Hybrid Dynamic/Quadratic Programming Algorithm for Interconnect Tree Optimization

Yu-Yen Mo and Chris Chu

Abstract—**We present an algorithm for delay minimization of interconnect trees by simultaneous buffer insertion/sizing and wire sizing in this paper. Both wire widths and buffer sizes are chosen from user-defined discrete sets. Our algorithm integrates the quadratic programming approach for handling a wire branch into the dynamic programming (DP) framework. Our experimental results show that our hybrid dynamic/quadratic programming algorithm is faster, more accurate, and uses considerably less memory than the pure DP approach.**

Index Terms—**Buffer insertion, interconnect, optimization, performance optimization, physical design, timing optimization.**

I. Introduction

As feature size becomes smaller and chip area becomes larger in integrated circuits, the importance of global interconnect delay increases rapidly with respect with the gate delay. As a result, interconnect delay at the global level has become a critical factor in determining the system performance of the deep submicrometer designs. New materials, such as copper and low dielectric constant (κ) materials, have been used to improve interconnect performance. However, at the global interconnect level, the benefit of material changes alone is insufficient to meet overall performance requirements. Even with the help of copper and low κ materials, it is predicted that interconnect delay is still likely to dominate the chip performance beyond the 0.18-μm technology [3]. Therefore, we can expect the significance of interconnect delay to rapidly increase in the near future.

In the past, gate delay was the dominant factor in determining circuit performance. Therefore, gate sizing and transistor sizing have been extensively studied in the literature [4]–[6]. As process technology has advanced, interconnect delay has played an increasingly important role in determining the performance of the circuit and, hence, wire sizing (WS) has recently become an active research topic [7]–[9]. In addition to sizing gates and wires, buffer insertion and buffer sizing have been proven effective in reducing delay and so have been extensively studied in the literature [10]–[13].

Since both buffer insertion and WS can optimize the performance of interconnects and their solutions can affect each other, several researchers have studied their simultaneous optimization. Chu and Wong [14] presented a closed form solution to solve the simultaneous buffer insertion/sizing and WS problem for a single wire segment. However, fringing capacitance and the bounds of wire width were not considered in their study. Later in [1], they proposed a quadratic programming (QP) approach for the simultaneous buffer insertion/sizing and WS problem. This approach handled fringing capacitance and discrete sizes of wires and buffers.

Lillis *et al.* [15] presented a dynamic programming (DP) algorithm to optimize an interconnect tree. Their algorithm is a generalization of the DP algorithm for buffer insertion by van Ginneken [10]. The algorithm was later extended to handle power dissipation and incorporate signal slew into the buffer delay model [2]. In order to obtain an accurate solution, the DP algorithms in [2] and [15] divide the wires into short segments, resulting in a large number of wire segments. For each wire segment, the set of all possible solutions for the whole downstream subtree must be computed and stored, which takes a lot of time and memory. To solve this shortcoming, Alpert and Devgan [11] tried to reduce the runtime using a wire segmenting technique. The idea was to trade off runtime with solution quality by using a coarser wire segmentation. Lai and Wong [16] tried using a recomputation technique to reduce the memory needed for the computation. Their idea was to trade memory against runtime by recomputing instead of storing values.

The algorithm presented in this paper is accurate, fast, and economical in its use of memory. This algorithm combines the DP framework of Lillis *et al.* [15] and the QP approach for interconnect optimization of a wire by Chu and Wong [1]. As shown in [1], the problem of simultaneous buffer insertion and WS for a wire can be formulated as a convex quadratic program and the convex quadratic program can be solved extremely efficiently using the active set method. In this paper, we use an approach similar to [1] to show that each wire branch can be handled as a whole—that is, without being divided into numerous segments. Therefore, the set of possible solutions of a wire branch can be found in less time and only one set per wire needs to be stored. To handle the tree structure (i.e., to combine the sets of solutions of adjacent wires together), DP is used. We call our hybrid algorithm dynamic/quadratic programming (DQP).

In addition, we present a constant reusing technique to more quickly solve the quadratic programs. To process the edges of the tree, a large number of quadratic programs must be solved. These quadratic programs are the same form, except for differences in some parameters such as downstream capacitance and wire length. We show that many constant values computed in one quadratic program can be stored and reused by other quadratic programs. Although this technique moderately increases the amount of memory used, it dramatically reduces runtime.

In this paper, the Elmore delay model [17] is used for delay calculation. A wire segment is modeled as a π-type model as shown in Fig. 1. A buffer is modeled as a switch-level *RC* circuit, as shown in Fig. 2.

The remainder of this paper is organized as follows. In Section II, we review the basic idea of the DP approach as in [2]. The QP formulation for a wire in [1] is reviewed in Section III. In Section IV, we present our hybrid DQP algorithm. We explain the modification that is needed to integrate the QP formulation into the DP framework. We also present the active set method and the constant reusing technique to solve the

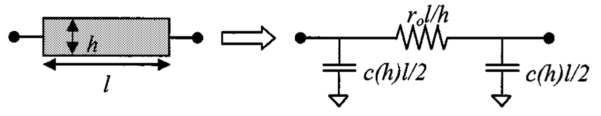

Fig. 1. Model of a wire segment of length l and width h by a π-type *RC* circuit. r_0 is the unit wire resistance. $c(h)$ is the wire capacitance per unit length for a segment of width h. In this paper, we assume that $c(h)$ is an increasing function.

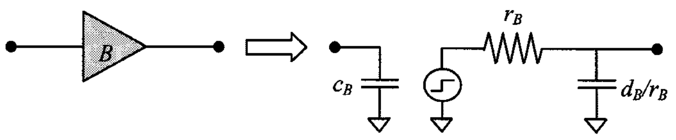

Fig. 2. Model of a buffer of size B by a switch-level *RC* circuit. c_B, r_B, and d_B are the input capacitance, output resistance, and the intrinsic delay, respectively.

