A Hierarchical Decomposition Methodology for Multistage Clock Circuits

Gary Ellis

IBM Microelectronics,
IBM, Austin, TX 78758

Lawrence T. Pileggi and Rob A. Rutenbar

Department of Electrical and Computer Engineering,
Carnegie Mellon University, Pittsburgh, PA 15213

Abstract†

This paper describes a novel methodology to automate the design of the interconnect distribution for multistage clock circuits. We introduce two key ideas. First, a hierarchical decomposition of the layout divides the problem into a set of local Steiner-wired latch clusters (to minimize and balance local capacitance) fed globally by a balanced binary tree (to maximize performance). Second, we recast the global clock distribution problem as a simultaneous optimization of clock topology, clock segment routing, wiresizing, and buffering. The hierarchical decomposition reduces the problem complexity and allows use of more aggressive optimization techniques. Integration of the geometric and electrical optimizations likewise allows more aggressive performance goals. Experiments with an industrial design comprising over 16,000 latches demonstrate the efficiency of the approach: a complete clock distribution solution met a 200MHz cycle time specification with only 310ps of skew, met strict current density constraints, exhibited good delay matching across uniform wire width and device variations, and was completed in under 10 CPU hours.

1 Introduction

It has been shown that clock skew can reduce system performance by as much as 10% in high speed VLSI designs [1]. Further, the delay dominance of the interconnects for modern technologies has put more control of this skew problem into the hands of the physical designer. As such, the convergence upon a good clock circuit layout requires tight interaction between the clock circuit synthesis, placement, and timing analysis steps. Currently, the automation process of multistage clock circuit design is divided into two steps; circuit topology selection with routing [2][4][6,9], followed by a buffer insertion and wiresizing step [5][10][11][12]. The first step produces a clock circuit topology with a wire or delay-balanced routing. The second step inserts repowering buffers into this large net and sizes the devices and individual wire segments to satisfy performance constraints. Although the buffer insertion steps consider delay sensitivities for manufacturability concerns, the routing algorithms which provide their input circuits do not consider this. As a result, arbitrary topology and routing decisions are made which are blind to the needs of the buffering and sizing steps and may seriously impact the quality of the results. Further, other manufacturability concerns, such as electromigration, have been left unaddressed. Finally, all of the algorithms for these two-step approaches deal with the circuit in a flat manner.

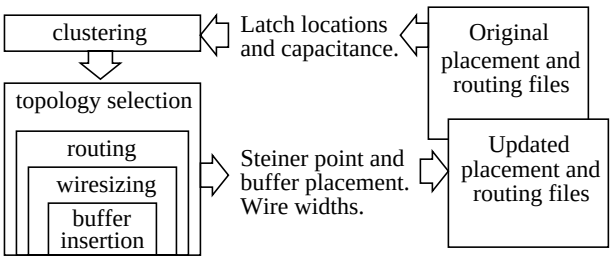
In this paper, a more complete framework has been developed which combines the steps of topology selection, routing, wiresizing, and buffer insertion. By combining all of these steps together, topologies can be selected which respect the requirements of the repowering buffers during the optimization. This permits the investigation of physical design solutions which balance the driven loads using the interconnect network alone, removing the need for additional *dummy* loading to equalize delays [15][16]. Further, a new wiresizing scheme is introduced to address the concerns of electromigration. Finally, the complexity problems associated with a flat treatment of the circuit have been avoided by the use of a hierarchical decomposition approach; we extend here the ideas of [20] from the simpler case of single-stage unbuffered clocks to the more general multistage problem. This paper presents an overview of the algorithms and implementation details that support this methodology; additional details may be found in [19].

This paper is organized as follows. The hierarchical decomposition is introduced in Section 2. The overall optimization strategies for the global and local portions of this decomposition are outlined in Sections 3 and 4. The details of the buffer modeling and wiresizing used in this strategy are given in Sections 5 and 6. The experimental results of Section 7 clearly show the merits of this approach, while Section 8 gives our concluding remarks and some suggestions for future work.

2 Hierarchical Decomposition

Since the best single stage routing methods require extensive topology search algorithms [8][9], designs larger than a few thousand latches remain intractable. Fortunately, the performance of the network is dominated by the large delays near the root of the tree, and this allows for a tremendous savings in effort by using a convenient partitioning of the circuit. It is apparent that the size of the problem could be reduced considerably by largely ignoring the details of the leaf wiring and concentrating on the longer upper wires of the tree during optimization. This observation has motivated a hierarchical decomposition of the problem: a collection of Steiner-wired latch clusters connected by a balanced wire tree. This decomposition of the circuit and how it fits in the design flow is illustrated in Figure 1.

Figure 1. Flow of clock optimization methodology.



First, the latch locations and latch pin capacitances are taken from the placement and routing files of the pre-placed ASIC. The latches are then clustered into small groups for the local clock signal distribution. These clusters are passed to the global distribution optimizer which uses a tightly integrated set of algorithms to construct the global clock tree. These algorithms include topology selection, routing (Steiner point placement), wiresizing, and buffer insertion. The placement of the clock objects and the wire width specifications are then passed along as constraints which augment the placement and routing files of the design.

3 Global Distribution Optimization

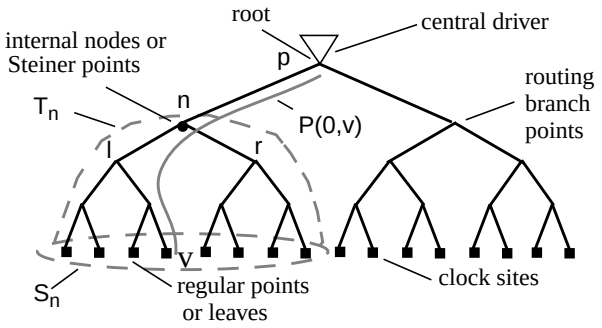
For the optimization of the global distributions, we have adopted a simulated annealing method similar to [9]. All simulated annealing

†. This work was supported by the Semiconductor Research Corporation under contract 96-DC-068.

