Low-Power Buffered Clock Tree Design

Ashok Vittal, *Member, IEEE*, and Malgorzata Marek-Sadowska, *Fellow, IEEE*

***Abstract*—We address the problem of low-power reliable clock tree design in this paper. We study the scaling of clock power dissipation with increasing die sizes, number of receivers, and operating frequencies. The analysis shows that buffering only at the root of the tree is not scalable. However, when buffered clock trees are allowed, the classical *H* tree is suboptimal in terms of both area and power dissipation. We show that the new power minimization problem is NP hard, and we propose a novel algorithm for low-power clock network design. Our algorithm designs the tree topology and inserts buffers simultaneously. The clock skew is guaranteed to be small in the presence of correlated process variations. Wire sizing is used when necessary, and clock skew can be inserted intentionally if required. The results obtained by our algorithm on benchmark problem instances are significantly better than previous approaches in terms of power dissipation, wire length, rise times, and buffer area. We report HSPICE simulation results for the clock trees designed by the new algorithm.**

***Index Terms*—Clock distribution, power estimation.**

I. Introduction

CLOCK routing is an important problem in the layout design of synchronous digital systems as it influences the correctness, area, speed, and power dissipation of the synthesized system. The design of the clock distribution network determines the clock skew, thus affecting the maximum attainable clock frequency. It is important to ensure small clock skew to avoid race conditions in the design—this has to be achieved in spite of process variations. Race condition avoidance is critical because it cannot be remedied by increasing the clock period. Clock tree design also determines the rise times of the signals at the clocked elements, which again limits the maximum frequency of operation [1]. The clock net is usually one of the first nets to be routed, and constitutes a blockage for nets routed subsequently. As it is one of the largest nets, the area occupied is also a concern.

Many high-speed circuit techniques require particularly reliable clock tree design. Architectural-level pipelining, fine-grain pipelining [31], [10], and the trend of increasing system complexity all increase the number of clocked elements. Pipelining also reduces the logic depth, and creates many short paths between sequentially adjacent flip-flops, making skew constraints difficult to meet. Besides, in test mode, many flip-flops are hooked together into a scan chain [5]. The skew requirements are stringent for the scan clock tree as every path during test mode is a short path. Further, the increasing complexity of designs leads to larger die sizes, again making good clock tree design critical.

The increasing popularity of portable applications, the acceleration of many failure mechanisms [1] at high operating temperatures, and the high costs of system cooling have increased the importance of low-power design. The clock net accounts for a significant fraction of the system power dissipation as it switches most frequently, and involves charging (and discharging) a huge capacitive load. The clock network power dissipation is typically one third of the total power dissipation in CMOS VLSI systems [24], and constitutes more than half the total power in some designs. It is therefore important to optimize the power dissipated in the clock tree.

There has been considerable previous work on clock routing. The method of means and medians [19] used purely geometric considerations for automatic clock net synthesis. This could lead to large skews. Subsequent research has aimed to get zero skew under a better delay model [28], better algorithms for the problem [12], [21], minimizing the wire length [2], [3], [14], path-delay optimization [6], [7], and better time complexity [13]. Reliability issues were addressed in [26], and a buffer redistribution algorithm was proposed in [6]. Planar clock net design was considered in [32]. None of these algorithms considered the power dissipated by the clock network. Besides, rise time constraints imposed by design specifications were not addressed. Buffered clock tree synthesis was considered in [8]. However, buffers were inserted as a postprocessing step after topology design, and the power was not optimized. The approach in [30] also considers buffer insertion as a postprocessing step.

This paper addresses the issues mentioned above, and is organized as follows. In Section II, we formulate the low-power clock routing problem. In Section III, we analyze the buffered H tree to demonstrate the need for buffer insertion at internal nodes of the tree. We also study the scaling behavior of clock tree power dissipation in regular H trees. This important special case highlights the need for low-power clock tree design. Limits on system size imposed by synchronicity are also derived. These results are of independent interest. Section IV discusses issues related to buffer insertion in general clock trees with irregular clocked element positions. In Section V, we prove that the problem is NP hard by reducing the minimum Steiner tree problem to it. The reduction gives a straightforward algorithm to modify any tree into a zero skew tree by inserting buffers. Reliability under process variation is addressed in Section VI. A new algorithm for low-power clock distribution is presented in Section VII, motivated by

Manuscript received July 10, 1996. This work was supported in part by the Defense Advanced Research Projects Agency under Contract DABT63-93-C-0039, the National Science Foundation under Grant MIP 9419119, and LSI Logic and Rockwell through the California MICRO Program. This paper was recommended by Associate Editor K. Cheng.

A. Vittal is with Silicon Graphics Inc., Mountain View, CA 94043 USA.

M. Marek-Sadowska is with the Department of Electrical and Computer Engineering, University of California, Santa Barbara, CA 93106 USA.

Publisher Item Identifier S 0278-0070(97)09001-5.

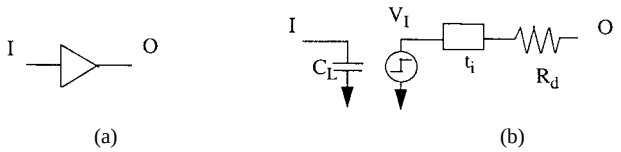

Fig. 1. Buffer modeling. (a) Buffer. (b) Equivalent circuit.

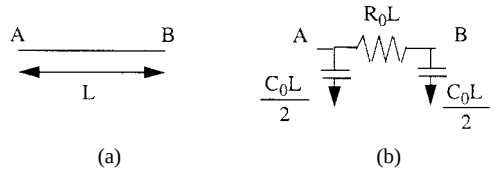

Fig. 2. Interconnect modeling. (a) Wire segment of length L. (b) Pi-equivalent circuit model of the wire.

the analysis and observations of Sections III–V. The algorithm simultaneously designs the topology and inserts buffers. Section VIII presents results obtained by our algorithm on benchmarks for several technologies. Section IX concludes.

II. Problem Formulation

We wish to design a clock distribution network given the clocked element (receiver) positions and technology parameters. We want the clock network to dissipate minimal power and satisfy rise time constraints at all receivers. The skew should remain small, even under process variations.

In order to calculate the delay, rise times, and power dissipation, we model a buffer by the equivalent circuit shown in Fig. 1 and a wire segment of length L by the equivalent circuit shown in Fig. 2. R_d and C_L are the buffer driver resistance and input capacitance, respectively. L is the interconnect length. R_0 and C_0 are the interconnect resistance and capacitance per unit length, respectively. A delay element is added in series with the driver resistance to account for the internal delay of the buffer.

The delay is defined to be the Elmore delay [15], [27], a model that has found applications in many timing simulators. The expression for the Elmore delay is

$$D_i = \sum_{k \in P_i} R_k C_k \tag{1}$$

where D_i is the delay to the ith sink, P_i is the set of resistances on the unique path from the root to the ith sink, R_k is the resistance of any resistor on the path, and C_k is the downstream resistance seen by R_k. The Elmore delay is a *dominant time constant approximation* to the circuit response at any sink. Thus, if we make the same approximation to calculate the rise times, we get the 10%-to-90% rise time to be

$$R_i = 2.2 \cdot \tau_i \tag{2}$$

where R_i is the rise time of the signal at the ith node and τ_i is the associated subtree time constant.