Manuscript received November 1, 2000. This paper was recommended by Associate Editor D. Hill.

Y.-Y. Mo was with the Department of Electrical and Computer Engineering, Iowa State University, Ames, IA 95129 USA. He is now with Sun Microsystems, Palo Alto, CA 94303 USA.

C. Chu is with the Department of Electrical and Computer Engineering, Iowa State University, Ames, IA 95129 USA.

Publisher Item Identifier S 0278-0070(01)03080-9.

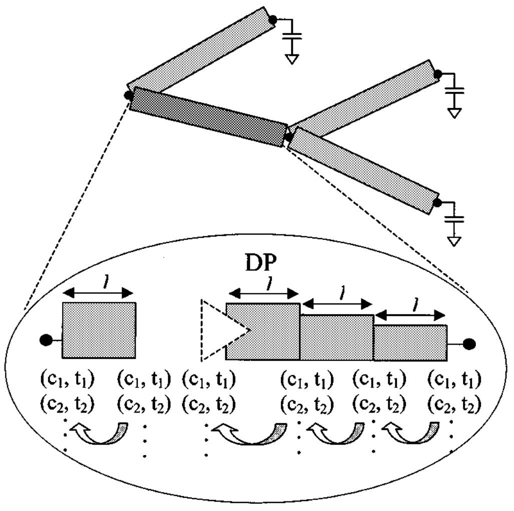

Fig. 3. Illustration of the DP approach for an interconnect tree.

quadratic programs. We present the experimental results in Section V. We then conclude this paper in Section VI.

II. Dynamic Programming Approach

In this section, we outline the DP technique for interconnect tree optimization in [2]. The algorithm adopts a bottom-up DP approach to minimize the maximum delay among all paths from source to sinks.

The basic idea of the DP technique is to build a new set of solutions for each segment based on the solution sets of its subtrees by traversing the tree structure in a bottom-up fashion. In the algorithm, instead of computing a single solution for each subtree, a set of solutions is computed and stored. Each member of the set is a downstream capacitance and delay time pair (c, t). The reason for doing so is that the optimal (c, t) combination for delay minimization cannot be determined without the upstream resistance. An illustration of the DP approach is shown in Fig. 3.

If there are two downstream branches for a node, each downstream branch will have a set of (c, t) pairs. These two sets can be combined into a single set and pruned according to the pair values. In the algorithm, the pruned list will have c in increasing order and t in decreasing order. If there are more than two downstream branches for a certain node, the node can be broken into several two-downstream-branch nodes with zero length in between.

The main drawback of the pure DP approach is that each wire must be divided into many small segments in order to achieve a quality solution. This drastically increases runtime and memory usage.

III. QP Approach

This section outlines the QP formulation of interconnect optimization for a single wire in [1].

We illustrate the idea by first considering WS alone. The extension to handle buffers is simple and will be presented afterwards. Chu and Wong [1] showed that the optimal wire shape could be described by a nonincreasing step function. Therefore, the WS problem can be formulated as follows. Given the wire length L, the driver resistance R_D, the load capacitance C_L, a set $H = \{h_1, \ldots, h_n\}$ of n choices of wire width such that $h_1 > \cdots > h_n$, WS is to determine the segment lengths $l_1, \ldots, l_n$ such that the delay from source to sink is minimized. WS is illustrated in Fig. 4. Note that this approach does not divide the wire into numerous segments. The number of segments is equal to the number of choices of segment width n, which is usually a small number.

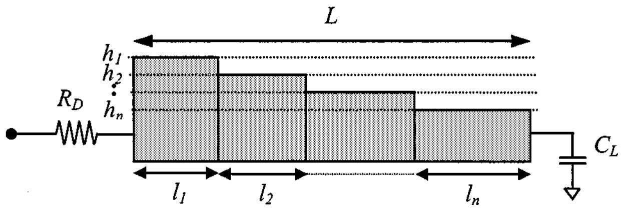

Fig. 4. WS problem for a single interconnect wire.

Let $c_i = c(h_i)$ for $1 \leq i \leq n$. For WS, the Elmore delay D for the wire is

$$\begin{aligned} D &= R_D(c_1 l_1 + c_2 l_2 + \cdots + c_n l_n + C_L) \\ &\quad + \frac{r_0 l_1}{h_1}\left(\frac{c_1 l_1}{2} + c_2 l_2 + \cdots + c_n l_n + C_L\right) \\ &\quad + \frac{r_0 l_2}{h_2}\left(\frac{c_2 l_2}{2} + c_3 l_3 + \cdots + c_n l_n + C_L\right) \\ &\quad \vdots \\ &\quad + \frac{r_0 l_n}{h_n}\left(\frac{c_n l_n}{2} + C_L\right) \\ &= \tfrac{1}{2}\boldsymbol{l}^T \Phi \boldsymbol{l} + \rho^T \boldsymbol{l} + R_D C_L \end{aligned}$$

where

$$\Phi = \begin{pmatrix} c_1 r_0/h_1 & c_2 r_0/h_1 & c_3 r_0/h_1 & \cdots & c_n r_0/h_1 \\ c_2 r_0/h_1 & c_2 r_0/h_2 & c_3 r_0/h_2 & \cdots & c_n r_0/h_2 \\ c_3 r_0/h_1 & c_3 r_0/h_2 & c_3 r_0/h_3 & \cdots & c_n r_0/h_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_n r_0/h_1 & c_n r_0/h_2 & c_n r_0/h_3 & \cdots & c_n r_0/h_n \end{pmatrix}$$

$$\rho = \begin{pmatrix} R_D c_1 + C_L r_0/h_1 \\ R_D c_2 + C_L r_0/h_2 \\ R_D c_3 + C_L r_0/h_3 \\ \vdots \\ R_D c_n + C_L r_0/h_n \end{pmatrix} \quad \text{and} \quad l = \begin{pmatrix} l_1 \\ l_2 \\ l_3 \\ \vdots \\ l_n \end{pmatrix}.$$