0-89791-993-9/97 $10.00 © 1997 IEEE

problem formulations may be described in four basic parts: 1) the state representation, 2) the move set, 3) the cost function, and 4) the cooling schedule. The annealer vigorously explores complete alternative tree topologies, and invokes deterministic Steiner point placement techniques to render a complete routing for each visited topology. Each complete tree is then sized and buffered, as we shall describe later in this section, for each annealing perturbation. The central idea is to search aggressively for an optimal clock layout, and to evaluate each layout after a fast, heuristic sizing and buffering calculation. For the global distribution, the state is represented by a *binary tree* as shown in Figure 2. The actual Steiner point locations are found by the *Deferred Merge Embedding* (DME) technique of [7]. Also, *snake wire* is not included in the routing as its effect is small and it is unlikely to be used in practice.

Figure 2. Binary tree representation of the state.



Each node, *n*, has both a left, *l*, and a right, *r*, child as well as a parent node, *p*. The other notation adopted here is defined concretely below:

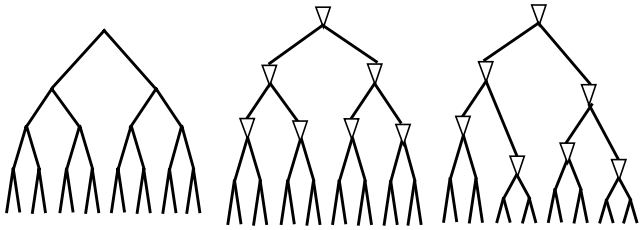
$$
\begin{aligned}
P(u,v) &= \text{set of segments on the path from node } u \text{ to } v \\
n &= \text{a segment on path } P(u,v) \\
l_n &= \text{Manhattan length of segment n} \\
&= |x_n - x_p| + |y_n - y_p| \\
T_n &= \text{subtree driven by segment n} \\
S_n &= \text{set of clock sites of subtree n} \\
Cd_n &= \sum_{j \in T_n} c_j + \sum_{j \in S_n} C_j = \text{downstream capacitance} \quad (1) \\
td_n &= \frac{t_l + td_l + t_r + td_r}{2} \\
&= 0 \text{ for clock sites} \\
t_n &= r_n\left(\frac{c_n}{2} + Cd_n\right) = \text{delay of wire segment n}
\end{aligned}
$$

The delays are computed in a bottom-up manner and td_n at any node is the *average* node-to-site path delay from *n* to any of its sites, $s \in S_n$. Since we are interested in zero-skew trees, these path delays are often (but not always) all the same.

To this point, current methodologies have held buffer insertion and wiresizing as strictly postprocessing steps. As it was pointed out in [12], the power-delay product of a clock distribution network can be reduced significantly with the insertion of repowering buffers. However, there did not seem to be a great difference between various buffer insertion-point and sizing configurations (although this may change for deep submicron technologies). In short, finding the best of these buffer insertion configurations for a given design may be too expensive with too little performance improvement to justify merging this optimization step with the others. On the other hand, it would be a mistake to simply optimize a single stage without any knowledge of the buffering to be added later. In this work, we choose a compromise: optimize a multistage tree with a fixed buffer and size configuration, *i.e.* fixed levels of the tree at which the Steiner branch points will be driven by buffers. We refer to this as a *constant* buffer insertion heuristic. The intention is to find a layout that satisfies this configuration well with only minor postprocessing modifications. It should be noted that the problem of selecting the buffer sizes and insertion points is not addressed in this paper. One possible approach is to use the algorithm presented in [12] on a MMM [2] or zero-skew [4, 9] clock tree to estimate the buffer sizes and locations, and then use this configuration with the optimization described here. It is assumed that such decisions will be made by the designer in a clock planning step. Finally, it should be noted that the constant buffer insertion heuristic used here is not an integral part of the optimization strategy and is adopted in the interest of computational time only.

This constant buffer insertion configuration method is illustrated in Figure 3. First, all tree topology manipulations are made without any knowledge of the buffering. This allows a full exploration of the geometric design space without being inhibited by the levelization that buffering inherently introduces. This is actually accomplished by *erasing* all of the buffers in the tree before changing the topology and then *inserting* buffers into the new tree according to the given constant buffer configuration. The tree is then routed from the bottom-up, now accounting for the delay effects of the buffers. Wire sizes are then selected for the segments to reduce current densities and interconnect delays without impacting skew.

Figure 3. Buffer insertion steps.



a) Manipulations performed on single stage tree. b) Buffer insertion step. c) Multistage tree is routed and wiresized.

For the move set, the fundamental state manipulator is the reconnection of a single subtree within the state. This is accomplished by removing the parent node from the tree graph and reconnecting it to form a different topology. The global optimization cost function is a weighted sum of figures of merit which include the *average delay*, a *skew* term, and the *total wire length* of the global route. These terms are defined concretely below:

$$
\begin{aligned}
\text{delay} &= RoCd_0 + \frac{1}{|S_0|}\sum_{p \in P}\sum_{i \in p} \rho l_i\left(\frac{\varepsilon l_i}{2} + Cd_i\right) \\
\text{skew sum} &= \sum_{i \in T_0} \left|td_l + t_l - td_r - t_r\right| \quad (2) \\
\text{wirelength} &= \sum_{i \in T_0} l_i
\end{aligned}
$$

Here, *P* is the set of all root-to-site paths in the network and the root is commonly denoted as node 0. In the skew sum term, td_l and td_r represent the subtree delays of the left and right children respectively. The delays t_l and t_r represent the interconnect delays of the wires connecting the left and right subtrees to node *i*. Ideally, these

