Combinatorial Algorithms for Fast Clock Mesh Optimization

Ganesh Venkataraman, *Member, IEEE*, Zhuo Feng, Jiang Hu, *Senior Member, IEEE*, and Peng Li, *Member, IEEE*

***Abstract*—Clock mesh has been widely used to distribute the clock signal across the chip. Clock mesh is driven by a top-level tree and a set of mesh buffers. We present fast and efficient combinatorial algorithms to simultaneously identify the candidate locations as well as sizes of the buffers driving the clock mesh. We show that such a sizing offers a better solution than inserting buffers of uniform size across the mesh. Due to the high redundancy, a mesh architecture offers high tolerance toward variations in clock skew. However, such a redundancy comes at the expense of mesh wire length and power dissipation. Based on survivable network theory, we formulate the problem to reduce the clock mesh by retaining only those edges that are critical to maintain redundancy. Such a formulation offers designer the option to tradeoff between power and tolerance to process variations. We present efficient postprocessing techniques to reduce the size of the mesh buffers after mesh reduction. Experimental results indicate that our techniques can result in *power savings up to 28% with less than 3.3% delay penalty*. We also present driver models that can help in simulating the clock mesh. Such models achieve near-HSPICE accuracy with significant speedup in run time.**

***Index Terms*—Circuit simulation, clock distribution, low power, variation.**

I. Introduction

CMOS process scaling has had a key contribution in helping designers meet the ever-increasing demand for faster circuits. Aggressive scaling has also lead to increasing uncertainty in design due to process variations. Clock skew, which is defined as the difference in the arrival time of the clock signal at different registers, is one such design parameter that is very sensitive to process variations [1], [2].

The function of clock distribution network (CDN) is to deliver the clock signal from the clock source to the clock sinks. The design of CDN could include multiple (and often conflicting) objectives such as wire length, power, signal slew rate, skew target, and tolerance of clock skew to variations. Tree-based distributions offer the advantage of simplicity (single path between source and sinks) as well as lower wire length [3] at the expense of skew tolerance [4], [5]. Nontree-based distributions on the other hand provide a high tolerance to variations [4]–[8].

One of the most widely used nontree-based CDN is a clock mesh [6]–[8]. Typically, clock mesh consists of a rectangular grid driven by a top-level tree. Buffers at the leaf of the top level tree shall henceforth be referred to as *mesh buffers*. Clock sinks or subnetworks are connected to the closest point in the mesh. Since there are several paths between the source and the clock sinks, there is a high level of redundancy and this redundancy translates to high tolerance to variations in clock skew. Mesh architecture is used mainly in high-performance systems [9], [13] with stringent skew requirements.

Clock mesh consumes significantly higher wire area compared to tree-based distributions. In fact, a hybrid mesh architecture could consume up to 168% higher area compared to tree [16]. Higher wire area leads to a higher load capacitance and power dissipation.

The maximum permissible delay of logic network between any two registers (i, j) is given by [2]

$$P^{ij}_{\text{delay}} = T_{\text{clock}} - t_{\text{setup}} - \text{skew}_{ij}. \tag{1}$$

In the previous equation, T_{clock} denotes the clock period, t_{setup} the set up time, and skew_{ij} denotes the skew between i and j. P^{ij}_{delay} represents the maximum permissible delay between i and j. $P_{\text{delay}} = \min_{\forall(i,j)} P^{ij}_{\text{delay}}$ *denotes the maximum permissible delay of the entire circuit*. The maximum permissible delay is a more direct metric to evaluate circuit characteristic than clock skew. It should be noted that the permissible delay is a conservative estimate on the actual delay budget. Typically, in zero skew design, 1skew_{ij} is designed to be zero. However, this may increase due to variations subsequently reducing the maximum speed at which the circuit can function. Higher skew would bring down the maximum permissible delay P_{delay}. A mesh architecture is suited well for high-performance systems since it mitigates clock skew at the expense of high resource consumption. With power occupying an increasingly important role in chip design, it is necessary to find the most desirable skew-power tradeoff. For CDN design, this could imply a higher skew for lower power dissipation. Since a significant portion of the total power consumption comes from the clock network (up to 40% of the total power dissipated [14]) and clock mesh consumes a high area/power overhead, there is a need to address more efficient ways of designing/optimizing the clock mesh. Even though there have been previous works on mesh architecture, the following issues remain largely unaddressed: 1) What are the ideal locations for the mesh buffers to drive the clock mesh? Is it ok to distribute the mesh buffers uniformly across the mesh? 2) Can the mesh buffers be sized differently? If yes,

Manuscript received October 10, 2007; revised February 22, 2008. First published April 14, 2009; current version published December 23, 2009. This work was supported in part by Semiconductor Research Corporation under Contract 2004-TJ-1205 and Contract 2006-TJ-1416.

