II
COMPARISON WITH KUNG

Circuit	Sinks	Kung		LEOPARD	LEOPARD +5% InCap	
		InCap	Area	Area	InCap	Area
1	5	53.28	916	906	55.94	739
2	4	68.34	1104	1093	71.76	907
3	6	34.18	462	457	35.88	381
4	10	236.41	1463	1451	245.05	1231
5	4	153.45	1296	1284	161.12	1079
6	7	156.40	1635	1619	164.22	1347
7	15	158.62	5358	5295	166.55	4210
8	12	29.24	4342	4290	30.70	3376
9	9	21.25	3868	3820	22.31	2995
10	11	21.28	5808	5735	22.34	4461

TABLE III
COMPARISON WITH SIS

		LEOPARD cont. sizing	LEOPARD discrete sizing		SIS
Circuit	Sinks	Σcap	Σcap	Area	Area
1	12	0.093	0.093	3920	5281
2	6	0.032	0.039	3902	4676
3	21	0.065	0.088	6090	20952
4	14	0.093	0.093	3920	5281
5	21	0.060	0.089	7220	11952
6	12	0.045	0.062	4814	7857
7	16	0.087	0.110	6327	12315

For all of the experiments, the minimization problems within the LEOPARD algorithm were solved using the Matlab Optimization Toolbox v. 2.0. Furthermore, p_{inv} was assumed to be 0.6. For each circuit, the capacitive load of the sink and the maximum capacitance that the source can drive were given. First, the path delay was minimized using Sutherland's method. Delay and area of minimum-delay buffer chain are reported in columns 2 and 3. Next, the resulting delay and polarity were used as the constraints for the area minimization problem in LEOPARD. In the 4th column, the minimum area generated by LEOPARD, subject to the given constraints, is shown. As expected, the area is almost the same because delay has been minimized and, hence, the timing constraint is so tight there will not be much room for reducing area. However, when LEOPARD was given a 5% additional slack, it can reduce area by an average of 29% as shown in columns 6 and 7. This shows how delay can be traded off for area to significantly reduce area using LEOPARD if a slight increase in delay can be afforded. Note that merging or rounding is not applied during this set of experiments and the area reported is the summation of input capacitances of all inverters.

In the next set of experiments, the results from LEOPARD are compared with the results of an implementation of Kung's algorithm [3].

For each circuit, a number of sinks with capacitive load, required time, and required polarity are given. The number of sinks for each circuit is shown in column 2. Kung's algorithm was first used to minimize capacitive load on the source. The resulting capacitance and area are reported in columns 3 and 4. The capacitance calculated by Kung's algorithm was then used as the capacitive constraint for area optimization in LEOPARD. The resulting area is reported in column 5. Finally, an additional 5% input capacitance was allowed for each circuit to further reduce area, and the resulting input capacitance and area are shown in columns 6 and 7. An average of 19% improvement in area is achieved in the expense of 5% additional input capacitance. Note that in this set of experiments, neither merging nor rounding were performed for Kung's algorithm or LEOPARD and the area reported in Table II is the total capacitance of inverters calculated by the algorithms rather than extracted from the library. p_{inv} is assumed to be 0.6.

Finally, our last set of experimental results compare LEOPARD with the sequential interactive synthesis (SIS) fanout optimization program. SIS runs different fanout optimization programs, namely *LT-Tree*, *Two-Level*, *Bottom-Up*, and *Balanced*, and the best one is reported [14]. In this set of experiments, a standard cell library consisting of ten different inverters was used. For each inverter, $\tau_{\mathrm{intrinsic}}$ and R_{out} were specified for the SIS library delay model and p_{inv} and τ were specified for the Sutherland delay model. A very good match between the SIS delay and logical effort delay model values was enforced.

The fanout optimization programs of SIS were first used to perform fanout optimization. The results are reported in column 6 of Table III. Then, the delay and input capacitance resulting from SIS were used as constraints for LEOPARD. The results, assuming a continuous-size buffer library, are reported in column 3. Then, merging and mapping to the real buffers in the ASIC library were performed, and the results are shown in columns 4 and 5. As shown in the table, in case of continuous sizing the area is expressed in terms of the capacitances but for the discrete-sized buffers, it is the actual buffer area extracted from the library. Results show an average of 38% area improvement for LEOPARD.