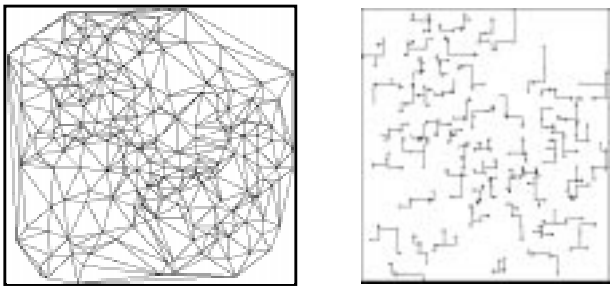
delays should balance, and their difference should be zero for all nodes in the design. In cases where the layout is poorly balanced, this sum can effectively capture the skew introduced by any and all nodes in the circuit.

A modified version of the Lam-Delosme cooling schedule [17] is used to control the progression of the state to a good globally optimized solution, and the number of moves performed at each temperature is proportional to the number of objects in the state. Although much care has been taken to *tune* the schedule to this problem, the parameters found for one design do not vary much for the other benchmarks. The specific values for the temperatures or weights are not presented here, but it should be noted that the most emphasis was placed on the wirelength term, and the least emphasis was on the average delay.

4 Local Distribution Optimization

The latch clustering ideas from [20] are used for the local distribution of the clock, which are reviewed here briefly for clarity. To find the topology and routing of the local distribution, the Delaunay triangulation is used. The interest in this construction is simply with the *near neighbor* information that it conveys about objects in a 2D space. The intention is to use a Delaunay triangulation to produce a graph of nearby clock sites to identify which clock sites are close to one another and can possibly be wired together. The connecting edges will also supply us with a means of estimating the cluster route lengths. An example of this is shown in Figure 4 below where a Delaunay triangulation has been constructed from the clock sites of the Tsay r1 benchmark (a) and the resulting latch clustering is shown in (b).

Figure 4. The Delaunay triangulation and latch clustering.



a) Delaunay triangulation b) latch clustering

As with the global route optimization, simulated annealing is used to find an optimal local clustering. The state of the local distribution is given in Figure 5. We will consider the vertices, *v*, as clock sites (latches) and the edges, *e*, as possible wire connections. If two sites are to be connected together, the edge connecting their corresponding vertices is deemed to be *on*. A cluster can therefore be represented by a subgraph, *Gc(Vc,Ec)*, whose sites correspond to vertices, $v \in V_c$, and connecting wires correspond to *on* edges, $e \in E_c$. For the sake of minimizing wire, the restriction that cluster subgraphs will not contain any cycles is imposed. The Steiner wire estimates, shown in Figure 5b, are found by using a greedy version of the L-flipping algorithm given in [14]. We have found through experimentation that this algorithm finds routes which are typically 30% shorter than a zero-skew route connecting the same points.

All of the fundamental moves employed to manipulate the state exchange a single latch between any two clusters. Whenever a latch is exchanged, the Steiner wire estimate algorithm is re-run on the source and target clusters to assess any change in total wire use and capacitive loading. In addition, cluster splitting moves have been implemented so that the total number of clusters need not remain constant. The local optimization cost function is a weighted sum of the following terms:

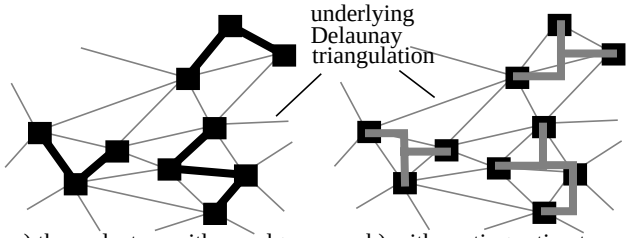
$$\begin{aligned} \text{load penalty} &= \sum_{k \in K} \left| C_k - C_{fan} \right| \\ \text{skew penalty} &= \sum_{k \in K} (S_k - S_{limit}) u(S_k - S_{limit}) \\ \text{total wirelength} &= \sum_{k \in K} \sum_{e \in E_k} |e| + \gamma D(\sqrt{|K|} - 1) \end{aligned} \tag{3}$$

where, *u(x)* is the usual Heaviside step function (*u(x) = 1* if $x > 0$, and *= 0* otherwise). *K* is the set of all clusters and E_k is the set of *on* edges in cluster *k* (|*e*| is the wire length of the connecting edge, as computed by the greedy L-flipping algorithm). The first term biases the clustering toward a solution where the average capacitive loading is near some specified fanout target, C_{fan}. The second term limits the size of each cluster so that the intra-cluster skew will remain low for each local route. It is important to note that the global optimization step will select the tap point to each cluster when the global tree is constructed and, consequently, the skew can only be estimated during the clustering step. The tap points are constrained to the latch locations themselves so the term, S_k, is a *worst-case* intra-cluster skew estimate:

$$S_k = \rho \sum_{e \in E_k} |e| \left(\frac{\varepsilon \sum_{e \in E_k} |e|}{2} + \sum_{v \in V_k} C_v \right) \tag{4}$$

Equation (4) is the delay found by driving all of the latches in a cluster through a single segment whose length is equal to the wire used by this local route. It is usually very pessimistic in practice, but provides us with a strict upper-bound. The final term in (3) is simply the total wirelength of all of the local routes plus an *estimate* of the global route length. The global estimator part is an empirically derived equation. Here, *D* is the chip diameter and γ is a constant selected to be about 1.2.

