Sizing of clock distribution networks for high performance CPU chips

Madhav P. Desai
Radenko Cvijetic
James Jensen
Digital Equipment Corporation, Hudson MA

Abstract: In a high performance microprocessor such as Digital's 300MHz Alpha 21164, the distribution of a high quality clock signal to all regions of the device is achieved using a complex grid with multiple drivers. The large capacitance of this distribution grid together with the high clock frequency results in substantial power dissipation in the chip. In this paper, we describe techniques to size the interconnect segments (thus reducing their capacitance) of the distribution network while meeting certain design goals. These techniques place no restrictions on the topology of the network being sized, and have been successfully used on very large examples.

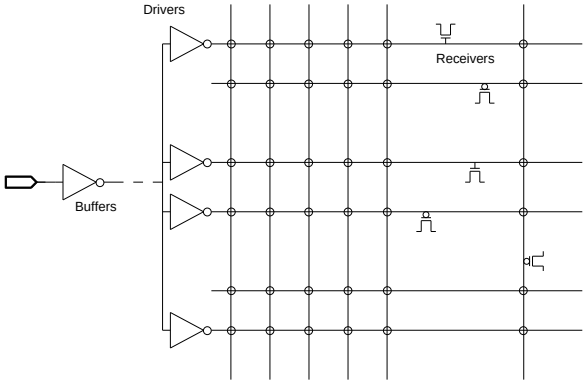

Figure 1: Clock grid

1 Introduction

The DC21164 microprocessor [4] is designed to operate at clock frequencies of more than 300MHz, and contains over 9 million devices. The distribution of such a high frequency clock signal throughout the chip is carried out using a complex grid with multiple synchronized drivers (Figure 1). Nearly half the power dissipated in the chip is due to the charging and discharging of the clock node capacitance (which includes the interconnect and device load capacitance).

The clock grid is designed to meet two goals:

1. The RC delay between the driver and any of the receivers is less than a specified $\delta > 0$ (this ensures that the clock skew between any two receivers is less than δ).
2. The average current in each metal segment is within an electro-migration limit.

The grid sizing problem is: minimize the interconnect capacitance (by resizing layout segments) while meeting these design goals. Recent papers have considered the problem of sizing a clock distribution network which is a *tree* (for example, see [7]). However the clock distribution network in a high performance design is chosen to be a grid for performance and design management reasons. In the DC21164[4], this clock grid has over a million segments, over 25000 drivers and over 200000 receivers. Consequently, there is a need for algorithms for sizing these large *non-tree* networks.

In this paper, we will describe some techniques for sizing such grids. We will assume that an initial clock grid which meets the design goals is available. Instead of obtaining an *optimal* resizing of this initial grid, we are interested in *efficient and scalable* sizing algorithms which will produce an appreciable capacitance reduction while maintaining the grid's design goals. We will achieve this by reformulating the sizing problem as a sequence of *network* problems, for which we present efficient solution techniques. Some of the results obtained from the sizing of very large grids in two production microprocessors from Digital will also be presented.

2 Preliminaries

The clock grid is modeled as a graph $G = (V, E)$, with vertex set $V = \{0, 1, 2, ..., N\}$ and edge set E. Each edge in E represents either a *metal segment* or a *contact* between two different layers in the layout of the grid. Let $D \subset V$ be the set of *driver nodes*, which are driven by an MOS device. Let $R \subset V$ be the *receiver nodes* which correspond to the loading points of the grid. We will assume that D and R are disjoint sets. The node 0 is considered to be the *ground* node.

An edge (i, j) in the grid has a *resistance* $R_{ij} > 0$, *capacitance* $C_{ij} \geq 0$ and an *electro-migration limit* $M_{ij} > 0$ associated with it. Each node i has a load capacitance $L_i \geq 0$ associated with it (if i is not a receiver node, $L_i = 0$). We will define the total capacitance at a node to be

$$C_i = L_i + \sum_{(i,j) \in E} C_{ij}/2 \quad (1)$$

33rd Design Automation Conference ®

DAC 96 - 06/96 Las Vegas, NV, USA

That is, each edge in the network is modeled by a π equivalent RC network (a long layout segment will be modeled by multiple sections). We will assume that the resistance of each segment is inversely proportional to its width. In general, the capacitance is a monotonically increasing function of the width. However, in the discussion that follows, we will assume that the capacitance of a segment is linearly dependent on its width.