V. CONCLUSION

This paper presented an optimal algorithm for buffer chains to minimize area with the assumption of continuous sizing for the buffers. The algorithm finds the optimum number of buffers and the optimum sizing for them by solving a posynomial minimization problem subject to posynomial inequality constraints which can be easily and quickly solved by a convex program solver. Based on this algorithm, a heuristic method was presented for the general case of buffer trees. Considering the fact that the number of discrete sizes for buffers in typical libraries has highly increased, the assumption of near-continuous buffer library is fairly accurate.

REFERENCES

[1] C. L. Berman, J. L. Carter, and K. F. Day, "The fanout problem: from theory to practice," in *Advanced Research in VLSI: Proceedings of the 1989 Decennial Caltech Conferences*, C. L. Seitz, Ed. Cambridge, MA: MIT Press, 1989, pp. 69–99.

[2] K. Kodandapani, J. Grodstein, A. Domic, and H. Touati, "A simple algorithm for fanout optimization using high-performance buffer libraries," in *Proc. Int. Conf. Computer-Aided Design*, 1993, pp. 466–471.

[3] D. S. Kung, "A fast fanout optimization algorithm for near-continuous buffer libraries," in *Proc. 35th Design Automation Conf.*, 1998, pp. 352–355.

[4] I. E. Sutherland and R. F. Sproull, "Logical effort: Designing for speed on the back of an envelope," in *Advanced Research in VLSI*. Santa Cruz, CA: Univ. of Calif., 1991.

[5] R. M. Corless, G. H. Gonnet, D. E. G. Hare, D. J. Jeffrey, and D. E. Knuth, "On the Lambert W function," *Adv. Computat. Math.*, vol. 5, pp. 329–359, 1996.

[6] P. M. Vaidya, "A new algorithm for minimizing convex functions over convex sets," in *Proc. IEEE Foundations Comput. Sci.*, Oct. 1989, pp. 332–337.

[7] C. Mead and L. Conway, *Introduction to VLSI Systems*. Reading, MA: Addison Wesley, 1980.
[8] P. Rezvani, A. Ajami, M. Pedram, and H. Savoj, "LEOPARD: A logical effort-based fanout optimizer for area and delay," in *Proc. Int. Conf. Computer-Aided Design*, Nov. 1999, pp. 516–519.
[9] M. C. Golumbic, "Combinational merging," *IEEE Trans. Comput.*, vol. 25, pp. 1164–1167, Nov. 1976.
[10] K. J. Singh and A. Sangiovanni-Vincentelli, "A heuristic algorithm for the fanout problem," in *Proc. 27th Design Automation Conf.*, June 1990, pp. 357–360.
[11] A. Salek, J. Lou, and M. Pedram, "A simultaneous routing tree construction and fanout optimization algorithm," in *Proc. Int. Conf. Computer-Aided Design*, Nov. 1998, pp. 625–630.
[12] P. Cocchini, M. Pedram, G. Piccinini, and M. Zamboni, "Fanout optimization under a submicron transistor-level delay model," *IEEE Trans. Computer-Aided Design*, vol. 9, pp. 339–349, Mar. 1990.
[13] Y.Yu Nesterov and A. Nemirovsky, *Interior point polynomial methods in convex programming*. Philadelphia, PA: SIAM, 1994.
[14] H. J. Touati, "Performance-Oriented Technology Mapping," Ph.D. dissertation, Univ. California, Berkeley, 1990.

Reliability-Constrained Area Optimization of VLSI Power/Ground Networks Via Sequence of Linear Programmings

Sheldon X.-D. Tan, C.-J. Richard Shi, and Jyh-Chwen Lee

***Abstract*—This paper presents a new method of sizing the widths of the power and ground routes in integrated circuits so that the chip area required by the routes is minimized subject to electromigration and *IR* voltage drop constraints. The basic idea is to transform the underlying constrained nonlinear programming problem into a sequence of linear programs. Theoretically, we show that the sequence of linear programs always converges to the optimum solution of the relaxed convex optimization problem. Experimental results demonstrate that the proposed sequence-of-linear-program method is orders of magnitude faster than the best-known method based on conjugate gradients with constantly better solution qualities.**