It was proved in [1] that the Hessian matrix Φ of the quadratic program is positive definite. Hence, the quadratic program is convex and polynomial-time solvable. Therefore, WS can be written as the following convex quadratic program:

$$\begin{array}{lll} \text{CQP:} & \text{Minimize} & 1/2\boldsymbol{l}^T \Phi \boldsymbol{l} + \rho^T \boldsymbol{l} \\ & \text{Subject to} & l_1 + \cdots + l_n = L \\ & & l_i \geq 0 \text{ for } 1 \leq i \leq n. \end{array} \tag{3.1}$$

Furthermore, it was also proved in [1] that Φ^{-1} is tridiagonal. In general, convex quadratic programs can be efficiently solved using a classical technique called active set method [18]. Each iteration of the active set method takes O(n^3) time. By making use of the property that Φ^{-1} is tridiagonal, [1] showed that each iteration of active set method for solving CQP can be done in only O(n) time. As a result, [1] presented an optimal algorithm to solve CQP, which runs in O(n^2) time in practice. Since n is usually a small number, the algorithm is extremely efficient in practice.

The extension of simultaneous buffer insertion and WS, as shown in [1], is straightforward. The corresponding quadratic program has exactly the same form as CQP above. The corresponding Hessian matrix is a block diagonal matrix; each block is the matrix Φ for WS above. Hence, it is positive definite and has a tridiagonal inverse. Therefore, the corresponding quadratic program can also be solved optimally using an active set method based O(mn^2) time algorithm, where m is the number of buffers inserted.

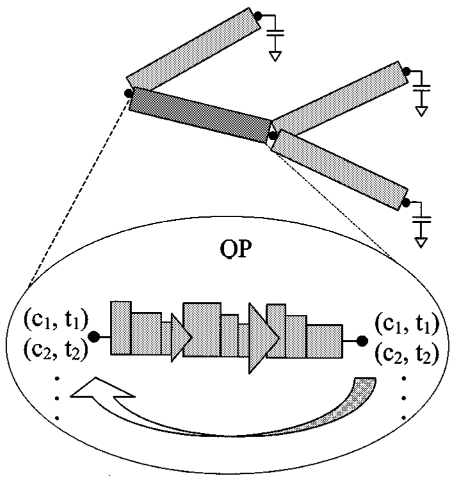

Fig. 5. Illustration of the DQP approach for an interconnect tree.

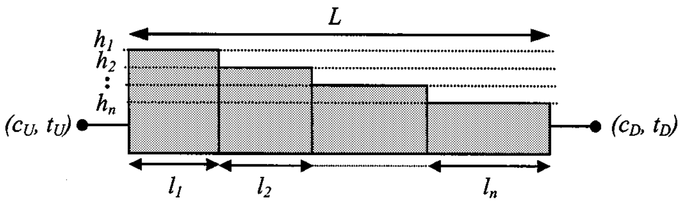

Fig. 6. New wire sizing problem WS' for a single wire branch in an interconnect tree.

Notice that the driver resistance R_D is assumed to be known in the QP formulation above. Since the upstream resistance is not known during the bottom-up DP traversal, it cannot be directly integrated into the DP framework. The modification needed is presented in the Section IV-B.

IV. Hybrid DQP Algorithm

This section introduces our hybrid DQP algorithm. To reduce runtime and memory usage, we integrated the QP approach into the DP framework. Instead of numerous small segments, each wire in the interconnect tree is handled as a whole by the QP approach. Fig. 5 illustrates the idea of this hybrid DQP approach.

In Section IV-A, we first present a modified QP formulation that can be integrated into the DP formulation. In Section IV-B, we present the active set method to solve the QP problem in Section IV-A. The modified DP framework and the DQP algorithm is presented in Section IV-C. Section IV-D introduces the constant reusing technique to more quickly solve the quadratic programs.

A. Modified Convex Quadratic Program

In this section, we modify the quadratic program CQP in Section III so that it can be integrated into the DP framework. We call the resulting quadratic program the modified convex quadratic program (MCQP). It is a building block of the DQP algorithm to handle a single wire branch.

The main difference between MCQP and CQP is that we ignore the driver resistance (R_D) in CQP and we include the upstream capacitance (c_U) into our formulation. First, consider the following new WS problem (WS') for a wire branch as shown in Fig. 6. The wire length L, the load capacitance c_D, and the set $H = \{h_1, \ldots, h_n\}$ of wire width choices are given as before. In addition, the delay time at the downstream node t_D (i.e., the delay time of the subtree at this node) and the capacitance seen from the upstream node of the branch c_U are also given. The objective is to minimize the upstream delay time t_U by changing the segment lengths $l_1, \ldots, l_n$. In other words, given a list of (c_D, t_D) pairs at the downstream node, MCQP can be used to find a list of (c_U, t_U) pairs at the upstream node. This is similar to the DP approach.