If the width is considered as an explicit variable in the sizing problem, then we obtain a non-linear optimization problem, which can be solved effectively only for small networks[8] or for networks with limited topologies, such as trees[7]. Instead, we will formulate the sizing problem so that the widths are *implicit* variables, and are determined in terms of network variables. This formulation is described in more detail later in the paper.

2.1 Analysis of the Clock Grid

We will briefly outline the analysis algorithm used to measure the performance of the grid. We are mainly interested in the RC delay between the drivers and the receivers. A first order approximation of the grid will be used to estimate this delay. Assume that the grid is initially charged up to V_{DD} volts, the supply voltage. All the drivers in the grid are turned on at time $t = 0$[1], and the grid discharges to 0 volts. Then, the first order (single pole) [5] approximation of the voltage at node $i \neq 0$ in the circuit may be written as $V_{DD}e^{-t/\tau_i}$, where τ_i is the time constant for the discharge of node i. If $\tau = [\tau_i]$ is the vector of time constants, then

$$\tau \;=\; G^{-1}\mathbf{C} \tag{2}$$

where G is the $N \times N$ admittance matrix of the network (obtained using nodal analysis [1], and $\mathbf{C} = [C_i]$ is the vector of node capacitances. The RC delay to the node i will be approximated by τ_i.

Thus, the first order estimates of the waveforms at the nodes in the grid are obtained by solving a simple RI network consisting of resistors and current sources. The resistors in the RI network are the same as those in the original RC network. At each node i, a current source of value C_i is placed. The node potentials obtained by solving this RI network are the time constants of decay of the voltage waveforms at the nodes. This solution also yields information about the average current in each network segment. Define the *flow* in edge (i, j) from i to j as

$$x_{ij} \;=\; \frac{\tau_i - \tau_j}{R_{ij}} \tag{3}$$

Then, if the clock period of the network is T_{clk} and the duty cycle of the clock signal is 50%, then the average current in edge (i, j) during the discharge of the grid is

$$I_{ij} = \frac{2V_{DD}}{T_{clk}} x_{ij}$$

We will require $I_{ij} \leq M_{ij}$ to protect the grid from electromigration (EM) failures.

Remark: Assume that the edges in the RI network are directed from higher node potentials to lower node potentials (so that the flow in (3) is non-negative for each edge). Then, the flow values x_{ij} and the potential values τ_i satisfy the following equations. For each edge (i, j) directed from i to j,

$$\tau_i - \tau_j \;=\; R_{ij}x_{ij} \tag{4}$$

For each node i

$$\sum_{(i,j)\in E,\ i\to j} x_{ij} \;-\; \sum_{(k,i)\in E,\ k\to i} x_{ki} \;=\; C_i \tag{5}$$

Thus, given a set of flow and potential values in the network, the resistances of the edges (and hence their widths) are uniquely determined.

2.2 An approach to the sizing problem

Any reduction in the width of a layout segment will effectively increase its resistance (and decrease its capacitance). Such a change in the resistance will naturally alter the flow and potential patterns in the network. Conversely, given a set of flow and potential values in the network, the *effective* resistance of each edge can be determined, and this in turn can be used to determine the width reduction of that edge. For example: if an edge (i, j) has flow x_{ij} and its end nodes have potentials p_i, p_j, then its effective resistance is $R_{ij}^{eff} = (p_i - p_j)/x_{ij}$. This corresponds to scaling the width of (i, j) down by a factor of R_{ij}/R_{ij}^{eff} (because the resistance of the unscaled edge is R_{ij}). Thus, we can work with the flow and potential values and choose them to obtain a reduction of the effective capacitance of the grid while maintaining design goals.

We will assume that the edges in the grid are oriented so that

1. there is at least one directed path from each receiver to some driver, and
2. there are no directed cycles in the grid.

We obtain such an orientation by solving (2), and then directing each edge from the node with higher potential (τ) to the node with a lower potential. The primary benefit gained by this simplification is algorithmic efficiency, as later discussion will show. The network G may then considered to be a directed graph.