***Index Terms*—Circuit modeling, linear programming, power distribution network, simulation and optimization.**

I. Introduction

Power/ground (P/G) networks connect the P/G supplies in the circuit modules to the P/G pads on a chip. An important problem in P/G network design is to use the minimum amount of chip area for wiring P/G networks, while avoiding potential reliability failures due to electromigration and excessive *IR* drops. Specifically, we are concerned with the problem of P/G-network optimization where the topologies of P/G networks are assumed to be fixed, and only the widths of wire segments are to be determined. Several methods have been developed to solve this problem [6]–[9]. However, to the best of our knowledge, none of these methods have been incorporated into commercial computer-aided design (CAD) tools and used by industry.

One major obstacle is that these methods are based on constrained nonlinear programming, a process known to be computationally intensive (NP-hard) [12]. These methods are applicable only to small size problems, while P/G networks in today's very large scale integration (VLSI) design may contain millions of wire segments (therefore, millions of variables). On the other hand, with the continuous shrinking of the chip feature size, P/G network optimization is becoming increasingly important, since more and more portions of the chip area are dedicated to P/G routings, and the problems of *IR* drop and electromigration deteriorate.

In this paper, we present a new method capable of solving the P/G optimization problem orders of magnitude faster than the best known method. Our method is inspired by a key observation made by Chowdhury that if currents in wire segments are fixed, and voltages are used as variables, then the resulting optimization problem is convex [8]. However, instead of using the conjugate gradient method as in [8], we show that the problem can be solved elegantly by a sequence of linear programs. We prove that there always exists a sequence of linear programs that converge to the optimal solution of the original convex optimization problem. Experimental results have demonstrated that usually a few linear programs are required to reach the optimal solution. The complexity of the proposed method is proportional to the complexity of linear programming (which can be solved in polynomial time [5], [12]). Therefore, our method is scalable, i.e., the CPU time increases approximately polynomially with the size of a network. In practice, we have observed that the new method is orders of magnitude faster than the conjugate gradient method with constantly better optimization results.

This paper is organized as follows. Section II reviews some previous work. Section III describes the formulation of the P/G network optimization problem. The new method is presented in Section IV. Some practical considerations are described in Section V. Experimental results from some large P/G networks are summarized in Section VI. Section VII concludes the paper.

II. Previous Work

It is generally assumed that the average current drawn by each module is known and is modeled as an independent current source (we do not consider the temporal correlations of current sources). The constraints from reliability and design rules include: 1) *IR* voltage drop constraints; 2) metal-migration constraints; 3) minimum width constraints; and 4) equal width constraints. The problem of determining the widths of wire segments of a P/G network to minimize the total P/G routing area subject to all these constraints is a constrained nonlinear optimization problem [6], [7].

In the method of Chowdhury and Breuer [6], resistance values and branch currents are selected as independent variables. Both the objective function and the *IR* voltage drop constraints become nonlinear. The augmented Lagrangian method combined with the steepest descent algorithm [1] is used to solve the resulting problem.

Dutta and Marek-Sadowska [9] used only resistance values as variables. All of the constraints expressed in terms of nodal (terminal) voltages and branch currents, which have to be obtained by explicitly solving an electrical network, become nonlinear. The feasible direction method [4] is employed to solve the nonlinear optimization problem. At each iteration step, extra effort is required to solve the electrical network for nodal voltages and branch currents, as well as their gradients by numerical differentiation.

Manuscript received August 17, 2002; revised February 3, 2003. Some preliminary results of this paper were presented at the ACM/IEEE 38th Design Automation Conference, New Orleans, LA, June 1999. This paper was recommended by Associate Editor M. Sarrafzadeh.

S. X.-D. Tan is with the Department of Electrical Engineering, University of California, Riverside, CA 92521 USA (e-mail: stan@ee.ucr.edu).

C.-J. R. Shi is with the Department of Electrical Engineering, University of Washington, Seattle, WA 98195 USA.

J.-C. Lee is with Synopsys Inc., Mountain View, CA 94043 USA.

Digital Object Identifier 10.1109/TCAD.2003.819429