Consider a particular (c_D, t_D) and c_U combination. Let $c_i = c(h_i)$ for $1 \le i \le n$. The delay t_U for this wire branch is

$$\begin{aligned} t_U = {} & \frac{r_0 l_1}{h_1}\left(\frac{c_1 l_1}{2} + c_2 l_2 + \cdots + c_n l_n + c_D\right) \\ & + \frac{r_0 l_2}{h_2}\left(\frac{c_2 l_2}{2} + c_3 l_3 + \cdots + c_n l_n + c_D\right) \\ & \vdots \\ & + \frac{r_0 l_n}{h_n}\left(\frac{c_n l_n}{2} + c_D\right) + t_D \\ = {} & \tfrac{1}{2}\boldsymbol{l}^T \Phi \boldsymbol{l} + \rho^T \boldsymbol{l} + t_D \end{aligned}$$

where

$$\Phi = \begin{pmatrix} c_1 r_0/h_1 & c_2 r_0/h_1 & c_3 r_0/h_1 & \cdots & c_n r_0/h_1 \\ c_2 r_0/h_1 & c_2 r_0/h_2 & c_3 r_0/h_2 & \cdots & c_n r_0/h_2 \\ c_3 r_0/h_1 & c_3 r_0/h_2 & c_3 r_0/h_3 & \cdots & c_n r_0/h_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_n r_0/h_1 & c_n r_0/h_2 & c_n r_0/h_3 & \cdots & c_n r_0/h_n \end{pmatrix}$$

$$\rho = \begin{pmatrix} c_D r_0/h_1 \\ c_D r_0/h_2 \\ c_D r_0/h_3 \\ \vdots \\ c_D r_0/h_n \end{pmatrix} \quad \text{and} \quad \boldsymbol{l} = \begin{pmatrix} l_1 \\ l_2 \\ l_3 \\ \vdots \\ l_{n1} \end{pmatrix}.$$

Therefore, WS' can be formulated as the following MCQP:

$$\begin{aligned} \text{MCQP:}\quad & \text{Minimize} && t_U = 1/2\boldsymbol{l}^T \Phi \boldsymbol{l} + \rho^T \boldsymbol{l} + t_D \\ & \text{Subject to} && l_1 + \cdots + l_n = L \\ & && c_1 l_1 + \cdots + c_n l_n + c_D = c_U \\ & && l_i \ge 0 \text{ for } 1 \le i \le n. \end{aligned} \tag{4.1}$$

Notice that the matrix Φ here is the same as the one in CQP. Hence, it is positive definite and has a tridiagonal inverse.

As shown in the original QP approach in [1], the above formulation can be easily extended to handle simultaneous buffer insertion and WS. For fixed (c_D, t_D) and c_U values, each combination of the number of buffers and buffer sizes corresponds to one instance of MCQP. However, if buffers of different sizes are considered, many instances of MCQP need to be solved. Suppose there are q different choices of buffer sizes in the buffer library and m buffers are inserted. Then there are q^m choices of buffer sizes and, hence, q^m instances of MCQP to solve. The algorithm will be slow if m is large.

In order to guarantee that only a small number of buffers will be inserted in each wire, we divide each long wire into several wires shorter than a critical length. The critical length is defined as the maximum length such that at most one buffer is needed in the optimal solution [19]. This length depends on the technology parameters and the bounds on wire width and buffer size and can be determined experimentally. Using this idea, for fixed (c_D, t_D) and c_U values, only $1 + q$ instances (one instance for no buffer and one instance for each of the q buffer sizes) need to be considered. The simultaneous buffer insertion and WS problem for one buffer of size B is shown in Fig. 7.

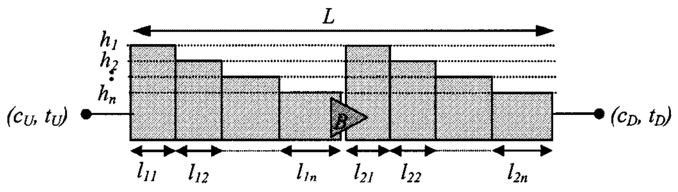

Fig. 7. Simultaneous buffer insertion and WS problem.

B. Extended Active Set Method

In this section, we use the idea of active set method to derive a very efficient algorithm to solve MCQP.

If a convex quadratic program consists of equality constraints only, it is particularly easy to solve. Consider the following program:

$$\begin{array}{ll}\text{Minimize} & 1/2\boldsymbol{l}^T\Phi\boldsymbol{l} + \rho^T\boldsymbol{l} \\ \text{Subject to} & \Gamma\boldsymbol{l} = b\end{array} \tag{4.2}$$

where Φ is positive definite and Γ is of full rank. Consider the associated Lagrangian:

$$\mathrm{L}(\boldsymbol{l}, \lambda) = 1/2\boldsymbol{l}^T\Phi\boldsymbol{l} + \rho^T\boldsymbol{l} + \lambda^T(\Gamma\boldsymbol{l} - \boldsymbol{b}). \tag{4.3}$$

The Lagrange necessary conditions of optimality are $\partial\mathrm{L}(\boldsymbol{l}, \lambda)/\partial l_i = 0$ and $\partial\mathrm{L}(\boldsymbol{l}, \lambda)/\partial\lambda_i = 0$ for all i. The conditions can be written in matrix form as follows:

$$\begin{aligned}\Phi\boldsymbol{l} + \Gamma^T\lambda + \rho &= 0 \\ \Gamma\boldsymbol{l} - \boldsymbol{b} &= 0.\end{aligned} \tag{4.4}$$

Since Φ is positive definite and Γ is of full rank, it can be shown that the conditions can be uniquely solved

$$\begin{aligned}\lambda &= -(\Gamma\Phi^{-1}\Gamma^T)^{-1}(\Gamma\Phi^{-1}\rho + \boldsymbol{b}) \\ \boldsymbol{l} &= -\Phi^{-1}\Gamma^T\lambda - \Phi^{-1}\rho.\end{aligned} \tag{4.5}$$

CQP and MCQP also consist of inequality constraints. It has been shown in [1] that inequality constraints in CQP can be handled efficiently by the active set method, which is a popular and efficient technique for solving QP problems. The idea underlying the active set method for solving a general convex quadratic program is to partition the inequality constraints into two groups: active and inactive. In each iteration, the active inequality constraints are treated as equality constraints and the inactive constraints are essentially ignored. Then, the resulting equality constrained program is solved. If the solution is infeasible with respect to the original program, some inactive constraints are added to the set of constraints. If the solution is feasible but not optimal (i.e., some Lagrangian multipliers are negative for the minimization problem), some constraints are removed from the current active set. The process is repeated until the optimal solution is found. We give a brief outline on using active set method to solve QP in the following. Readers are encouraged to read [18, Ch. 11] for more details of active set method.