For each oriented edge $(i, j) \in E$, introduce a *flow* variable x_{ij}, and for each node i, introduce a *potential* variable p_i. A *feasible* flow in the network is a choice of flow and potential values such that the following conditions are met. For each edge (i, j), the flow variable must meet an electro-migration constraint:

$$0 \leq\; x_{ij} \;\leq\; M_{ij}\frac{T_{clk}}{2V_{DD}} \tag{6}$$

In addition, the resistance of the edge (i, j) is only allowed to increase (only reduction in widths are allowed):

$$p_i - p_j \;-\; R_{ij}x_{ij} \;\geq\; 0. \tag{7}$$

For each node i, we must impose a flow balance constraint (analogous to Kirchoff's current law):

$$\sum_{i\to j} x_{ij} - \sum_{k\to i} x_{ki} \;=\; C_i \tag{8}$$

Note that this constraint does not take into account the reduction in edge capacitance due to scaling, and thus is conservative. Finally, we impose the maximum RC delay constraint:

$$0 \leq\; p_i \;\leq\; \tau_{max} \tag{9}$$

[1] Connect each driver node to ground through a resistance modeling the driver.

where τ_{max} is the user-defined maximum RC-delay limit.

Given a feasible flow in the network, we define an edge $e = (i, j)$ to be *useful* with respect to the flow if the following conditions are met.

1. The flow $x_{ij} > C_{ij}/2$.
2. The edge e is contained in a directed path of edges (each satisfying the constraint in step 1), starting at some receiver and ending at the ground node.

In other words, an edge e is useful if it is helping to transport charge from some receiver to ground. If an edge is not useful, it serves no purpose in the clock grid and may be safely removed from the grid. Given a feasible flow, the width of each useful edge (i, j) may be scaled by a trim factor

$$t_{ij} = \frac{R_{ij}x_{ij}}{p_i - p_j}. \tag{10}$$

without violating the RC delay constraints. This trimming results in a reduction of the capacitance of the network.

2.3 The Sizing Algorithm

We will use the following algorithm to solve the clock sizing problem:

1. Solve the network (as in (2)) to obtain the initial τ values, and orient the edges from nodes with higher τ to nodes with lower τ.
2. Find a feasible flow that minimizes

$$\sum_{(i,j)\in E} C_{ij}x_{ij} \tag{11}$$

   We will call this problem the *flow redistribution* problem.

3. Keeping the values of x_{ij} obtained in step 2, choose the node potentials p_i to maximize

$$\sum_{(i,j)\in E} \frac{C_{ij}}{R_{ij}x_{ij}}(p_i - p_j) \tag{12}$$

   We will call this problem the *potential adjustment* problem.

4. Delete all edges which are not useful with respect to the flow and potential values obtained in steps 2,3. Scale the remaining edges using (10).

Each execution of steps 1-4 will be termed a *sweep* of the algorithm. It is of course possible to apply multiple sweeps of the algorithm to further reduce the interconnect capacitance.

Remark: Consider the scaled network obtained in Step 4 above. Suppose we solve this network as in (2) to obtain RC delay and average current estimates. It is easy to show that the scaled network will always meet the RC delay constraints. However, the actual flows in the scaled network could be different from the flows at the end of Step 3. This can happen because the edge capacitances in the scaled network are less than (or equal to) the edge capacitances in the original network, and the flow values in Steps 2 and 3 are calculated based on the original edge capacitances. Consequently, a reverification step is necessary to ensure that the currents in the scaled network are within limits. In practice however, we find the average currents in the scaled network

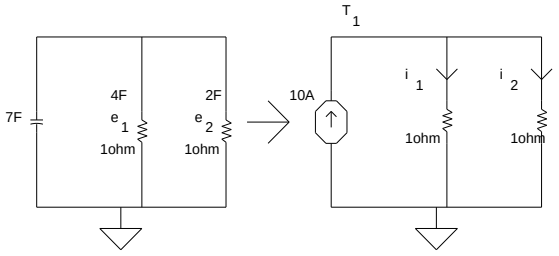

Figure 2: Example of grid to be sized

are usually less than those predicted by the flow values at the end of Step 4.

Loosely speaking, the two linear programs described above are a means of choosing flow and potential values so that the widths of high capacitance edges in the grid are trimmed. After solving the *flow redistribution* problem in step 2, we obtain a *minimum cost* feasible flow, so that the flow from a receiver is diverted into lower capacitance paths to ground. After solving the *potential adjustment* problem in step 3, we would effectively increase the *potential difference* across the higher capacitance edges in the network, further reducing their widths in the scaled network.