Figure 5. The state representation of the latch clustering.



a) three clusters with *on* edges b) with routing estimates

5 Buffer Modeling

In this section, the modeling for the repowering buffers is described. The design of the repowering buffer is a pair of series connected inverters. The first inverter acts as an input stage and cleans up the risetime of the input signal for the second inverter. Without the first inverter, there would be a potential for signal degradation if the previous stage is heavily loaded. The second stage has been sized (30X minimum transistor width) to drive a typical interconnect network on a 1cm chip in a 0.35um process.

In this work, we use a linear circuit approximation, similar to that of [18], to model this buffer design: a time-dependent time-delayed ramp voltage source with a fixed output impedance. The time delay and risetime of the ramp voltage source are controlled only by the input signal to the buffer input. The driven interconnects are modeled by the admittance approximation technique of O'Brien and Savarino [13]. In our application, a reduced order circuit (pi-model)

is computed which approximates the response of the complete interconnect network with the driven loads. To compute the buffer delays, an equation for the transient response of the linearized buffer model when driving an arbitrary pi-model has been derived. This parameterized equation is solved by using the Newton-Raphson technique for each buffer in the design.

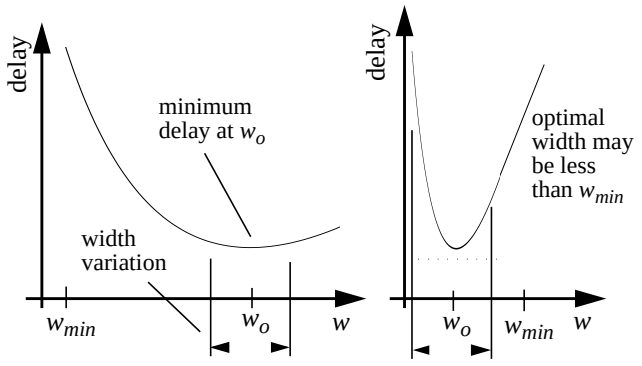
The accuracy of such linear buffer models and interconnect driving-point model has been demonstrated in [18] and [13] respectively, and we have observed similar results in this work. In Section 7, we will investigate how the accumulation of these errors over several levels of buffering impacts the final design. In the next section, the wiresizing technique used with this buffer modeling is discussed.

6 Wiresizing a Multistage Design

Recall that our goal is to wiresize the routed clock layout we create after each perturbation of the clock topology proposed by the annealer. Hence, the constraints are not only to handle sizing for performance and reliability, but also to complete this operation very quickly. It has been demonstrated in [12] that *reducing* the delay sensitivity of a multistage design is often more desirable than *minimizing* it in the interest of wire area (and power). Further, we will show here that such a minimization of sensitivity (and delay) may not be possible for some multistage designs. In this section, a new fast wiresizing technique useful with multistage clock circuits is discussed.

For the single stage designs of [5][10][11], the delay can be minimized as the driver output impedance, *Ro*, is usually small for large drivers (~10 Ohms) and the loading, *Cd*, is very large for root wires. This results in very wide root wires (10X to 20X w_{min}) which have little sensitivity to width variations. For multistage designs, however, the repowering buffers are much smaller and, therefore, have larger output impedances (100 to 500 Ohms). Coupled with driving smaller loads, such delay minimization algorithms can produce very narrow interconnect wires, with the optimal wire width near or below w_{min} in many cases. Although these widths are optimal and the design has low delay sensitivity at the nominal point, the design is still susceptible to wire width variations. This is illustrated in Figure 6.

Figure 6. Effect of width variations on an optimal sizing.



a) single stage design b) multistage design

In these plots, the possible wire widths are plotted collectively against the total delay of the design (device + interconnect). In this abstraction, the x-axis represents the wire widths for all of the interconnects in the design and w_o represents their optimal values. Therefore, as all of the interconnects deviate from these w_o values, the total delay increases.

In practice, a monotonically decreasing sizing is preferred. The high current carrying wires near the root are widened to reduce current densities while, at the leaves, the currents are low so they are set to w_{min} to reduce loading and save power. In this work, a special *balanced wiresizing scheme* is adopted which follows this idea. In this scheme, a fixed maximum wire width is selected for the segments that are directly driven by the buffer. Then, where each wire splits (Steiner point) as the route progresses to the leaves, the wire segment width is reduced by half. This sizing progression continues to the sink pins of the subtree or until the minimum wire width is reached.

The motivation for this sizing is an attempt to equalize the current densities across the design. As a first order approximation, the peak current, i_n, through any wire segment, *n*, is a function of the capacitive load driven by the segment, Cd_n, and the risetime of the driving signal (Figure 7). In this diagram, r_n and c_n are the wire resistance and capacitance of the segment computed from the width, w_n, the length, l_n, and the height, *h* (assumed to be a constant for all interconnects), of *n* and the bulk resistivity, ρ, and capacitance per unit area, ε, of the interconnect layer. dV_n/dt is the fastest rate of change of the voltage at node *n*. It may be approximated by *Vdd/2tr* (where *tr* is the 10% to 50% risetime). The maximum current density is this peak current divided by the cross-sectional area of the segment, $w_n h$.