The same idea used in [1] to solve CQP can be applied to MCQP. The MCQP problem has an extra equality constraint on c_U. So, the major difference between solving MCQP and CQP is that at least two wire segments need to be inactive at any time in the active set method. This is because we have two equality constraints (i.e., the total wire length constraint and the upstream capacitance constraint) that need to be satisfied. To ensure the feasibility of the solution, we need to start the

```
1.  Find an initial feasible solution for $\boldsymbol{l}$.
2.  Set the active set $\boldsymbol{A} = \varnothing$.
3.  Solve for $\boldsymbol{l}'$ with respect to $\boldsymbol{A}$ as in CQP.
4.  Calculate $\boldsymbol{d} = \boldsymbol{l}' - \boldsymbol{l}$.
5.  If ($\boldsymbol{l}'$ is not feasible)
6.      Calculate $\alpha_k$ according to $\boldsymbol{d}$ and $\boldsymbol{l}$ (where $\alpha_k$ is the step size
        which is selected to be as large as possible to maintain
        feasibility).
7.      Calculate new $\boldsymbol{l} = \boldsymbol{l} + \boldsymbol{d}\,\alpha_k$.
8.      Add the $l$ element which gives $\alpha_k$ to $\boldsymbol{A}$. Go back to step 3.
9.  Else if ($\boldsymbol{l}'$ is feasible)
10.     If ($\boldsymbol{d} \neq 0$)
11.         Update $\boldsymbol{l} = \boldsymbol{l}'$ and go back to step 3.
12.     Else if ($\boldsymbol{d} = 0$)  /* local minimal reached */
13.         Solve for $\lambda$.
14.         If ($\lambda < 0$)  /* check for optimality */
15.             Drop all the $l$ which correspond to negative $\lambda$ from $\boldsymbol{A}$.
16.             Go back to step 3.
17.         Else if ($\lambda \geq 0$)
18.             Optimal $\boldsymbol{l}$ obtained.
```

Fig. 8. EASM algorithm.

active set method with a feasible initial solution for vector $\boldsymbol{l}$. Then, we iteratively calculate a new solution $\boldsymbol{l}'$ which will be the new direction to move $\boldsymbol{l}$. The idea is to move $\boldsymbol{l}$ stepwise toward the optimal solution and make sure that each step stays within the feasible region. Fig. 8 summarizes the algorithm extended active set method (EASM), which we used to solve MCQP.

C. Hybrid Algorithm

The DP approach discussed in Section III-A is the basic framework of our DQP algorithm. The DP idea is used to handle the tree structure of interconnects (i.e., to combine the set of solutions of adjacent subtrees together). We do not divide the wires into a lot of segments and then handle the segments by DP. Instead, with the QP approach, each wire branch is handled as a whole.

However, since we do not know the upstream resistance at a node during the bottom-up traversal of the DP, we need to consider many different c_U values and calculate the optimal delay t_U corresponding to each c_U value. In general, except for the leaf nodes (the nodes which connect to sinks), each wire branch can have more than one (c_D, t_D) pair at the downstream node and a set of c_U at the upstream node. Each combination of c_U and (c_D, t_D) forms a MCQP problem instance.

Let N_c be a user-defined parameter specifying the number of different c_U values used at each node. Let q be the number of choices of buffer sizes. For each wire branch, there are $1+q$ cases to consider (one case for no buffer and one case for each of the q buffer sizes). For each case, the set of c_U is chosen by first determining an upper bound and a lower bound on the upstream capacitance. Then $N_c/(q+1)$ discrete values are selected evenly from the range. For the case without buffer, the upper bound of c_U is the MAX(c_D) plus the wire capacitance of this wire branch with maximum wire width only. The lower bound of c_U is the MIN(c_D) plus the wire capacitance of this wire branch with minimum wire width only. For the cases with buffer, the upper bound of c_U is wire capacitance of this wire branch with maximum wire width only, plus the input capacitance of the buffer (insert the buffer at the downstream node). The lower bound of c_U is simply the input capacitance of the buffer (insert the buffer at the upstream node). Note that

```
1.  Start from a leaf node. /* the node which connect to a sink */
2.  Append a branch of wire to the upstream of this node.
3.  Apply EASM to calculate an upstream delay time $t_U$ for each
    selected upstream capacitance $c_U$ according to all available
    downstream delay time $t_D$, and capacitance $c_D$.
4.  If (current node is the root)
5.      Search for the smallest delay according to the driver resistance
        and (c, t) pair set at the root.
6.      Construct the solution top-down and then exit the program.
7.  Else if (number of children of a node = 2)
8.      Merge the two (c, t) lists and prune redundant (c, t) pairs.
9.      Go back to step 2.
10. Else if (number of children of a node = 1)
11.     ($c_D$, $t_D$) set of the current node = ($c_U$, $t_U$) set of the child of the
        current node.
12.     Go back to step 2.
```

Fig. 9. Hybrid DQP algorithm.

because we are using discrete buffer sizes, these c_U ranges may not overlap each other.

After the root of the interconnect tree is reached, the interconnect optimization solution can be constructed by a top-down traversal. With the knowledge of the driver resistance, we can select the best (c, t) pair at the root, which results in minimum delay time. The rest of the solution can be obtained by recursively traversing the tree in a top-down manner and selecting the best (c, t) pair at each node.

The DQP algorithm is summarized in Fig. 9. Section IV-D introduces a constant reusing technique to more quickly solve the MCQP instances in Step 3.

D. Constant Reusing Technique

In this subsection, we present a constant reusing technique to increase the speed of the DQP algorithm.