A simple example will help to illustrate the algorithm. Consider the network shown in Figure 2. The load capacitance at node 1 is $7F$, edge e_1 has capacitance $C(e_1) = 4F$, and e_2 has capacitance $C(e_2) = 2F$. The edges are represented by their π models and the resulting RI network is also shown. We are required to maintain the RC delay to node 1 at less than 12 seconds.

The flow variables are i_1 and i_2, and the potential variable is τ_1. The constraints on the variables are

$$i_1 + i_2 = C_1 = 10. \tag{13}$$

$$0 \leq \tau_1 \leq 12 \tag{14}$$

$$\tau_1 \geq R_k i_k \quad k = 1, 2 \tag{15}$$

The solution to this flow redistribution problem is $i_1 = 0$, $i_2 = 10$ and $\tau_1 = 10$. This solution implies that e_1 may be removed from the network, and e_2 will be left untouched. We apply the potential adjustment next, and obtain $\tau_1 = 12$, to obtain a scaling of e_2 by 5/6. Thus, after flow redistribution and potential adjustment, the total edge capacitance is reduced by 72.2% (while meeting the RC delay constraint).

If we solve the sizing problem in Figure 2 exactly (this can be done easily in such a simple example), we obtain a scaling of e_1 by 0 (that is, e_1 can be deleted) and of e_2 by 7/11. Thus, the optimal solution results in an edge capacitance reduction of 78.8%. As expected, there is an optimality gap between the approximate solution in the earlier paragraph and the exact solution. The causes for this gap are twofold: In the approximate algorithm, we are not accounting for capacitance reduction due to scaling during the flow and potential adjustments. Secondly, we are not using the exact nonlinear cost function, preferring to solve simpler problems instead.

We note however, that multiple sweeps of the approximate algorithm can narrow this gap by taking advantage of the reduction in capacitance due to one sweep in the next. We consider the scaled network after one sweep, recalculate the capacitances of the remaining edges, and repeat the flow and potential adjustment. Assume for now that the capacitance is linearly dependent on the edge widths. Then, we find that a second sweep scales e_2 by an additional factor of 0.78. That

is, two *sweeps* of flow and potential adjustments yield a total reduction in capacitance of 78.2%.

3 Implementation

For small clock grids, we could use a standard linear program solver to implement the sizing algorithm. However, when the grid has more than a million edges in it, we need to develop alternative implementations of the sizing algorithm. The flow redistribution problem is superficially similar to the well known *min-cost flow* problem[2]. Thus, we expect that a network flow algorithm to solve this problem could be devised. The potential adjustment problem is a generalization of the shortest path problem, and we will demonstrate a very efficient network algorithm for its solution.

3.1 A network algorithm for the flow redistribution problem

Before proceeding, we introduce some terminology. Suppose we have a feasible choice of edge flow and node potential values in the network. An edge (i,j) is *forward feasible* if $x_{ij} < M_{ij}$. An edge (i,j) is *backward feasible* if $x_{ij} > 0$. An edge (i,j) is *critical* if $p_i - p_j = R_{ij}x_{ij}$. A path is critical if every edge on it is critical. A node i is *saturated* if $p_i = \tau_{max}$. A critical path for a node i is a critical path starting at node i and ending at the ground node. A critical path for a saturated node is termed a *saturated critical path.*

A *legal* cycle in the network is a closed path of edges, where each forward edge (traversed from head to tail) is forward feasible, and each backward edge (traversed from tail to head) is backward feasible. The sets of forward and backward edges in a cycle L are denoted by $F(L)$ and $B(L)$ respectively. The incremental cost of a cycle L is defined to be

$$\sum_{(i,j)\in F(L)} C_{ij} \quad - \quad \sum_{(i,j)\in B(L)} C_{ij}$$

The *intersection* of a cycle L with any path P yields a set of edges. The *effective resistance* of this intersection is

$$\sum_{(i,j)\in F(L)\cap P} R_{ij} \quad - \quad \sum_{(i,j)\in B(L)\cap P} R_{ij}$$

The *potential drop* across any edge (i,j) is defined to be $f_{ij} = R_{ij}x_{ij}$. The *potential difference* across (i,j) is $g_{ij} = p_i - p_j$. The *edge slack* in (i,j) is defined as $h_{ij} = g_{ij} - f_{ij}$. The potential drop along a path is the sum of the potential drops in the edges of the path. If a node has *in-degree* 0 in the network, it is referred to as a *source node.* The *potential drop* to a node i (denoted by d_i) is defined as follows. If i is a source node, then $d_i = 0$. Otherwise,

$$d_i = \max_{(k,i)\in E} d_k + R_{ki}x_{ki} \tag{16}$$

Thus, the drop to a node i is determined by the maximum potential drop path from a *source* node to i.