The power dissipated by a clock net can be attributed to the charging and discharging of the wiring and load capacitances through the interconnect resistance and driver resistance and to the static power dissipated, if any, by the buffers

$$P = P_{\text{static}} + P_{\text{dynamic}} \tag{3}$$

$$= P_0 \cdot \sum_{\text{buffers}} k_i + f \cdot C_0 \cdot V^2 \cdot L(T). \tag{4}$$

Here, P_0 is the static power dissipated by a minimum size buffer, k_i is the size of the ith buffer, f is the frequency of operation, V is the voltage swing, C_0 is the capacitance per unit length, and $L(T)$ is the tree length.

The clock network design determines the buffer sizes, their locations, and the interconnect topology. It therefore affects the summation in the first term and the wire length in the second term. For CMOS VLSI, the static power consumed by the buffers is negligible, so that the problem reduces to one of minimizing the total capacitance (contributed by both wiring and buffers). In ECL, the static power dissipated by the buffers dominates. For multichip modules, both dynamic and static power consumption may be equally important. Guided by these considerations, we formulate the problem as one of finding the interconnect topology, buffer locations, and sizes to

$$\text{minimize}\left(L(t) + \alpha\left(\sum_{i=1}^{B} K_i \right) \right) \tag{5}$$

where α is a technology-dependent constant (given), $L(T)$ is the wire length of the synthesized buffered tree T, B is the number of inserted buffers (determined by the design, not given), and K_i is the size of the ith buffer. Note that buffer sizing is allowed by our formulation as K_i and B are not prespecified. While we do consider wire width optimization in Section III, the formulation above assumes uniform wire widths. We do not restrict the sizes of clock buffers to be discrete because the widths of inserted buffers are typically large, and a continuous formulation is sufficient.

The skew constraints are

$$(\forall (i,j)) \qquad D_i = D_j \tag{6}$$

where D_i and D_j are the Elmore delays to the ith and jth sinks, as before.

The rise time constraints are

$$(\forall i) \qquad (T_i \leq T_{\max}) \tag{7}$$

where $T_{\max}$ is a specified maximum allowed rise time and T_i is the rise time of the signal at the ith sink. Note that this constraint is extremely important as any recognizable clock must have a clock period of at least three times the 10%-to-90% rise time. The rise time of classically designed clock nets imposes a limit on the frequency of operation, even if logic delays are small.

We call problem (5) with (6) and (7) as constraints the *power optimal buffered clock network design* (POBCND) problem. In the next section, we analyze regular H trees to show the need for buffer insertion at internal nodes of the tree.

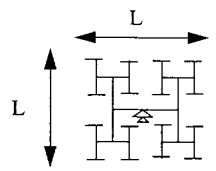

Fig. 3. An H tree with $N(=32)$ clocked elements distributed uniformly on an $L \times L$ die.

III. Power Estimates for Buffered H Trees

In this section, we provide *analytical* power estimates for two clocking strategies for regular clocked element arrays—the H tree with tapered buffers at the root and the H tree with distributed buffering. This analysis is important for the following reasons.

- Feasibility studies can use our expressions to get ballpark numbers during the early phases of the design process.
- The analysis results can be used by high-level synthesis steps (like scheduling) and logic synthesis steps (e.g., retiming) to evaluate their impact on power dissipation.
- They are useful in identifying bottlenecks, and also help quantify tradeoffs that can be made. Besides, asymptotic scaling behavior (with number of sinks/feature sizes/die size) can be studied in order to determine limits on system size imposed by synchronicity.

Consider an H tree with N nodes distributed regularly on an $L \times L$ die as shown in Fig. 3. It can be shown [29] that the asymptotic wirelength is $W = 1.5 \cdot L \cdot \sqrt{N}$. The total clock sink load is $C_s = N \cdot C_L$, where C_L is the load capacitance. Thus, a lower bound on the power dissipation of this clocking scheme is $P_C = f \cdot V^2 \cdot (C_0 \cdot W + C_s)$, where f is the clock frequency, V is the logic swing on the clock lines, and C_0 is the capacitance per unit length of interconnect. The total power dissipation is given by (8), where P_B is the power dissipated by the buffers. With

$$P = P_C + P_B \tag{8}$$

tapered buffers at the root [23], [20], [10], the buffer power dissipation is $P_{B,\,\text{taper}} = 0.5 \cdot f \cdot V^2 \cdot (C_0 \cdot W + C_s)$. Thus, for the tapered buffer scheme, the total power dissipated in the clock tree is $P = 1.5 \cdot f \cdot V^2 \cdot (C_0 \cdot W + C_s)$. Interconnect resistance has been ignored in this calculation. When included, the drive strengths of the buffers will have to be increased, leading to larger buffer power dissipation. The expression above is, therefore, a lower bound for the clock tree power dissipation.

We now analyze the H tree including interconnect resistance. As the H tree is symmetric, nodes at equal levels can be shorted together. It reduces the analysis to that of the RC line shown in Fig. 4.

e delay from the root to the leaf for the RC line in Fig. 4 (with minimum width wiring) is given by (proof omitted for brevity)

$$\tau = R_0 \cdot L \cdot (0.21 \cdot N \cdot C_L + 0.28 \cdot \sqrt{N} \cdot C_0 \cdot L). \tag{9}$$

With $C_0 = 0.15\,\mu$m, $R_0 = 0.007\ \Omega/\mu$m, $N = 10\,000$, $L = 1$ cm, and $C_L = 2$ fF (typical numbers for a submicron ASIC), we get a time constant of 3.2 ns (a bandwidth of about 90 MHz). This is clearly unacceptable for clock frequencies above 100 MHz. The use of thicker metal lines and more conductive materials (Al–Cu instead of Al, as in the Alpha microprocessor) will yield smaller R_0, with a proportional increase in bandwidth. However, the trend of increasing N and L (the DEC 21 164 has 340 665 receivers over a 16 mm $\times$ 17 mm die) still calls for optimized clock distribution design.

The shape of the RC line in Fig. 4 has much poorer delay characteristics than an optimally sized RC line [4], i.e., the shape is an increasing exponential instead of a decaying exponential function. Wire sizing will, therefore, considerably improve the bandwidth, while significantly compromising the power dissipation. The driver resistance (at the root) should also be small if the product of driver resistance and load capacitance is to be small. This conflicts with the need to avoid ringing due to the net being overdriven [25], i.e., the reflection coefficient looking into the driver from the transmission line should not be negative. The constraint to avoid overdriving the net is, therefore, $R_d \geq Z_0$. With a line impedance of about 20 Ω, the driver resistance-load capacitance product is 4.9 ns for the ASIC example. Wire sizing will only increase this time constant. The driver at the root strategy is inherently unscalable.