The loads, *Cd*, in a clock net are wire capacitance dominated and are typically very large in comparison to the individual segment capacitances, *c*. As any path is traversed in such a binary tree from a driver to any pin, the driven load is cut roughly in half at each branch (or Steiner) point, splitting the driving current equally. With each child segment carrying half of the parent's current, only half the wire width is needed to achieve the same current density as the parent segment. In short, this sizing scheme tries to achieve the same current density for all of the segments along any driver-to-pin path. Although this sizing scheme is intended for the equalization of current densities, it is still possible to seek minimum delay sensitivity. To see this, consider the average root-to-latch delay:

$$\overline{td} = \frac{1}{N}\sum_{s \in S_0}\left(\sum_{i \in B(root,s)} tbuff_i + \sum_{i \in P(root,s)} tint_i\right) \quad (5)$$
$$= \frac{1}{N}\sum_{s \in S_0}\left(\sum_{i \in B(root,s)} Ro_i Cd_i + \sum_{i \in P(root,s)} 2^i \rho \frac{l_i}{w}\left(\frac{\varepsilon l_i w}{2^{i+1}} + Cd_i\right)\right)$$

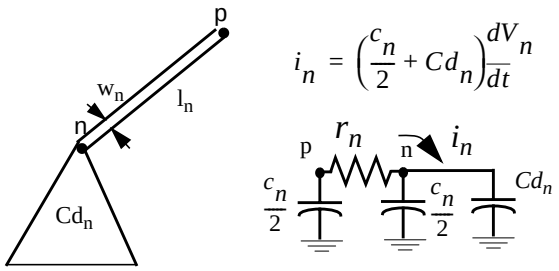
which includes both buffer and interconnect delays ($tbuff_i$ and $tint_i$). The notation used here is the same as above with the addition of *B(u,v)* which is the set of buffers on the path from node *u* to *v*. Here, the buffer delays are modeled by constant output impedances, Ro_i, and the interconnect delays are in the usual Elmore [3] form. This is a lower order approximation when compared to the modeling of the previous section, but it does capture the dominant behavior of the circuit. All paths from the *root* node to each of the *N* latches in the design, $s \in S_0$, are considered in this sum. The wire width, *w*, corresponds to the maximum wire width of the route at the driving point of each buffer and the wire segments decrease in width as the paths trace out to their targets. From the derivative of this delay with respect to *w*, the optimal width is:

$$w_o = \sqrt{\frac{\sum_{s \in S_0}\sum_{i \in P(root,s)} 2^i \rho l_i \sum_{j \in S_i} C_j}{\sum_{s \in S_0}\sum_{i \in B(root,s)} Ro_i \varepsilon \sum_{j \in T_i} \frac{l_j}{2^j}}} \quad (6)$$

When such an optimal initial width is realizable the design will

have minimum sensitivity, otherwise a larger width must be selected and the sizing scheme can only produce a roughly equal distribution current densities along any path. In the next section, the experimental results will demonstrate the utility of this sizing scheme and the buffer modeling in this optimization framework.

Figure 7. Estimating the current through an interconnect.



a) abstraction of a wire segment and subtree

b) circuit equivalent of (a)

7 Experimental Results

To investigate the validity of this methodology, the clock optimization framework and supporting algorithms given above have been implemented in C and were run on several test cases. In Section 7.1, the results from [20] are revisited to demonstrate the advantages of the hierarchical decomposition. Then, the merits of our new simultaneous application of the topology search, routing, and wiresizing are demonstrated in Section 7.2. Section 7.3 gives a detailed look at the new balanced wiresizing scheme and Section 7.4 demonstrates the complete framework on a large multistage ASIC design.

7.1 Hierarchical Decomposition

To investigate the validity of the hierarchical decomposition the standard set of Tsay benchmarks [4] have been used. Our intention here is to compare the solutions obtained by splitting the problem to those obtained from a *flat* circuit approach. The optimizer was first run on the benchmark set to produce flat single stage designs. The same benchmarks were then broken into local and global problems and solved in tandem by our algorithms. All routes use minimum wire widths, although wiresizing will be considered in the next section. Table 1 shows the difference in wireability and performance between the flat solutions and the split versions.

Table 1. Comparison of flat and split solutions.

benchmark		flat			hierarchical decomp.		
	sites	wire (Kum)	delay (ns)	skew (ps)	wire (Kum)	delay (ns)	skew (ps)
r1	267	124	1.13	2	115	0.42	28
r2	598	242	3.28	17	231	1.10	41
r3	862	310	3.78	27	294	1.24	46
r4	1903	625	14.4	38	596	4.53	34
r5	3101	952	26.5	4	891	8.40	47

First, we note that even our flat solutions compare favorably with the best previously disclosed solutions in [8][9], *e.g.*, the average wirelength improvement is about 2%. But, the shortest-path wiring used at cluster level in our hierarhical solution saves an average of 6% over the our best flat solutions. This reduction of leaf wire use also reduces the downstream loading capacitance and improves the performance of the network by more than 3X in most cases. Such a reduction in delay can reduce the process induced skew. Finally, the sacrifice to the timing is minimal as the skew (worst-case intra-cluster skew + global skew) is less than 50ps for all designs. In all of these cases, the designs have been optimized without the use of *snake wire*. The small amount of skew introduced by omitting this extra wire shows that such detour paths are unnecessary.

7.2 Tandem vs. Simultaneous Routing and Sizing

To demonstrate the merits of our new simultaneous application of the topology search, routing, and wiresizing, the circuits from the above hierarchical decomposition results have been wiresized using a single-pass delay optimizer, similar to the one described in [5]. This wiresizing algorithm is not the same as the one presented in Section 6, but simply finds a minimum delay wiresizing, without the electromigration constraints. The same circuits were then re-optimized with this single-pass wiresizing delay optimizer in the inner-loop of the annealing. A comparison of the results is given below.

Table 2. Comparison of sized solutions.

benchmark		tandem			simultaneous		
	sites	wire (Kum)	wire area (Kum^2)	delay (ns)	wire (Kum)	wire area (Kum^2)	delay (ns)
r1	267	115	153	0.13	117	148	0.12
r2	598	231	268	0.75	231	292	0.26
r3	862	294	409	0.27	297	380	0.29
r4	1903	596	788	0.77	608	765	0.73
r5	3101	891	1124	1.36	905	1121	1.21

In these results, wire area has been included as it is a good figure of merit when comparing different wiresizing solutions. The skew results have been omitted as they remain below 50ps in all cases. First, notice that the wirelength, wire area, and delay results are very similar for both the tandem and the simultaneous solution strategies. In one case (r2), however, the performance of the tandem solution is nearly three times that of its simultaneously solved counterpart. Upon further inspection, we have found the problem to lie with the selected topology for this circuit. This suggests that performing simultaneous topology, routing, *and* sizing will avoid poor topology choices which can occur if the steps are kept separate.