The potential at a node i will be calculated as

$$p_i = \max_{(i,j)\in E} p_j + R_{ij}x_{ij} \tag{17}$$

with $p_0 = 0$ being the potential of the ground node. In other words, the potential at node i is defined by the maximum potential drop path from i to ground. The *potential slack* at a node i is defined as

$$s_i = \tau_{max} - (p_i + d_i) \tag{18}$$

The flow redistribution problem is similar to the min-cost flow problem, with the addition of the node potential constraints (7) and (9). Let us first characterize an optimal solution to this problem. We have the following result:

Theorem 3.1 *A feasible assignment of flow and potential values in the network is optimal if and only if every negative cost legal cycle has a positive resistance intersection with some saturated critical path.*

Proof: In the interest of brevity, we will omit the proof of this Theorem, which is based on the duality theory of linear programming.

The result in Theorem 3.1 provides the basis for an algorithm to solve the flow redistribution problem. We start with the feasible flow obtained by the initial solution of the network, and then modify this flow in order to reduce the cost. To obtain such a modification, we look for a legal negative cost cycle whose intersection with every saturated critical path has a non-positive resistance.

Finding a negative cost legal cycle involves solving a shortest path problem with both positive and negative edge weights. However, the algorithms needed to solve such a shortest path problem have worst case complexity $O(|V|^3)$[6], which is unacceptable for large networks. Instead, we use a different approach, which borrows certain from the *out-of-kilter* algorithm [2], and involves solving a shortest path problem with non-negative weights.

We build a *shortest path tree* K which, for each node i in the network, contains the minimum capacitance path from i to the ground node[2]. Given a shortest path tree K, an edge (i,j) is *forward amenable* if $(i,j) \in K$, $s_i > 0$, and $x_{ij} < M_{ij}$. The edge (i,j) is *backward amenable* if $x_{ij} > 0$. An amenable cycle is one in which all forward edges are forward amenable and all backward edges are backward amenable. If an amenable cycle exists, it is guaranteed to have negative incremental cost and a non-positive resistance intersection with every saturated critical path. An amenable cycle can be found using a *breadth first search* [6], which has worst case complexity $O(|E|)$. Once such a cycle has been found, we will check if a certain amount of flow can be sent around it, and change the flow in the network accordingly. This process of cycle flow adjustments is continued until no further cycle can be found.

Algorithm 1:

1. Solve the initial network as in (2) to obtain τ_i, $i > 0$, and orient the edges in the network using the initial solution. For each directed edge $(i,j) \in E$, obtain the initial flow x_{ij} as

$$x_{ij} = \frac{\tau_i - \tau_j}{R_{ij}} \tag{19}$$

   and set the node potential values as $p_i = \tau_i$ for $i > 0$.
2. Find the shortest path tree K.
3. Calculate node drops d_i using (16).

[2] In the limit, if $\tau_{max} = \infty$, the optimal flow distribution will only use edges from K.

4. Pick an unmarked backward amenable edge $e = (i, j)$ which is not in K. If no such unmarked edge exists, stop. Otherwise mark e and proceed to the next step.
5. If edge e has positive flow, find an amenable cycle L that traverses e backward. This cycle may be found using an appropriate breadth first search starting at i. If no such cycle L exists, go to the previous step.
6. Calculate the flow that can be sent along L as follows: First calculate the total forward resistance in L as

$$R_{f(L)} = \sum_{(i,j)\in F(L)} R_{ij}$$

Next, calculate the minimum slack on L as

$$s_L = \min_{(i,j)\in F(L)} s_i + h_{ij}$$

The allowable flow adjustment around L is then calculated as

$$\delta f_L = min\{s_L/R_{f(L)}, \min_{(i,j)\in F(L)} M_{ij} - x_{ij}, \min_{(i,j)\in B(L)} x_{ij}\}$$

It can be seen that sending δf_L units of flow around L will maintain feasibility of the flow. If δf_L is greater than a small parameter $\epsilon > 0$, then send δf_L units of flow around L. Otherwise go to step 4.
7. Calculate node potentials (using (17)) and drops (using (16)) with the revised flow values, and go to the previous step. Each execution of steps 6 and 7 will be termed a *cycle adjustment*.