There is a quadratic dependence on die size in the dominant term of the time constant expression in the example above. This indicates that buffer insertion can help. The total driven capacitance is given by (10) (proof again omitted for brevity). Given the required time

$$C_l = N \cdot C_L + 1.5 \cdot C_0 \cdot L \cdot \sqrt{N} \tag{10}$$

constant τ (typically a fifth of the clock period), the total amount of buffering (the inverse of the sum of buffer conductances) required is

$$R_{\text{total}} = \frac{\tau - (R_b \cdot C_b)}{C}. \tag{11}$$

R_{total} is inversely proportional to the total buffer power dissipation. Using this in (8), we compute the total clock power dissipation. Fig. 5 plots the variation of total clock power with the number of sinks for various die sizes when the clock frequency is constant (400 MHz) for a CMOS technology with 1 μm gate lengths. The buffers used have a resistance of 450 Ω and an input capacitance of 260 fF. At 400 MHz, HSPICE simulations show that the buffer dissipates 0.5 mW total power. The figure also shows maximum allowed power dissipation assuming that the clock power is 40% of total power and the maximum allowed power dissipation density is 20 W/cm^2 (a typical number for air cooling [18]). As the number of sinks (N) grows, the capacitive wiring load and the corresponding buffering requirements grow. Further, when the die size increases, the lengths of lines gets longer, leading to larger power dissipation. The clock power dissipation curve for $L = 1$ cm is about to transition from a square-root dependence on the number of sinks to a linear dependence due to the total receiver capacitance. The curve shows that 1 million clock receiver networks may not be achievable in 1 cm $\times$ 1 cm dice unless the supply voltage is scaled down further or better (and

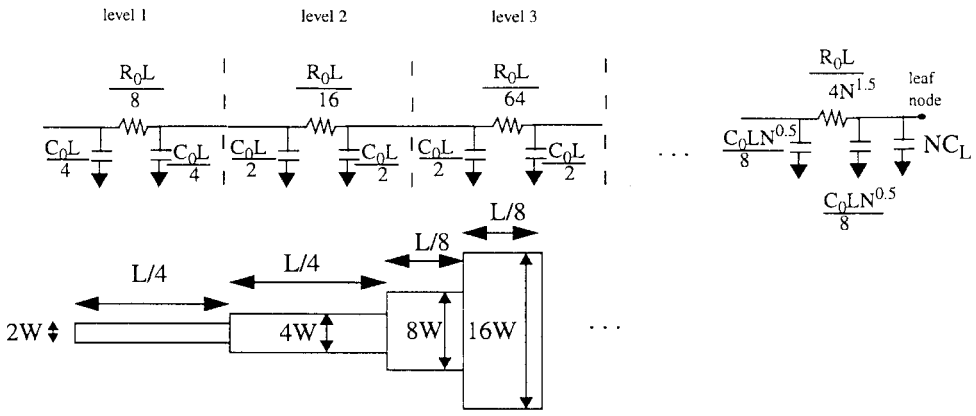

Fig. 4. The RC line equivalent of the H tree. Level 1 corresponds to two branches of the tree, level 2 to four branches, etc. Symmetry enables nodes at equal levels to be shorted together.

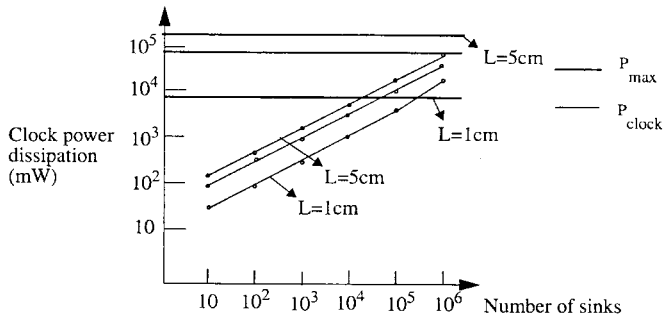

Fig. 5. Scaling behavior of clock power dissipation with number of sinks.

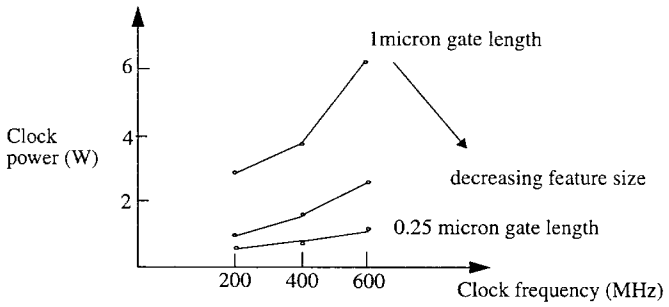

Fig. 6. Clock tree power dissipation variation with frequency.

more expensive) cooling techniques are used. This is because the power dissipation allowed is 8 W, while the clock network dissipates close to 18 W.

Subdividing the clock network into multiple globally asynchronous regimes does *not* help when the total capacitance is dominated by the wiring. This is because subdividing into k regimes leads to a factor of reduction in die size and a factor of $\sqrt{k}$ reduction in the number of receivers for each regime. The wiring capacitance of each regime is therefore reduced by a factor of k, as seen from the second term in (10). The first term in (10) also goes down by a factor of k. However, there are k regimes, and the total power dissipated to charge the load has not changed.

The scaling of clock power with clock frequency is also of interest, particularly with technologies of smaller feature size. Fig. 6 shows the scaling of total clock power dissipation with clock frequency for various feature sizes, keeping the number of sinks constant at 10^5. With finer feature sizes, the die size is smaller for the same (scalable) design. The short-circuit current gain cutoff frequency of the transistor is higher, so that interconnect resistance becomes a significant bottleneck. Our analysis also shows that *clock power dissipation scales superlinearly with clock frequency*. This is because both f and the switched capacitance increase. The switched capacitance increases because the rise time constraint is now more stringent and larger buffering is required.

The power dissipated in the buffers is not negligible, so clock routers should be aware of the implications on power dissipation when buffering. This is especially the case when the operating frequency goals are aggressive. Scaling down feature sizes clearly decreases power dissipation because wiring and load capacitances get smaller, while driver resistances decrease. The wire length of the tree should also be optimized, if possible. Buffering in non-H-tree topologies for wire length reduction is considered next.

IV. Buffer Insertion for Wire Length Optimization

In this section, we show that buffers can be used to significantly *reduce the wire length* of the clock tree. We also show how buffer insertion for non-H-tree topologies can still maintain zero skew.

Consider the regular 16-point clock routing instance shown in Fig. 7(a). For such an instance, the H tree [16] is popular. This incurs a wiring cost of 18 units. By inserting buffers as shown in Fig. 7(b), the wiring cost can be reduced to 15. The buffer sizes are chosen so that zero skew is maintained. Asymptotically, the H tree for a regular grid of clock entry points requires $1.5D\sqrt{N}$ wire length where D is the length of the die and N is the number of clocked elements. On the other hand, the minimum rectilinear Steiner tree needs only $D\sqrt{N}$ wire length. Thus, 50% savings in wire length is possible by just inserting buffers appropriately, while maintaining zero skew.

A natural question at this point is: How much wire length savings are possible for irregular pin distributions? The bar graph in Fig. 8 answers this question. We compare the best previous clock routing results with a minimum Steiner tree heuristic [22]. We see that even for irregular pin distributions,

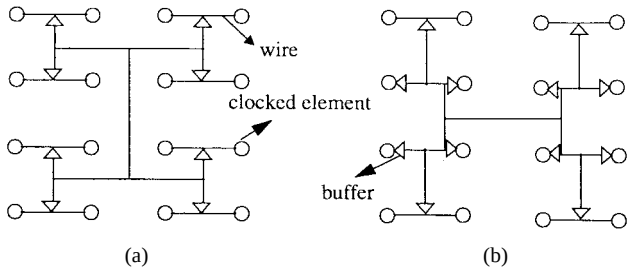

Fig. 7. Wire length optimization possible using buffers. (a) The H tree uses 18 wire length units. (b) A minimum Steiner tree with inserted buffers uses 15 units of wirelength. The buffers are used as variable delay elements to make the delays to the sinks equal. Buffer sizes are different.