7.3 Balanced Wiresizing Scheme

The next experiment will demonstrate the new wiresizing scheme of Section 6 and validate equation (6). Here, the r2 benchmark of Tsay [4] has been selected from the optimized flat circuits of Table 1 as an example of a large single stage clock circuit. When applied to this circuit, equation (6) produced an optimal width of 10.5um for the widest wire segment. Here, the driver impedance is Ro = 10 Ohms. The tree was then sized with no lower bound restriction on the wire widths. Although unrealistic without such a width restriction, this sizing produces the theoretical *minimum delay sensitivity* circuit. The wires were then resized with a minimum wire width limit of w_{min} = 1um imposed, which is consistent with the technology of this benchmark. Figure 8 compares the total delay of each circuit (driver + interconnect) as all wire widths are varied from 50% to 150% of their nominal values. This plot shows that equation (6) does indeed predict the optimal width for minimum delay sensitivity as well as the impact of the minimum wire width restriction.

The sized circuit with the minimum wire width restriction was extracted and simulated with HSPICE and the peak currents and current densities of the upper 14 wire segments in the tree were measured and are given in Figure 9 (the current density values are in parentheses). All current values are in mA and all current densities are in mA/um^2. The original unsized circuit (minimum wire widths for

all segments) has been included for comparison. These figures clearly show how this sizing scheme equalizes current densities across the interconnect network.

Figure 8. Delay sensitivity minimization.

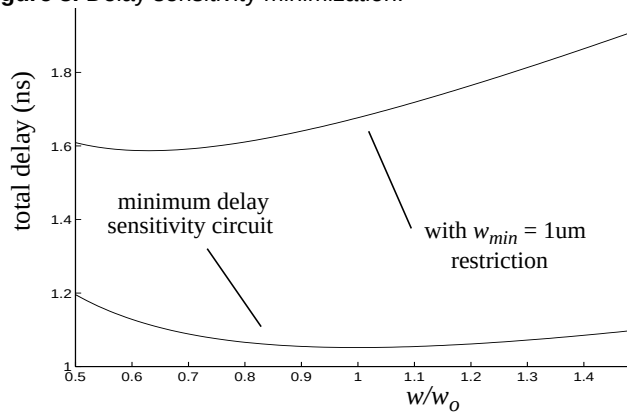
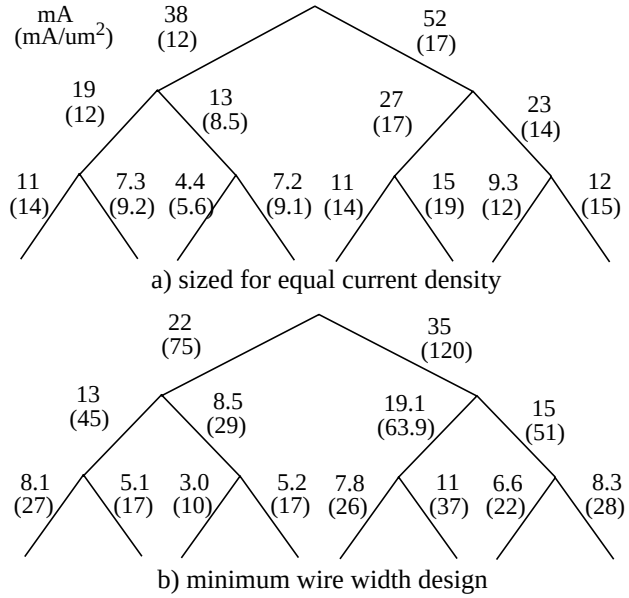


Figure 9. Peak segment currents and current densities.



7.4 Large ASIC design

A 16,818 latch ASIC design on a 1.2cm X 1.2cm chip has been obtained from the IBM ASIC foundry of Burlington, VT, and was processed by these techniques. Many of the detailed specifications of this circuit are regarded as confidential and could not be released. As a result, the circuit has been cast in a 0.35um technology, giving each latch an input capacitance between 1fF and 5fF (ρ = 42 mOhm um, ε = 0.35 fF/um^2, h = 0.15 um). Realistic circuit performance specifications were selected for this design and are summarized as follows:

Table 3. Performance specs for ASIC design.

	design target
performance	200MHz (T = 5ns)
skew tolerance	500ps

The skew tolerance here stems from the rule of thumb that the skew should not be permitted to impact the design by more than 10% of the cycle time.The circuit was clustered with these fan-out and skew limitations: C_{target} = 500fF, S_{limit} = 50ps. All of the clusters are routed with wires of minimum width (0.35um) for wireability and power concerns. The output impedance of the repowering buffers used gives an optimal maximum wire width of 0.04um for this design, which is too small to be realized. We know that by selecting any larger wire width, the delay will only increase and the sensitivity will be roughly constant across the design space (see Figure 6). As a result, the maximum wire width for the global interconnects has been set to 1.4um (4X that of w_{min}) to meet the delay target of 5ns.

The resulting layouts of the local and global distribution are given in Figure 11 and the physical details are summarized in Table 4. Note that the chip layout is very detailed with 16,818 latches, so only the lower 1mm X 1mm portion of the latch clustering is shown. The solve time for this circuit was 1.1 hours for the local distribution and 8.0 hours for the global distribution on an IBM PowerPC 850 running AIX 4.1. This circuit was then extracted and simulated using HSPICE. Both the estimated and simulated performance of the circuit are given in Table 5.