The algorithm EASM can be applied directly in DQP to solve the MCQP instances for all wires. However, for each wire branch, there are many (c_D, t_D) pairs at the downstream node, many c_U values at the upstream node and several choices of buffer size. Therefore, there can be a lot of MCQP instances. If those MCQP instances are solved independently by EASM, experimental results show that the DQP approach has only a very limited runtime improvement over the pure DP approach. However, we recognize that all MCQP instances are the same except that the parameters L, c_U, c_D, and t_D are different. Many constant values computed in one MCQP instance can be stored and reused by other MCQP instances.

By observing (4.5), we recognize that $\boldsymbol{l}$ and λ can be expressed by two separate linear functions in terms of downstream capacitance c_D, total length of the wire branch L, and upstream capacitance c_U. Upstream delay time t_U can be expressed by a quadratic function in terms of c_D, L, c_U and downstream delay time t_D as shown in the following:

$$\lambda = \boldsymbol{v}_{\lambda 1} c_D + \boldsymbol{v}_{\lambda 2} L + \boldsymbol{v}_{\lambda 3} c_U, \qquad \boldsymbol{l} = \boldsymbol{v}_{l1} c_D + \boldsymbol{v}_{l2} L + \boldsymbol{v}_{l3} c_U$$

and

$$t_U = v_{t1} c_U^2 + v_{t2} L^2 + v_{t3} c_D^2 + v_{t4} c_U L + v_{t5} L c_D + v_{t6} c_D c_U + t_D. \tag{4.6}$$

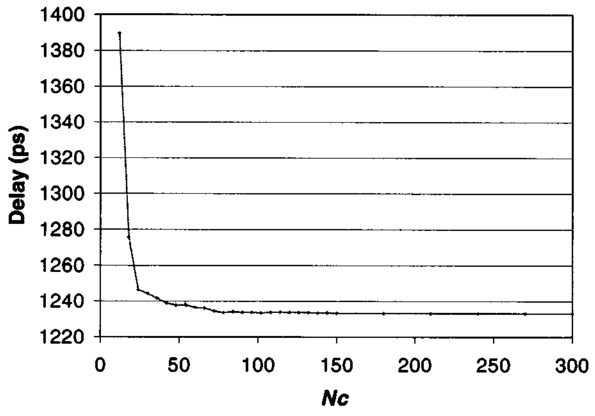

Fig. 10. Delay versus N_c for DQP.

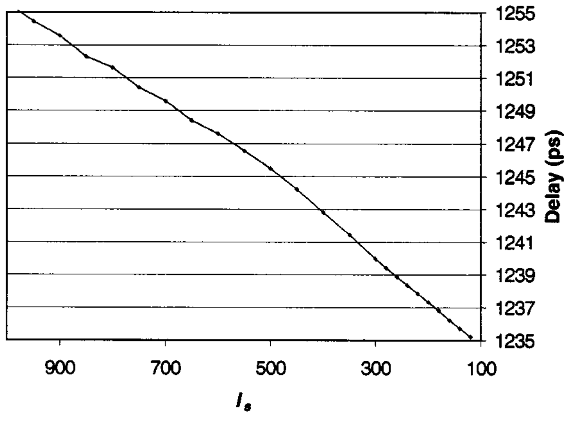

Fig. 11. Delay versus l_s for DP.

All v_{ti}'s, $\boldsymbol{v}_{\lambda i}$'s, and $\boldsymbol{v}_{ti}$'s in (4.6) are independent of L, c_U, c_D, and t_D. Their values depend only on the electrical parameters (e.g., r_0, c_0, and c_f) and the set of wire widths and buffer sizes that are being used. In the active set method, some segments are set to inactive at each iteration. Hence, the set of wire widths being used may only be a subset of H. So, when a particular set of wire widths is first used in some iteration of the active set method, the constants v_{ti}, $\boldsymbol{v}_{\lambda i}$, and $\boldsymbol{v}_{ti}$ can be computed and stored. When the same set of wire widths is used again in another iteration, the stored constants can be reused. This constant reusing technique only moderately increases the memory usage but dramatically reduces the runtime. Please refer to the Appendix for the equations to calculate constants v_{ti}, $\boldsymbol{v}_{\lambda i}$, and $\boldsymbol{v}_{ti}$.

V. Experimental Results

The DQP algorithm was implemented as a C program. We tested the program on a PC with a 500-MHz Pentium III processor and 256-MB of memory (with RedHat Linux 6.0 operating system). We used the parameters for the 0.18-μm technology listed in [20]. The results were compared with the pure DP approach. We used six trees with 2–100 sinks. The length of the tree wires range from 1000–15 000 μm.

Fig. 10 shows the delay time of the solutions obtained by DQP versus the number of upstream capacitance choices N_c on the input tree with ten sinks. The results in Fig. 10 show that N_c need not be a large value to obtain a good solution quality. Fig. 11 shows a similar chart for the

TABLE I
COMPARISON BETWEEN DP AND DQP

# of Sinks	Optimal Delay (ps)	***DP*** ($l_s = 200$)				***DQP*** ($N_c = 90$)			
		Run Time (sec.)	Memory Used (Mb)	# of Buffers Inserted	Delay (ps)	Run Time (sec.)	Memory Used (Mb)	# of Buffers Inserted	Delay (ps)
2	293.72	1.77	6.17	2	294.70	0.27	0.00+0.9	2	295.59
5	666.60	6.44	22.12	9	668.83	0.76	0.01+0.9	8	668.53
10	1232.95	12.66	42.93	23	1237.34	1.41	0.02+0.9	18	1233.80
15	1430.64	20.82	70.35	40	1435.33	1.95	0.03+0.9	28	1431.50
30	1937.31	38.20	129.95	75	1939.95	4.09	0.05+0.9	57	1937.86
100	2256.82	N/A	N/A	N/A	N/A	13.24	0.17+0.9	189	2265.29

pure DP with the same input tree. The l_s is the segment length in micrometers that we used in DP to divide a wire branch into equal-length segments.