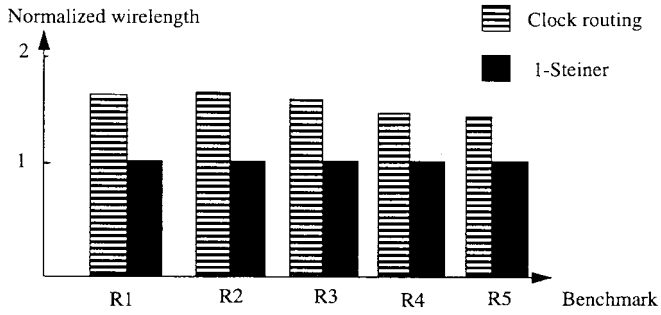

Fig. 8. Wire length savings possible on clock-routing benchmarks.

the wire length savings possible range from 40 to 60%. The results are normalized with respect to the 1-Steiner [22] wire lengths.

The examples above give us *existence proofs* of solutions that are better, when buffers are inserted in the tree. Buffer placement for minimal Elmore delay given an initial route using dynamic programming has been reported in [17], and delay optimal buffering for clock trees given an initial zero skew route using similar techniques has been proposed in [8]. For the H-tree example, however, note that we can never obtain the wire length optimal solution by inserting buffers after topology design. Similarly, inserting buffers into a minimum Steiner tree may result in unacceptable buffer sizes. The algorithm should design the topology and insert buffers simultaneously if we are to get good results.

The opportunities provided by buffer insertion for reducing the wire length can be expressed in terms of the *deferred merge embedding* technique [13], [3], [2]. This method builds a topology bottom up, starting from a set of clock points and successively merging until a zero skew clock tree results. During each merge of subtrees to form a larger subtree, the locus of possible Elmore zero-skew merge points on the Manhattan plane is determined. The delays from the sinks to the root of the new subtree must be equal (for zero skew), and the wire length used must be as small as possible (the distance between the two subtree roots). The set of possible zero-skew merge points under these constraints is then a line segment.

Consider the possibilities if we can add a buffer into either branch of the tree when merging subtrees T_1 and T_2 shown in Fig. 9. The position of the roots on the Manhattan plane is shown in Fig. 9(a) and (d) as circles. When no buffer is added, as in classical deferred merge embedding, the locus of zero-

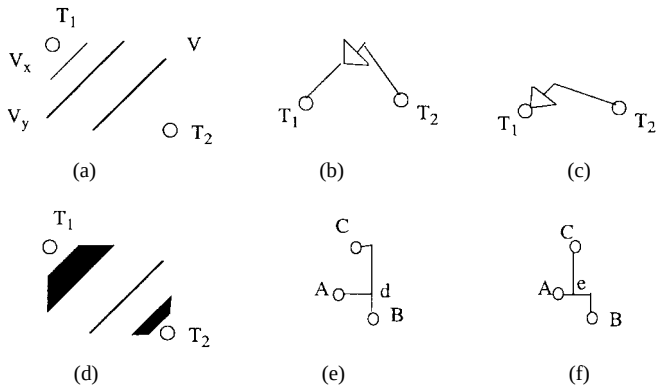

Fig. 9. Buffer insertion during merging. (a) Subtrees T_1 and T_2 are merged. V is the line segment representing possible merge points when no buffer is inserted. V_b and V_c are the line segments for configurations in (b) and (c). (d) Final set of possible merge points. (e) Clock route when no buffers are inserted. (f) Clock route when a buffer is inserted just before A (smaller wire length).

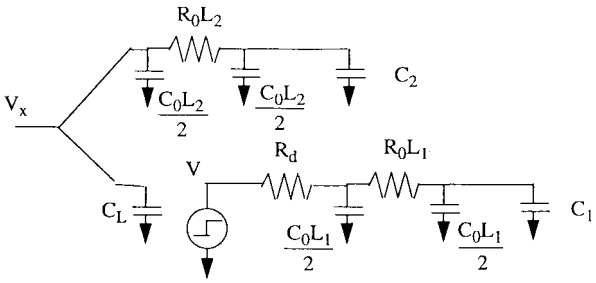

Fig. 10. Equivalent circuit for finding zero-skew merge point.

skew merge points is the line segment V. With a fixed size buffer at the start of the left subtree as shown in Fig. 9(b), we get the line V_x as the set of possible merge points. The zero-skew merge point is closer to T_1 as the buffer has added delay in the left branch. With a buffer just before T_1, as shown in Fig. 9(c), we get the merging segment V_y. We get similar line segments when a buffer is inserted into the right subtree. The complete set of possible merge points is shown in Fig. 9(d). The regions correspond to buffers being added at points in between the beginning and end of the two subtrees. Fig. 9(e) and (f) show how the clock route is improved by this freedom. When clock points A and B are merged without buffers, the closest merge point to C is d. With buffers, the closest merge point to C is e. Wire length savings have resulted.

We now explain how the positions of the line segments V_x and V_y can be calculated. Let the capacitances of the subtrees T_1 and T_2 be C_1 and C_2 and the corresponding delays be D_1 and D_2. Let the distances of the zero-skew merge point from the roots of T_1 and T_2 be L_1 and L_2, respectively. For zero skew, the delay from any sink in the new subtree to the root must be equal after merging. Consider the configuration in Fig. 9(b). The equivalent circuit is shown in Fig. 10. The driver resistance is R_d, the buffer input capacitance is C_L, the resistance per unit length of interconnect is R_0, and the capacitance per unit length of interconnect is C_0. The zero skew equations are

$$D = D_2 + R_0 L_2 \left(C_2 + \frac{C_0 L_2}{2} \right) \tag{12}$$

$$D = D_1 + R_0 L_1 \left(C_1 + \frac{C_0 L_1}{2} \right) + R_d (C_1 + C_0 L_1) \quad (13)$$

where D is the delay to any sink from the root of the new tree.

As we do not want any detour wiring, the merge point should be within the rectangle having the two subtree roots as corners (the "bounding box"). We therefore have

$$L = L_1 + L_2. \quad (14)$$

The new tree capacitance is

$$C = C_{\mathrm{L}} + C_2 + C_0 L_2, \quad (15)$$

L_1, L_2, D, and C are the unknowns, and their values are easily determined from (12)–(15).

Solving similar equations, we get the other delimiting segment V_y. Sliding the position of the buffer along the subtree moves the merge point from one delimiting segment to the other. As the merge point has to be within the bounding box, the zero-skew merge region with left subtree buffering is a polygon as shown in Fig. 9(d). The other polygon in Fig. 9(d) corresponds to right subtree buffering. The two polygons may overlap, and may also include the segment V.

Note that the locus in Fig. 9(d) is for a buffer of given size. A different buffer size results in a similar region of possible merge segments with different delimiting segments. Similarly, wire sizing also changes the positions of the two polygons. Buffer insertion is, therefore, an important degree of freedom. However, it makes clock tree design intractable, as we show next.