Table 4. Summary of 16K latch circuit physical design.

tree wirelength (Kum)	521	no. of clusters	1032
tree wire area (Kum^2)	345	avg. latches/cluster	16
cluster wirelength (Kum)	1,465	no. of buffer levels	5
total wirelength (Kum)	1,986	no. of buffers	87

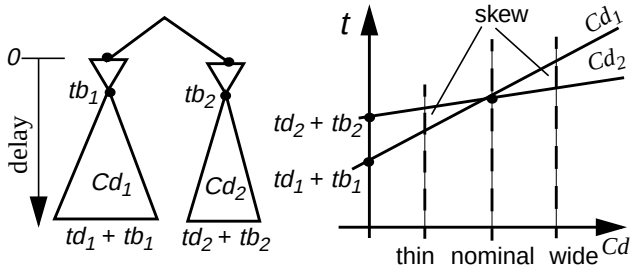
Table 5. Summary of 16K latch circuit performance.

	predicted	HSPICE
delay (ns)	5.32	4.76
skew (ps)	224	310

Although the actual average root-to-latch delay (measured from the input of the central driver to the input of the latches) was below the performance target (5ns), both the actual skew and average rise-time were higher than expected. Nonetheless, the optimization does a good job of balancing the buffer loading and delays, resulting in low predicted and actual skew. Further, the accuracy of the modeling is sufficiently good and does not accumulate significant error over the many stages of this large design.

As a final experiment into the quality of the circuit generated, we investigate the effect of uniform process variations on the interconnect design. If the interconnect distribution has been well designed and balanced, then we should expect that only a small amount of skew will be introduced when the design is exposed to uniform over- and under-etching. To clarify this, consider the illustration of load and delay matching in Figure 10.

Figure 10. Delay mismatch due to wire width variation.

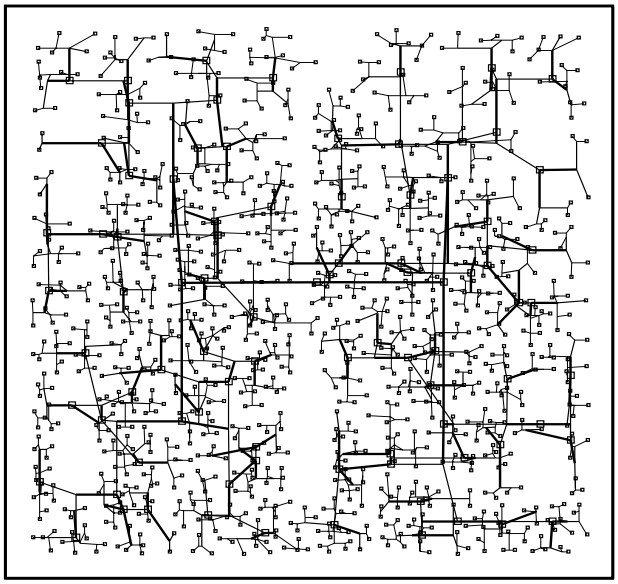


a) matching two buffers b) load line of (a)

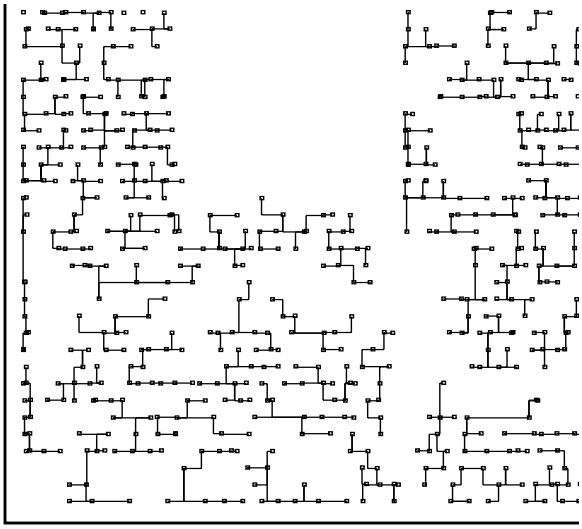
In this vertical delay diagram, two identical buffers with differ-

ent subtree loads (Cd_1 and Cd_2) are being matched. In this example, subtree 1 is larger (in wirelength and wire area) than subtree 2 but they are both routed in such a way as to equalize their total interconnect and buffer delays (*i.e* $td_1 + tb_1 = td_2 + tb_2$). As a result, any wire width variations will cause a greater change in the path delay of buffer and subtree 1 ($td_1 + tb_1$) then in buffer and subtree 2 ($td_2 + tb_2$), as illustrated in Figure 10b. A similar argument can be made for device variations (where the output impedance varies instead of the capacitive load), so the minimization of this type of effect is desirable.

Figure 11. ASIC latch clustering and multistage tree layout.



a) global tree layout



b) cluster sample
(lower left 1mm X 1mm corner)

To see the effect of over- and under-etching on the ASIC design, each interconnect in the layout was varied by +/- 0.1um. Further, the circuit was also simulated at the extreme process corners (fast N and P devices and slow N and P devices) to observe the effect of uniform device variations on the performance. The results are shown in Table 6.

Table 6. Effect of uniform process variations.

	nominal	over-etched (-0.1um)	under-etched (+0.1um)	fast N and P	slow N and P
delay (ns)	4.76	4.51	4.99	3.91	6.58
std dev. (ps)	58	64	55	63	54
skew (ps)	310	323	305	327	285

For this design, the devices cause more delay variation than the interconnects. The interconnect-caused delay variations are within +/- 250ps while the device-caused variation is about an order of magnitude larger. Nonetheless, the standard deviation and skew numbers indicate that the distribution of delays does not change much as the various loads tend to increase or decrease together. This is a result of both the optimization and wiresizing scheme. The optimizer balances delays by finding similar subtrees for all of the buffers at each stage. This creates interconnect subtrees which are of similar size across any given buffer level. Further, the wiresizing scheme tends to create similar segment delays across any level of the tree by enforcing the same wire width on segments which are of roughly the same length. Taken together, these properties tend to make the clock circuit look similar, both geometrically and electrically, along any root-to-latch path. These results demonstrate this and show the benefits of such a design approach in the face of uniform process variations.