For the purpose of comparison, we set the parameters l_s and N_c such that the quality of solution for DP and DQP were similar. In our experiment, $l_s = 200$ and $N_c = 90$. The experimental results are shown in Table I.

The estimated optimal delay value was obtained by setting N_c to 1800 in DQP. In our experiment, DP with $l_s = 200$ could not be run successfully for the input file containing 100 sink nodes, possibly because of memory limitations on the PC that we used. The memory data column of DQP contains two parts. The first part is the memory dynamically allocated for (c, t) pairs. The second part is the memory used to store the constants being reused.

Our experimental results suggest that DQP is better than DP in runtime, memory, and accuracy (except for the delay of the smallest input tree). For small problems and for the same quality of solution, the runtime advantage of DQP over DP is relatively less. This is because of the overhead and the lower constant reusing rate of the EASM algorithm. However, for bigger problems, the runtime advantage of DQP over DP is more significant.

We also observe that the constant reusing technique can enhance the efficiency of our algorithm significantly. By applying the constant reusing technique, the runtime of our program can be reduced by around 75% compared to the implementation without constant reusing technique.

VI. DISCUSSION AND CONCLUSION

As the experimental results suggest, our hybrid DQP algorithm is faster and more accurate than the pure DP approach. It also requires less memory and can optimize a bigger interconnect tree.

In this paper, only the timing optimization of an interconnect tree is presented. However, interconnect power and area optimization can be easily applied to the same algorithm framework. Also, this paper assumes that buffer insertion can occur anywhere on the wires. In practice, however, we may not be able to insert a buffer in some parts of a wire due to certain conditions, such as space restriction or over-cell route. Such additional restrictions can be handled by adding linear constraints to the quadratic programs as in [1].

APPENDIX
EQUATIONS FOR CONSTANT REUSING TECHNIQUE

The WS' problem in Fig. 6 is used below to illustrate the idea. Extension to include buffer insertion is easy. Consider the following convex quadratic program formulation corresponding to an active set with wire width choices $\{h_1, \ldots, h_n\}$

$$\begin{array}{ll}\text{Minimize} & 1/2\boldsymbol{l}^T \Phi \boldsymbol{l} + \rho^T \boldsymbol{l} \\ \text{Subject to} & \Gamma \boldsymbol{l} = \boldsymbol{b}\end{array} \tag{a1}$$

where

$$\boldsymbol{\Gamma} = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ c_1 & c_2 & \cdots & c_n \end{pmatrix}, \qquad \boldsymbol{l} = \begin{pmatrix} l_1 \\ l_2 \\ \vdots \\ l_n \end{pmatrix}$$

and

$$\boldsymbol{b} = \begin{pmatrix} L \\ c_U - c_D \end{pmatrix}.$$

From (4.5) and (a1), we can derive the following:

$$\lambda = \begin{pmatrix} D - A \\ F - B \end{pmatrix} c_D - \begin{pmatrix} C \\ E \end{pmatrix} L - \begin{pmatrix} D \\ F \end{pmatrix} c_U \tag{a2}$$

where

$$\begin{pmatrix} C & D \\ E & F \end{pmatrix} = (\Gamma \Phi^{-1} \Gamma)^{-1}$$

$$\begin{pmatrix} A \\ B \end{pmatrix} = \begin{pmatrix} C \sum_{j=1}^{n} \sum_{i=j-1}^{j+1} (\theta_{ij} r_0 / h_j) + D \sum_{j=1}^{n} \sum_{i=j-1}^{j+1} (c_i \theta_{ij} r_0 / h_j) \\ E \sum_{j=1}^{n} \sum_{i=j-1}^{j+1} (\theta_{ij} r_0 / h_j) + F \sum_{j=1}^{n} \sum_{i=j-1}^{j+1} (c_i \theta_{ij} r_0 / h_j) \end{pmatrix}$$

and θ_{ij} is the element in Φ^{-1}

$$\boldsymbol{l} = \alpha c_D + \beta L + \gamma c_U \tag{a3}$$

where α, β, and γ are as shown in (a4) at the top of the next page, where

$$\sigma = \begin{pmatrix} r_0/h_1 \\ r_0/h_2 \\ \vdots \\ r_0/h_n \end{pmatrix}.$$

$$\alpha = \begin{pmatrix} -(D-A)\sum_{j=1}^{2}(\theta_{1j}) - (F-B)\sum_{j=1}^{2}(\theta_{1j}c_j) - \sum_{j=1}^{2}(\theta_{1j}\,r_0/h_j) \\ -(D-A)\sum_{j=1}^{3}(\theta_{2j}) - (F-B)\sum_{j=1}^{3}(\theta_{2j}c_j) - \sum_{j=1}^{3}(\theta_{2j}\,r_0/h_j) \\ -(D-A)\sum_{j=2}^{4}(\theta_{3j}) - (F-B)\sum_{j=2}^{4}(\theta_{3j}c_j) - \sum_{j=2}^{4}(\theta_{3j}\,r_0/h_j) \\ \vdots \\ -(D-A)\sum_{j=n-1}^{n}(\theta_{nj}) - (F-B)\sum_{j=n-1}^{n}(n_{nj}c_j) - \sum_{j=n-1}^{n}(\theta_{nj}\,r_0/h_j) \end{pmatrix}$$