V. Computational Complexity

In this section, we show that the decision version of the power-optimal buffered clock tree design problem is NP complete by reducing the minimum Steiner tree problem to it. This result is important because the problem size can be very large.

The problem is in NP as finding the delays and rise times of a given tree can be done in linear time [27]. We need the following lemma to prove the reduction.

Lemma 1: Any tree can be transformed into a zero-skew tree by inserting buffers into the tree.

Proof Outline: Our proof is constructive, and is given by the algorithm in Fig. 11. It involves a linear time traversal that inserts buffers in the tree to equalize delays. The correctness of the algorithm can be proved by induction. The procedure Insert_Buffer involves choosing an appropriately sized buffer at the root of the smaller delay subtree using (12)–(15) in Section IV. For arbitrary delays and capacitances at the two subtrees, a driver resistance can always be found to satisfy the zero-skew (12)–(15). The buffer size changes the value of R_d. Any required driver resistance can be synthesized by varying the buffer size. All we need to do is to solve the zero-skew equations for the value of R_d and find the appropriate size. Note that sizes need not be integers.

We now reduce the rectilinear minimum Steiner tree problem to the POBCND problem.

[Rectilinear Minimum Steiner tree problem]

Instance: Point set P, number L.

```
Algorithm Equalize_Delays
Input: A non-zero skew tree T
Output: A zero skew buffered tree T*
begin
    if isnt_a_leaf(T)
        begin
          Equalize_Delays (left_sub_tree(T))
          Equalize_Delays(right_sub_tree(T))
          if(not_zero_skew(T))
              begin
                 Insert_Buffer(T)
              end
        end
end
```

Fig. 11. Achieving zero skew using buffers.

Question: Is there a Steiner tree whose length is less than L?

Given an arbitrary instance (P, L) of the rectilinear minimum Steiner tree problem, we construct the POBCND problem instance (point set $P, \alpha, 0, R_{\max} = \infty$). The two instances are equivalent. The correctness of the reduction follows from the following theorem.

Theorem 1: P has a Steiner tree of length less than L if and only if the POBCND problem has a solution of cost less than L.

Proof: If P has a Steiner tree of length less than L, the same topology with buffers inserted using Equalize_Delays has zero skew. The cost is L as $\alpha = 0$. The rise time constraints are satisfied as they are absent. This proves that if the Steiner tree problem has a solution, then our POBCND instance also has a solution.

If the POBCND instance has a solution, there is a solution whose cost is less than L. If we remove the inserted buffers, we are left with a Steiner tree whose length is less than L. □

Theorem 1 suggests that it is unlikely that there is a polynomial time exact algorithm for the POBCND problem. In Section VI, we propose a heuristic for our problem. We first need to see how the tree merge of Section IV can handle reliability issues.

VI. Reliability Issues

In this section, we analyze the process variation tolerance of clock routes. We then use the analysis to motivate a new algorithm for clock distribution design.

The reliability analysis in [26] assumes uncorrelated process variations, which leads to pessimistic design. The total skew is allocated equally to all levels, which may not be optimal.

Designs which dissipate less power become possible if the assumption of correlated process variations holds. This is best illustrated using the H-tree example. The wire sized solution returned by the algorithm in [26] is shown in Fig. 12(a). The wires close to the root are four times the minimum width, while the others are twice minimum width. The total wiring capacitance is $8L$. The unlikely process variation

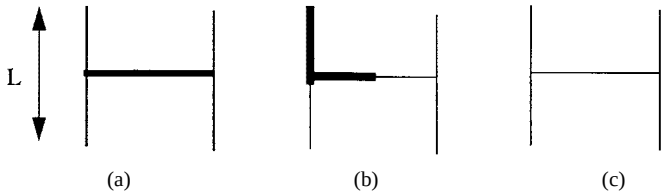

Fig. 12. H tree wire sizing for reliability to process variations. (a) Wire sizing assuming uncorrelated process variations. (b) Worst case random process variation. (c) Wire sizing assuming correlated process variations.

which necessitates this wire widening is shown in Fig. 12(b). Two wire segments are wider, while the other four wire segments are thinner. The wiring capacitance for an H tree with minimum width wires is only $3L$. If process variations are correlated, all wires are wider or thinner by the same amount. The H tree is still symmetric, and the skew remains zero. The necessary layout area and the power dissipation when process variations are correlated are a factor of 2.67 less than the area necessary assuming the uncorrelated model. Besides, a smaller buffer suffices to drive the load.

The algorithms in [26] and [30] choose the wire width of a branch such that the delay through the branch is minimized. This leads to the derivative of delay with respect to width to be zero (the derivative of a function of one variable is obviously zero at the global minimum of the function). However, this optimal wire width is proportional to the square root of the downstream capacitance [26], [30], and becomes too large close to the root of the tree. We consider the problem of choosing the wire width so that the reliability under correlated process variation is within limits, i.e., we assume that the wire widths of all lines in a subtree increase (or decrease) by the same amount w. Our modification for reliable bottom-up tree merging is explained next.

Consider the merge of trees T_1 and T_2 in Fig. 13. We allow merges between trees with an equal number of buffers from root to leaf only. We wish to compute widths w_1 and w_2 such that the worst case skew (under correlated process variation) in the new tree is within a specified fraction of the total delay. Using a Taylor series expansion with only one term, the delay from the root to the leaf is given by $D = D_0 + w \cdot (\partial D/\partial w)$, where w is the wire width variation of *all* wires in the tree. The capacitance is, similarly, $C = C_0 + w \cdot (\partial C/\partial w)$. Each tree is modeled by (delay, capacitance) pairs, and also the corresponding first derivatives with respect to w. The delay and capacitance derivatives for a clock receiver are both zero. The required line widths w_1 and w_2 are found, using (12)–(15) to keep the new delay D's derivative with respect to width below the required fraction of D, and the process continues. There are two new variables w_1 and w_2. There are also two new constraints: the delay from P to a sink in either T_1 or T_2 should both maintain a small derivative with respect to w. The objective is to minimize the increase in layout area.

VII. A Clock-Routing Algorithm

We have shown how bottom-up tree merging can handle buffer insertion and reliability issues while maintaining zero skew. We have also seen that power minimization is NP hard.

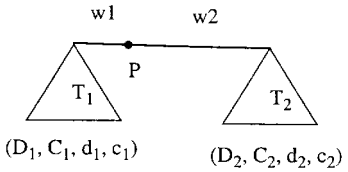

Fig. 13. Merge point selection.

A heuristic which leads to good results is outlined in this section. Several practical issues related to the heuristic are also discussed.

A. The Heuristic

The greedy clustering algorithm of [12] returns unbuffered clock trees with the smallest wire length of the algorithms compared in [6]. It is therefore reasonable to expect a greedy algorithm to work well for our problem as well. Some important changes have to be made, however, as we have to fulfill rise time constraints, achieve a reliable design even under process variations, and utilize the possibility of adding buffers.