8 Conclusions

We have presented a comprehensive clock circuit optimization framework for ASICs. This approach has been motivated by the shortcomings of the current multistep design methodology, which are exhibited when faced with the aggressive performance and manufacturability concerns of modern submicron ASIC technology. This methodology allows for the simultaneous application of topology selection, routing, wiresizing, and buffer insertion algorithms for the optimization of multistage clock circuits. A hierarchical decomposition of the circuit has been used to reduce the complexity of these large multistage ASIC designs. This methodology has been applied to a realistic ASIC design of 16,818 latches in a 0.35um process and has produced a clock distribution with a performance level over 200MHz and a skew of less than 400ps. Further, these skew results remain tight under both interconnect and device process variations. These results were obtained in less than 10hours of CPU time on an IBM PowerPC 850 and have been verified by HSPICE. In future work, we will explore methods for optimization with respect to cross-chip non-uniform variations. Further, improving the accuracy of the linearized device models used seems limited. Therefore, an investigation into either delay look-up tables or an accurate wire and device sizing postprocessing step is prudent.

Acknowledgment

We are grateful to David Hathaway of IBM for his many insightful discussions about clock circuit design methodologies and to Dr. Micheal Trick of IBM for providing the ASIC design. We would also like to thank John Shewchuk of the Computer Science Department at Carnegie Mellon University for providing his tool, *Triangle*, which produced the Delaunay triangulations used in this work. Finally, we would like to acknowledge Dr. Ren-Song Tsay of Avant!, Inc. for providing the benchmark latch placements.

References

[1] H. Bakoglu, *Circuits, Interconnections, and Packaging for VLSI*. Reading, MA: Addison-Wesley, 1990.

[2] M. Jackson, A. Srinivasan, and E. Kuh, "Clock routing for high performance ICs," *Proc. of ACM/IEEE DAC*, pp. 573-579, June 1990.

[3] W. C. Elmore, "The transient response of damped linear networks with particular regard to wideband amplifies," *Journal of Applied Physics*, Volume 19, pp. 55-63, January 1948.

[4] R.-S. Tsay, "An exact zero-skew clock routing algorithm," *IEEE Trans. on CAD*, Vol. CAD-12, No. 2, pp. 242-249, February 1993.

[5] M. Edahiro, "Delay minimization for zero-skew routings," *Proc. of IEEE ICCAD*, pp. 563-566, November 1993.

[6] T.-H. Chao, J.-M. Ho, Y.-C. Hsu, K. Boese, and A. Kahng, "Zero-skew clock routing with wirelength minimization," *IEEE Trans. on Circuits and Systems*, vol. 39, pp. 799-814, November 1992.

[7] K. D. Boese and A. B. Kahng, "Zero-skew clock routing trees with minimum wirelength," *Proc. of IEEE International Conference on ASICs*, pp. 17-21, September 1992.

[8] M. Edahiro, "A clustering-based optimization algorithm in zero-skew routings," *Proc. of ACM/IEEE DAC*, pp. 612-616, June 1993.

[9] N.-C. Chou and C.-K. Cheng, "Wire length and delay minimization in general clock net routings," *Proc. of IEEE ICCAD*, pp. 552-555, November 1993.

[10] S. Pullela, N. Menezes, and L. T. Pillage, "Reliable non-zero skew clock trees using wire width optimization," *Proc. of ACM/IEEE DAC*, pp. 165-170, June 1993.

[11] N. Menezes, A. Balivada, S. Pullela, and L. T. Pillage, "Skew reduction in clock trees using wire width optimization," *Proc. of IEEE CICC*, pp. 9.6.1-9.6.4, May 1993.

[12] S. Pullela, N. Menezes, J. Omar, and L. T. Pillage, "Skew and delay optimization for reliable buffered clock trees," *Proc. of the IEEE ICCAD*, pp. 556-562, November 1993.

[13] P. O'Brien and T. Savarino, "Modeling the driving-point characteristic of resistive interconnects for accurate delay estimation", *Proc. of the IEEE ICCAD*, pp. 512-515, November 1989.

[14] J.-M. Ho, G. Vijayan, C. Wong, "New algorithms for the rectilinear steiner tree problem," *IEEE Trans. on CAD*, Vol. 9, No. 2, pp. 185-193, February 1990.

[15] S. Ganguly and S. Hojat, "Clock distribution design and verification for PowerPC microprocessors," *Proc. of the IEEE ICCAD*, pp. 58-61, November 1995.

[16] A. Rincon, M. Trick, T. Guzowski, "A proven methodology for designing one-million-gate ASICs," *Proc. of the IEEE CICC*, pp. 4.1.1-4.1.8, May 1996.

[17] J. Lam and J. M. Delosme, "Performance of a new annealing schedule", *Proc. of ACM/IEEE DAC*, pp. 306-311. June 1988.

[18] N. Menezes, R. Baldick, L. Pileggi, "A sequential quadratic programming approach to concurrent gate and wire sizing", *Proc. of the IEEE ICCAD*, November 1995.

[19] G. Ellis, *The Physical Design of Clock Circuits*, Ph.D thesis, Department of Electrical and Computer Engineering, Carnegie Mellon University, 1997.

[20] G. Ellis, L. T. Pileggi, R. A. Rutenbar, "A hierarchical decomposition methodology for single-stage clock circuits", *Proc. of the IEEE CICC*, pp. 7.1.1-7.1.4, May 1997.