In the actual implementation of the algorithm, we sort the edges in decreasing order of capacitance, so that in step 4 of the algorithm, edges with higher capacitances are considered earlier for flow reduction. Because we have chosen an acyclic orientation in Step 1, every step in the algorithm above (other than Step 1) has complexity $O(|E|)$. The cost of the flow is reduced by each cycle adjustment in Step 6, and each cycle adjustment can be performed with worst case complexity $O(|E|)$.

It is not possible to predict the number of cycle adjustments that will be needed before the algorithm terminates. In practice, we set a limit on the number of edges that will be considered in Step 4. The number of flow adjustments for each edge picked in Step 4 is limited by choosing ϵ to be sufficiently large in Step 6. In general, this algorithm will produce a sub-optimal solution to the flow redistribution problem. However, we find that the reduction in interconnect capacitance is appreciable for very large networks (See Section 4).

3.2 An algorithm to solve the potential adjustment problem

Let us state the potential adjustment problem formally. We are given a set of flows $\{x_{ij}\}$ in the network (we assume without loss of generality that $x_{ij} > 0$ for each edge (i, j)). We compute the node potentials p_i for this set of flows using (17). For each node i, define

$$\theta_i = \sum_{(i,j)\in E} \frac{C_{ij}}{R_{ij}x_{ij}} - \sum_{(k,i)\in E} \frac{C_{ki}}{R_{ki}x_{ki}} \qquad (20)$$

Also, let $m_i = \tau_{max} - p_i$ for each node $i > 0$, and for each edge (i, j), let $s_{ij} = p_i - p_j - R_{ij}x_{ij}$.

The potential adjustment problem requires us to choose, for each node i, an adjustment $\pi_i \geq 0$ such that

$$\sum_{i\in V} \pi_i\theta_i \qquad (21)$$

is maximized, subject to the restriction that

$$p_i \leftarrow p_i + \pi_i \qquad (22)$$

is a feasible potential distribution for the network (that is, (7) and (9) are satisfied).

In other words, we are asked to maximize (21) subject to the constraints

$$\pi_i \leq m_i \text{ for each node } i > 0 \qquad (23)$$

and

$$\pi_j - \pi_i \leq s_{ij} \text{ for each edge } (i, j) \qquad (24)$$

We will assume that $\pi_0 = 0$. This problem bears a certain resemblance to a shortest path problem [3].

As before, we will first characterize an optimal solution to the potential adjustment problem. This characterization will yield a natural algorithm to solve the problem. Before stating the result, we introduce some notation. Let A denote the set of critical edges in the network, and define a new network $N = (V, A)$. For each $i \in V$, let U_i denote the set of all nodes j such that there is a path in N from j to i (include i in U_i). Define

$$\alpha_i = \sum_{j\in U_i} \theta_j \qquad (25)$$

and

$$\beta_i = \min_{j\in U_i} m_j - \pi_j \qquad (26)$$

Then, we have the following result. Once again, we omit the proof in the interest of brevity. This result is also proved using the duality theory of linear programs.

Theorem 3.2 *A choice of π values satisfying (23) and (24) is optimal if and only if*

$$\max_{i\in V} \alpha_i\beta_i \leq 0 \qquad (27)$$

We use Theorem 3.2 to devise an algorithm to find the optimal π values.

Algorithm 2:

1. For each $i \in V$, set $\pi_i = 0$.
2. For each $i \in V$, calculate α_i and β_i. These can be computed using a traversal of the network in $O(|V|)$ time.
3. Find an $i \in V$ such that $\alpha_i\beta_i > 0$. If no such i exists, stop because the optimal π values have been found. Otherwise proceed to the next step.
4. For each $j \in U_i$, set $\pi_j \leftarrow \pi_j + \beta_i$. Go to step 2.

Referring to Theorem 3.2, it is evident that this algorithm will find the optimal solution to the potential adjustment problem. The running time of the algorithm can be bounded as follows. Let $W \subset V$ be the set of source nodes in the network N. It may be seen that after each execution of step 4, at least one $j \in W$ has $\pi_j = m_j$. Thus, step 4 is repeated at most $|W|$ times. Each execution of steps 2,3,4 takes $O(|V|)$ time. Thus, the running time of the algorithm is $O(|V||W|)$.