- The greedy algorithm involves choosing the move that is locally optimal. During the zero-skew merge operations, we have to choose the merge that results in minimum cost increase. The cost is given by (4).
- When two subtrees are merged without buffers, the locus of possible merge points which do not involve detour routing is a line segment. As discussed in Section IV, with buffers, we have to store two polygons and a line segment as the set of possible merge points. When a merge point corresponding to buffering is chosen, a buffer is being added for wire length savings.
- After having merged two subtrees, we have the possibility of inserting a buffer for future rise time savings. This move should be made only if the rise time constraint is very stringent, or else we end up adding too many buffers. Besides, the move should be made only if there will be rise time savings due to the move. This move is therefore made only if the normalized cost increase entailed by the buffer addition is small compared to the normalized rise time constraint saving and if rise time savings result.
- The rise time constraint should guide the topology design as well, especially if it is difficult to satisfy. When merging subtrees, rise time information is available, and we adaptively modify the cost to reflect the rise time constraint: the classical penalty function approach.
- In order to maintain the same number of buffers on any path from the root of the clock tree to any sink, we only allow merges between equal level trees, where the level denotes the number of buffers from the root to any leaf of that subtree. The initial level of all sinks is zero, and the level of a subtree is incremented when a buffer is inserted at its root.
- The algorithm introduces buffers of only one user-specified size. This is done so that the layout of each of the buffers can maintain the same orientation. In this way, the variation of buffer characteristics across the chip due to isotropic process variations is minimized.

```
Algorithm GRIN
Input: Point set P, parameter a, technology parameters, rise time limit T_max
Output: Buffered zero-skew power-optimal clock tree T
begin
   repeat |P| -1 times
      begin
         minimum_cost_merge();
         if buffering_required() and cost_decreases()
               add_buffer()
      end
   Cascaded_drivers_at_root()
end
```

```
minimum_cost_merge()
   find equal-level pair such that the cost is minimized
      Cost = (ΔL + α · ΔK) / (ΔL_prev + α · ΔK_prev) + ΔT / (T_max − T)
   where K is the buffer size, L is the length and T is the rise time.
   Merge the two nodes to get polygons representing possible merge points
```

```
buffering_required
   if delay(buffer+interconnect) < delay(interconnect)
      return (true)
   else
      return(false)
```

Fig. 14. Algorithm GRIN.

When all buffers are identical, the local environments for the devices within the buffers are the same, so the chip temperatures at each of the devices are similar.

These key ideas lead to the algorithm GRIN (greedy internal buffering) shown in Fig. 14. It is easy to see that Theorem 2 holds.

Theorem 2: The Elmore delays to the clock sinks of the tree designed by GRIN are *all* equal.

Proof: By induction. □

We now show how various practical issues are handled by our clock router. It is necessary for clock routers to handle these issues if they are to make the transition to a synthesis system.

B. Practical Issues

In this subsection, we discuss design methodology issues (the position of our clock router in the design process), and show how our algorithm can handle intentional skew insertion requirements, clock gating, engineering change, and contrast our algorithm with clock meshes.

1) Design Methodology Issues: During clock routing, the latch positions are assumed given. Distributed buffering might cause nontrivial changes to the placement unless special care is taken: in a hierarchical design system, module placements should have empty place holders for buffers. Clock distribution is constrained to use buffers only at these locations. The analysis of H-tree buffering in Section III gives us estimates of the number of place holders required. In order to force the placement to leave place holders at reasonable locations, we perform placement assuming given place-holder locations. The clock network is removed from the netlist during placement, so that the placement objective function does not see extraneous forces.

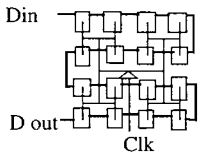

Fig. 15. Sixteen-bit shift register with H-tree clock layout.

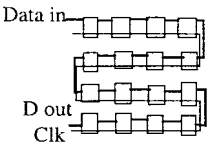

Fig. 16. Shift register with power-optimum clock tree.

2) Intentional Skew Insertion: Skew is intentionally inserted into clock networks in order to improve the timing margins on the chip, and possibly to increase the clock frequency. For each clocked element, there is a phase range over which races are avoided and data have enough time to propagate. Design should ensure that the clock edge occurs within that range for all possible process variations. Our clock distribution scheme handles this by allowing delays to be specified at the clock tree leaves (for nominal design), and offering the guarantee that process variations will result in bounded variations about the nominal, i.e., these variations will not send the phase of any clock element beyond its valid range.

The clock router handles intentional skew by initializing the (delay, capacitance) pair of sinks to (prescribed delay, input capacitance) instead of (0, input capacitance).

A subtle issue arises when aggressive intentional skew insertion is required: the optimal clock skew schedule depends on the delay, which is not known until after placement and routing. This can be handled by obtaining delay estimates following placement (with place holders for buffers) and designing the clock tree for the clock skew schedule obtained using these delay estimates. We have found this to be adequate in our design flow.

Intentional skew insertion can also be used to improve the power dissipation on clock routes. Consider the example of a 16-bit shift register with the layout shown in Fig. 15. This design is susceptible to races as the paths between successive latches is a short path. Two sequentially adjacent flip-flops get their clock inputs from entirely different subtrees. The design in Fig. 16 is much less likely to fail because of races. The clock and data lines are routed in opposite directions, so that races are avoided. The wire length of the scheme in Fig. 16 is smaller, so that the clock tree is not only more process variation tolerant, but also dissipates less power. Thus, it is important to be able to handle partial orders on delays to clock tree sinks, while designing the clock distribution network. Note that neither the method means and medians [19], the algorithm in [21], nor equal path-length-based trees [32] can find the design shown in Fig. 16. It is not uncommon for

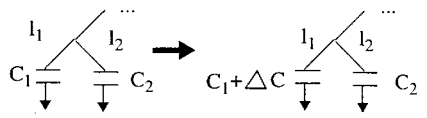

Fig. 17. Change in a load capacitance of a zero-skew tree.

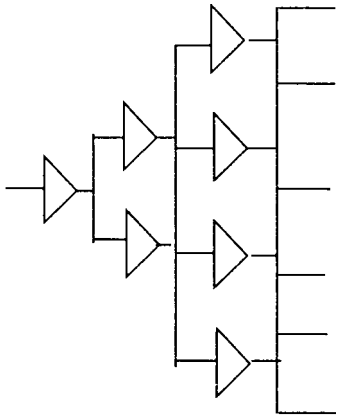

Fig. 18. Clock mesh.

such a problem to exist in designs. In fact, shift registers and scan clock trees [5] which are present in almost all current designs present exactly this problem. Current clock routers are, however, ill equipped to deal with this problem, so that manual workarounds become necessary. Our clock router obviates the need for such manual intervention.

3) Clock Gating: When clock gating is performed prior to placement (during logic synthesis, for instance), our clock router can handle it easily—each clocked regime needs to be in the same subtree. During bottom-up tree merging, we need to allow merges only between subtrees in the same clock regime.