$$\beta = \begin{pmatrix} C\sum_{j=1}^{2}(\theta_{1j}) + E\sum_{j=1}^{2}(\theta_{1j}c_j) \\ C\sum_{j=1}^{3}(\theta_{2j}) + E\sum_{j=1}^{3}(\theta_{2j}c_j) \\ C\sum_{j=2}^{4}(\theta_{3j}) + E\sum_{j=2}^{4}(\theta_{3j}c_j) \\ \vdots \\ C\sum_{j=n-1}^{n}(\theta_{nj}) + E\sum_{j=n-1}^{n}(\theta_{nj}c_j) \end{pmatrix} \quad \text{and} \quad \gamma = \begin{pmatrix} D\sum_{j=1}^{2}(\theta_{1j}) + F\sum_{j=1}^{2}(\theta_{1j}c_j) \\ D\sum_{j=1}^{3}(\theta_{2j}) + F\sum_{j=1}^{3}(\theta_{2j}c_j) \\ D\sum_{j=2}^{4}(\theta_{3j}) + F\sum_{j=2}^{4}(\theta_{3j}c_j) \\ \vdots \\ D\sum_{j=n-1}^{n}(\theta_{nj}) + F\sum_{j=n-1}^{n}(\theta_{nj}c_j) \end{pmatrix}$$

$$\begin{aligned} t_U &= 1/2\boldsymbol{l}^T\Phi\boldsymbol{l} + c_D\sigma^T\boldsymbol{l} + t_D = (1/2\alpha^T\Phi\alpha + \sigma^T\alpha)c_D^2 + (1/2\beta^T\Phi\beta)L^2 \\ &\quad + (1/2\gamma^T\Phi\gamma)c_U^2 + (\alpha^T\Phi\gamma + \sigma^T\gamma)c_Uc_D + (\alpha^T\Phi\beta + \sigma^T\beta)c_DL + (\beta^T\Phi\gamma)c_UL + t_D \end{aligned} \tag{a4}$$

References

[1] C. C. N. Chu and D. F. Wong, "A quadratic programming approach to simultaneous buffer insertion/sizing and wire sizing," *IEEE Trans. Computer-Aided Design*, vol. 18, pp. 787–798, June 1999.

[2] J. Lillis, C.-K. Cheng, and T.-T. Lin, "Optimal wire sizing and buffer insertion for low power and a generalized delay model," *IEEE J. Solid-State Circuits*, vol. 31, pp. 437–447, Mar. 1996.

[3] International Technology Roadmap for Semiconductors, Semiconductor Industry Association. (1999). [Online]. Available: http://www.itrs.net/ntrs/publntrs.nsf

[4] M. A. Cirit, "Transistor sizing in CMOS circuits," in *Proc. ACM/IEEE Design Automation Conf.*, June 1987, pp. 121–124.

[5] J. P. Fishburn and A. E. Dunlop, "TILOS: A posynominal programming approach to transistor sizing," in *Proc. ACM/IEEE Int. Conf. Computer-Aided Design*, Nov. 1985, pp. 326–328.

[6] S. S. Sapatnekar, V. B. Rao, P. M. Vaidya, and S. M. Kang, "An exact solution to the transistor sizing problem for cmos circuits using convex optimization," *IEEE Trans. Computer-Aided Design*, vol. 12, pp. 1621–1634, Nov. 1993.

[7] J. Cong and K.-S. Leung, "Optimal wiresizing under the distributed Elmore delay model," *IEEE Trans. Computer-Aided Design*, vol. 14, pp. 321–336, Mar. 1995.

[8] C.-P. Chen, H. Zhou, and D. F. Wong, "Optimal nonuniform wire-sizing under the Elmore delay model," in *Proc. ACM/IEEE Int. Conf. Computer-Aided Design*, Nov. 1996, pp. 38–43.

[9] S. S. Sapatnekar, "*RC* interconnect optimization under the Elmore delay model," in *Proc. ACM/IEEE Design Automation Conf.*, June 1994, pp. 387–391.

[10] L. P. P. P. van Ginneken, "Buffer placement in distributed rc-tree networks for minimal Elmore delay," in *Proc. Int. Symp. Circuits and Systems*, May 1990, pp. 865–868.

[11] C. Alpert and A. Devgan, "Wire segmenting for improved buffer insertion," in *Proc. ACM/IEEE Design Automation Conf.*, June 1997, pp. 588–593.

[12] S. Dhar and M. A. Franklin, "Optimal buffer circuits for driving long uniform lines," *IEEE J. Solid-State Circuit*, vol. 26, pp. 32–40, Jan. 1991.

[13] N. Hedenstierna and K. O. Jeppson, "CMOS circuit speed and buffer optimization," *IEEE Trans. Computer-Aided Design*, vol. CAD-6, pp. 270–281, Mar. 1987.

[14] C. C. N. Chu and D. F. Wang, "Closed form solution to simultaneous buffer insertion/sizing and wire sizing," in *Proc. ACM Int. Symp. Physical Design*, Apr. 1997, pp. 192–197.

[15] J. Lillis, C.-K. Cheng, and T.-T. Lin, "Optimal and efficient buffer insertion and wire sizing," in *Proc. Custom Integrated Circuit Conf.*, May 1995, pp. 259–262.

[16] M. Lai and D. F. Wong, "A memory-efficient implementation of dynamic programming for interconnect optimization," unpublished.

[17] W. C. Elmore, "The transient response of damped linear network with particular regard to wideband amplifiers," *J. Appl. Phys.*, vol. 19, pp. 55–63, 1948.

[18] D. G. Luenberger, *Linear and Nonlinear Programming*, 2nd ed. Reading, MA: Addison-Wesley, 1984.

[19] J. Cong and Z. Pan, "Interconnect delay estimation models for synthesis and design planning," in *Proc. Asia South Pacific Design Automation Conf.*, Jan. 1999, pp. 97–100.

[20] J. Cong, L. He, K.-Y. Khoo, C.-K. Koh, and Z. Pan, "Interconnect design for deep submicronics," in *Proc. ACM/IEEE Int. Conf. Computer-Aided Design*, Nov. 1997, pp. 478–485.