Sweep No.	Cap. Reduction	CPU time
1	10.7%	18hrs
2	14.4%	19hrs
3	16.0%	19hrs

Figure 3: Summary of results for DC21064A grid

Sweep No.	Cap. Reduction	CPU time
1	3.7%	26hrs
2	6.1%	28hrs
3	8.3%	28hrs
4	9.6%	27hrs

Figure 4: Summary of results for DC21164 grid

4 Results

The algorithms described in this paper were implemented in a CAD tool which was applied to several examples, of which two were very large clock networks. The tool proved its effectiveness by achieving significant reductions in interconnect capacitances in all examples. We summarize the performance of the tool on the two largest problems. All CPU times were measured on a DEC 7000 Model 750.

4.1 Clock Network of DC21064A

The DC21064A microprocessor contains nearly 3 million devices and is designed to operate at clock frequencies of 275MHz and higher. The clock grid contains 1015500 edges, 15660 drivers and 81300 receivers. The interconnect capacitance of the grid as initially designed was 1.4nF. The maximum driver-receiver RC delay was 0.16nsec.

We applied three sweeps of the sizing algorithm to the problem, keeping the RC delay constraint at 0.16nsec. The number of edges considered in step 4 of the flow redistribution algorithm were limited to 10000. The results are summarized in Figure 3.

We note that a significant interconnect capacitance reduction of 16% was obtained by three sweeps of the sizing algorithm, at the expense of about 3 days of CPU time.

4.2 Clock Network of DC21164

The DC21164 microprocessor contains over 9 million devices and is designed to operate at clock frequencies of 300MHz and higher [4]. The clock grid has 1279896 edges, 17670 drivers and 340665 receivers. The total interconnect capacitance of the initial grid was 2nF. The maximum RC delay from driver to receiver in the network was 0.1nsec. Thus, this network had significantly more receivers than the DC21064A case, and the RC delay constraints were also more stringent.

After four sweeps of the sizing algorithm (with a limit of 10000 set on the number of edges considered for cycle adjustments), a reduction of 9.6 percent was observed in the interconnect capacitance of the clock grid. The results are summarized in Figure 4. The capacitance reduction on this grid, though significant, was less than that observed in the DC21064A case, which could be explained by the tighter RC delay constraint on the problem.

5 Conclusions

In this paper, we have presented a novel approach to size high performance clock distribution networks with arbitrary topologies. We presented a sizing algorithm which involves the solution of a sequence of two linear programs. We have presented efficient network algorithms to solve these linear programs. These algorithms have been effectively applied to size very large clock networks in complex microprocessors, and have provided substantial reductions in the interconnect capacitance of these networks.

Acknowledgement: The first author would like to thank Bill Bowhill, Shashank Goel, Sachin Sapatnekar and H. Narayanan for their helpful suggestions.

References

[1] L.O. Chua and P.M. Lin, *Computer-aided Analysis of Electronic Circuits,* Englewood Cliffs, NJ: Prentice-Hall Inc, 1975.

[2] E.L. Lawler, *Combinatorial Optimization: Networks and Matroids*, New York: Holt, Rinehart and Winston, 1976.

[3] C.H. Papadimitrou and K. Stieglitz, *Combinatorial Optimization: Algorithms and Complexity*, Englewood Cliffs, NJ: Prentice-Hall Inc, 1982.

[4] J. Edmondson et. al., "Internal Organization of the Alpha 21164, a 300-MHz, 64-bit, Quad-Issue, CMOS RISC Microprocessor," *Digital Technical Journal,* vol. 7, no. 1, 1995.

[5] T-M Lin and C.A. Mead, "Signal Delay in General RC Networks," *IEEE Transactions on Computer-Aided Design,* vol. CAD-3, pp 331-349, October 1984.

[6] S. Even, *Graph Algorithms*, Rockville, MD: Computer Science Press, 1979.

[7] J. Cong and K-S. Leung, "Optimal wiresizing under the distributed Elmore delay model," *Proceedings of the International Conference on Computer-Aided Design*, pp. 634-639, 1993.

[8] S.S. Sapatnekar, "RC Interconnect Optimization under the Elmore Delay Model," Tech. Report ISU-CPRE-93-SS12, Department of Electrical Engineering and Computer Science, Iowa State University, Ames IA, 1993.