4) Handling Engineering Change: Small design changes can be handled by reusing as much of the clock tree as possible. Consider a change ΔC in the load capacitance of a particular sink. This change could be due to the addition or removal of some latches in the design. Fig. 17 shows how the algorithm handles these changes. We wish the change to be localized to the line connecting C_1 to its parent, so that prior clock tree layout can be preserved. The downstream capacitance should be maintained, so we have $C_0 \cdot l_1 \cdot w_1 = \Delta C + C_0 \cdot l_{1n} \cdot w_{1n}$. In order to maintain zero skew, $l_{1n} \cdot w_1 \cdot (C_0 \cdot w_{1n} \cdot l_{1n} + C_1 + \Delta C) = l_1 \cdot w_{1n} \cdot (C_0 \cdot w_1 \cdot l_1 C_1)$. The two unknowns are w_{1n} and l_{1n}. There is a unique solution to these two equations. The capacitance-preserving equation gives us the new w–l product, which can be used in the second expression to get the w/l ratio, giving us the required solution.

5) Meshes Versus Trees: Clock meshes have been used for clock distribution in the past [11]. An example is shown in Fig. 18. The power dissipated in the mesh is suboptimal because it contains lines shorting equidelay points, and these lines can be removed without affecting the skew. The only reason why a mesh may be superior is the insensitivity to process variations and the ease of handling engineering changes. Our clock tree design method reduces the importance of both of these potential advantages. The high-frequency characteristics of meshes are poor because of the presence of many more reflection sources. Besides, the transmission coefficient at a branch point with three downstream lines is 0.5, while it is 0.67 for a branch point with two downstream lines [25], so the voltage wave that travels toward the leaves is smaller in a mesh.

TABLE I
1 μm CMOS CLOCK-ROUTING RESULTS

Benchmark	#sinks	Power(mW)		Buffer area		# buffers	
		GRIN	ROOT	GRIN	ROOT	GRIN	ROOT
R1	267	42	77	8	32	6	4
R2	598	88	173	14	86	5	5
R3	862	103	198	15	86	6	5
R4	1903	220	462	39	235	11	6
R5	3101	276	548	23	235	14	6

VIII. CLOCK-ROUTING RESULTS

We implemented our algorithm in C on a SPARC 10. We ran two sets of experiments. The first set of experiments compared the results of our approach with buffer insertion at internal nodes (Grin) with greedy clustering and buffer insertion only at the root (Root) using the techniques of [23], [20] popularized in [10]. This experiment verifies the use of internal buffering on the de facto clock-routing benchmark instances from [28] for three CMOS technologies. The second set of experiments explores buffer area—wire length tradeoffs for the benchmarks.

In order to demonstrate the improvement possible using buffering, we compare our algorithm with buffers at internal nodes to one with buffers only at the root. The topologies are designed by the same greedy algorithm. The buffer area and power dissipation of the corresponding trees designed using the OBTS algorithm [8] were not available [9] and could not be compared. We ran the two algorithms on the clock-routing benchmarks $r1$–$r4$ [28]. The parameters used correspond to 1, 0.8, and 0.5 μm CMOS technologies. The results are shown in Tables I–III. Note that we have used the classical cascaded drivers with exponentially increasing sizes [10], [1] at the root of both trees. Buffers at internal nodes in the tree are of a single size only. The buffer area unit is the active area of a minimum size driver. Note that the number of buffers inserted grows only linearly with the number of clock pins. The tree with buffers only at the root has to drive a large RC load, and the buffer sizes required are tremendous. By spreading the buffers over the entire tree, especially in the branches close to the root, much smaller buffer area will be required.

The power is calculated for a frequency of 200 MHz. Note that this frequency of operation cannot be achieved by the unbuffered clock tree for some instances as the rise time is too large. We see that the power and rise times are considerably smaller for the distributed buffered clock tree. The results get better as technology scales down. This is not entirely unexpected—the buffer RC time constant to interconnect RC time constant ratio decreases as technology

TABLE II
0.8 μm CMOS RESULTS

Benchmark	Power (mW)		Buffer area		# buffers	
	GRIN	ROOT	GRIN	ROOT	GRIN	ROOT
R1	25	49	4	32	4	4
R2	66	114	33	86	5	5
R3	62	128	14	86	5	5
R4	119	304	17	235	8	6
R5	161	354	19	235	10	6

TABLE III
0.5 μm CMOS RESULTS

Benchmark	Power (mW)		Buffer area		# buffers	
	GRIN	ROOT	GRIN	ROOT	GRIN	ROOT
R1	8.8	19	5	32	3	4
R2	18	24	13	32	4	4
R3	21	50	13	86	4	5
R4	37	63	15	86	6	5
R5	54	139	17	235	8	6

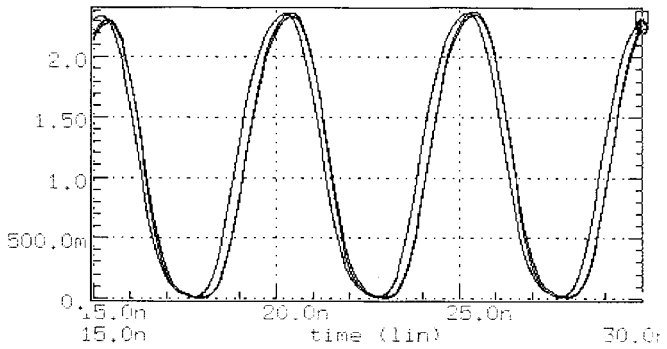

Fig. 19. HSPICE simulation results for the $R1$ clock network.

scales down. This means that using buffers becomes a more attractive proposition. As the die size increases too, we need to pay more attention to the larger examples for the finer feature size technologies.

HSPICE simulation results are shown in Fig. 19. The figure shows leaf voltage waveforms with maximum skew. The skew is clearly within limits.

The parameter α in GRIN allows us to artificially vary the "cost" of buffers. This allows us to explore buffer area–wire length tradeoffs. Fig. 20 shows us the resulting tradeoff for the instances $R1$–4 in 1 μm CMOS technology.

IX. CONCLUSIONS

We have studied the impact of scaling of technology on the clock-routing problem. Our analysis of the H tree shows that buffers *have to be* inserted at internal nodes of a clock tree. Buffer power dissipation may be quite large when the design pushes the limits of the technology, and is therefore important to consider during clock tree design. Our H-tree

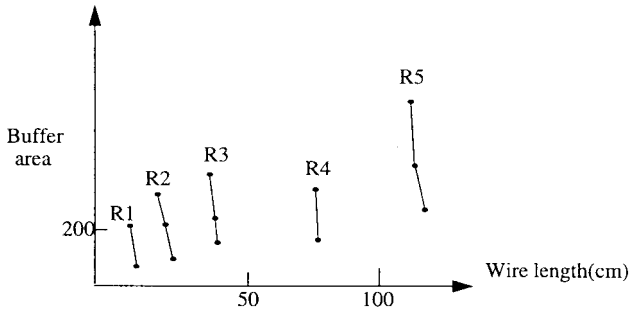

Fig. 20. Buffer area–wire length tradeoffs for 1 μm CMOS.

analysis shows that low-power clock distribution will continue to be an important problem. Limits on system size imposed by synchronicity have been explored—our analysis can predict *a priori* the maximum number of clocked elements given the clock frequency and the die size.

We have shown that buffers may be used to *optimize the wire length* of the clock tree. This degree of freedom may render H trees suboptimal by as much as 50% for regular array instances. This suboptimality is larger for some of the clock benchmarks. This implies that the *entire class* of clock-tree generation methods, which start by designing the topology and then insert buffers into the tree, could be suboptimal by 50%.

We have proposed a new problem formulation for low-power clock tree design. We have shown that the problem is NP hard, and proposed a novel heuristic to solve it. Our clock router is the only known router which *concurrently* designs the tree topology and inserts buffers. The clock router guarantees that the first moment of the impulse response at each of the sinks will be identical. Rise time constraints imposed by the design are also handled. Buffer area–wire length tradeoff is also possible.

Several practical issues related to our heuristic, including intentional skew insertion, engineering change, design flow, and clock gating, have been discussed. Our clock router handles all of these issues in a single *unified* framework.

We have also studied the *correlated* process variation model, and have shown that, when applicable, it can lead to designs which dissipate substantially less power and require much smaller layout area than when uncorrelated process variations need to be assumed.

Our clock-routing results show better power, area, and rise times than the DME algorithm [12] with buffers at the root. The rise times for our algorithm are better than the optimal buffered tree synthesis method which inserts buffers into a given topology. In addition, we have presented HSPICE results for our clock distribution algorithm, verifying small skew.

ACKNOWLEDGMENT

The authors would like to thank Dr. R.-S. Tsay of Avant! Corporation for the benchmarks and many useful comments. They also thank Prof. G. Robins and T. Zhang of the University of Virginia, Charlottesville, for the use of their 1-Steiner code.

REFERENCES

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

[2] K. D. Boese and A. B. Kahng, "Zero skew clock net routing with minimum wire length," in *Proc. IEEE Int. Conf. ASIC*, 1992, pp. 1.1.1–1.1.5.

[3] T.-H. Chao, Y.-C. Hsu, and J.-M. Ho, "Zero skew clock net routing," in *Proc. Design Automation Conf.*, 1992, pp. 518–523.

[4] C.-P. Chen, Y.-P. Chen, and D. F. Wong, "Optimal wiresizing formula under the Elmore delay model," in *ACM Phys. Design Workshop*, 1996, pp. 21–26.

[5] K.-T. Cheng and V. D. Agrawal, "A partial scan method for sequential circuits with feedback," *IEEE Trans. Comput.*, vol. 39, pp. 544–548, Apr. 1990.

[6] J.-D. Cho and M. Sarrafzadeh, "A buffer redistribution algorithm for high speed packaging," in *Proc. Design Automation Conf.*, 1993, pp. 537–542.

[7] N.-C. Chou and C.-K. Cheng, "Wirelength and delay minimization in general clock net routing," in *Dig. Tech. Papers, IEEE Int. Conf. Computer Aided Design*, 1993, pp. 552–553.

[8] J. Chung and C.-K. Cheng, "Optimal buffered clock tree synthesis," in *Proc. IEEE ASIC Conf.*, 1994.

[9] J. Chung, private communication, Aug. 1994.

[10] L. Conway and C. Mead, *Introduction to VLSI Systems.* Reading, MA: Addison-Wesley, 1980.

[11] D. W. Dobberpuhl *et al.*, "A 200 MHz, 64-bit, dual-issue CMOS microprocessor," *IEEE J. Solid-State Circuits*, vol. 27, pp. 1555–1566, Nov. 1992.

[12] M. Edahiro, "A clustering based optimization algorithm in zero skew routings," in *Proc. Design Automation Conf.*, 1993, pp. 612–616.

[13] ——, "An efficient zero skew routing algorithm," in *Proc. Design Automation Conf.*, 1994, pp. 375–380.

[14] ——, "Minimum skew and minimum path length routing in VLSI layout design," *NEC Res. Develop.*, vol. 32, no. 4, pp. 569–575, 1991.

[15] W. C. Elmore, "The transient response of damped linear networks with particular regard to wide-band amplifiers," *J. Appl. Phys.*, vol. 19, no. 1, pp. 55–63, 1948.

[16] A. L. Fisher and H. T. Kung, "Synchronizing large VLSI processor arrays," *IEEE Trans. Comput.*, vol. C-34, pp. 734–740, Aug. 1985.

[17] L. P. P. P. van Ginneken, "Buffer placement in distributed RC tree networks for minimal Elmore delay," in *Proc. Int. Symp. Circuits Syst.*, 1990, pp. 865–868.

[18] L. A. Glasser and D. W. Dobberpuhl, *The Design and Analysis of VLSI Circuits.* Reading, MA: Addison-Wesley, 1985.

[19] M. A. B. Jackson, A. Srinivasan, and E. S. Kuh, "Clock routing for high performance IC's," in *Proc. Design Automation Conf.*, 1990, pp. 573–579.

[20] R. C. Jaeger, "Comments on 'An optimized output stage for MOS integrated circuits,'" *IEEE J. Solid-State Circuits*, vol. SC-10, pp. 185–186, 1975.

[21] A. B. Kahng, J. Cong, and G. Robins, "High performance clock routing based on recursive geometric matching," in *Proc. Design Automation Conf.*, 1991, pp. 322–327.

[22] A. B. Kahng and G. Robins, "A new class of Steiner tree heuristics with good performance: The iterated 1-Steiner approach," in *Dig. Tech. Papers, IEEE Int. Conf. Computer Aided Design*, 1990, pp. 428–431.

[23] H. C. Lin and L. W. Linholm, "An optimized output stage for MOS integrated circuits," *IEEE J. Solid-State Circuits*, vol. SC-10, no. 2, pp. 106–109, 1975.

[24] D. Liu and C. Svensson, "Power consumption estimation in CMOS VLSI circuits," *IEEE J. Solid-State Circuits*, vol. 29, pp. 663–670, June 1994.

[25] S. I. Long and S. E. Butner, *Gallium Arsenide Digital IC Design.* New York: McGraw-Hill, 1990, ch. 5.

[26] S. Pullela, N. Menezes, and L. T. Pillage, "Reliable nonzero skew clock trees using wire width optimization," in *Proc. Design Automation Conf.*, 1993, pp. 165–170.

[27] J. Rubinstein, P. Penfield, and M. A. Horowitz, "Signal delay in RC tree networks," *IEEE Trans. Computer-Aided Design*, vol. CAD-2, no. 3, pp. 202–210, 1983.

[28] R.-S. Tsay, "Exact zero skew," *IEEE Trans. Computer-Aided Design*, vol. 12, pp. 242–249, 1993.

[29] A. Vittal and M. Marek-Sadowska, "Power optimal buffered clock tree design," in *Proc. Design Automation Conf.*, 1995, pp. 497–502.

[30] J. G. Xi and W. W.-M. Dai, "Buffer insertion and sizing under process variations for low power clock distribution," in *Proc. Design Automation Conf.*, 1995, pp. 491–496.

[31] J. Yuan and C. Svensson, "High-speed CMOS circuit technique," *IEEE J. Solid-State Circuits*, vol. 24, pp. 62–70, Jan. 1989.

[32] Q. Zhu and W. W.-M. Dai, "Perfect balance planar clock net routing with minimal path length," in *Dig. Tech. Papers, IEEE Int. Conf. Computer Aided Design*, 1992, pp. 473–476.

Ashok Vittal (S'93–M'93), for a photograph and biography, see p. 298 of the March 1997 issue of this TRANSACTIONS.

Malgorzata Marek-Sadowska (M'87–SM'95–F'97), for a photograph and biography, see p. 518 of the May 1997 issue of this TRANSACTIONS.