G. Venkataraman is with Magma Design Automation, San Jose, CA 95110 USA (e-mail: ganeshv@magma-da.com).

Z. Feng, J. Hu, and P. Li are with the Department of Electrical and Computer Engineering, Texas A&M University, College Station, TX 77843-3128 USA (e-mail: fengzhuo@neo.tamu.edu; jianghu@ece.tamu.edu; pli@neo.tamu.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVLSI.2008.2007737

then does it work better than sizing uniformly? 3) A mesh has a high level of redundancy. Can some amount of redundancy be sacrificed to reduce the wire length? If yes, then how much is the tradeoff? Can we quantify the skew versus power tradeoff? In this paper, we address all the aforementioned issues. Our contributions include: the following.

1) We propose a set-cover-based algorithm for finding the mesh buffer locations and their sizes. Our algorithm works fast on a discrete library of buffer sizes. We show that such a buffer placement and sizing yield better results compared to uniform sizing.
2) We formulate the mesh reduction problem by using survivable network theory. We present heuristics for solving the formulation efficiently. Experimental results indicate up to *29% reduction in wire length, 28% reduction in power with less than 3.3% increase in delay penalty*.
3) Our techniques allow the designer to tradeoff between skew and power dissipation. In fact, the formulation presented is flexible enough to allow a high range of tradeoff (i.e., either a high-skew–low-power design or a low-skew–high-power design or anywhere in between).
4) Our algorithms run very fast. It can process test cases with over a thousand sinks within a few seconds.
5) We present an efficient gate delay model suited for clock mesh. Such a model achieves near-HSPICE accuracy with speedup up to *62X* compared with HSPICE.

II. Preliminaries and Problem Statement

We shall introduce certain notations and conventions that will be followed throughout the paper.

1) $m \times n$ denotes the dimension of the clock mesh. I denotes the set of nodes in the mesh.
2) Clock buffers of B sizes $\{b_1, b_2, \ldots, b_k\}$ in nondecreasing order. Buffer b_i can drive a load of capacitance at most c_i.
3) *Buffer mapping (BM) function*: $i \rightarrow j$ maps each node location $i \in I$ to $j \in B$. $\mathrm{BM}(i) = \phi$ implies that the location i has no buffer in it.
4) $S = \{s_1, s_2 \ldots s_c\}$ denotes the set of clock sinks. Without loss of generality, we assume that each sink s_i is connected to node $i \in I$ (in reality, the clock sinks will be connected to the closest point in the mesh that need not be in the intersection of the horizontal and vertical grid lines). The node i in the mesh is referred to as the *connection node* of sink s_i. d_{ij} denotes the minimum distance between nodes i and j in the mesh.
5) P^{ij}_{delay} denotes the maximum permissible delay of the combinational logic network between two registers i and j. $P_{\mathrm{delay}} = \min_{\forall i,j} P^{ij}_{\mathrm{delay}}$.

A. Simultaneous Mesh Buffer Placement and Sizing

Find the function BM or for each candidate buffer location (there are mn such locations), find: 1) if a buffer is required and 2) the size of buffer needed such that the following constraints are satisfied: a) each node in the mesh is allocated to at least one buffer and b) each buffer drives less than the maximum load it can drive. The objective is to minimize the total buffer size.

B. Mesh Reduction

Remove edges from the mesh such that: 1) each sink s_i has at least k node locations such that for each such node location $j, d_{ij} \leq L_{\max}, \mathrm{BM}(j) \neq \phi$ and there exists at least l edge disjoint paths between j and i and 2) the number of edges removed is maximized. $k, L_{\max}$, and l are user-defined constants.

The mesh reduction problem attempts to remove edges such that there exists at least certain number of buffers that connect each clock sink with short paths. Note that $\mathrm{BM}(j)$ can be ϕ for some nodes. However, for each sink node, the number of nodes with $\mathrm{BM}(j) \neq \phi$ at a distance of at most $L_{\max}$ should be at least k. In a mesh, the load driven by each buffer is computed using *covering regions* described in Section III. The user-defined parameters control impact the solution in the following manner: setting k and l high would mean more redundancy and hence more tolerance to variations but less number of edges removed or more power dissipation. By varying the parameters k and l, the designer has the flexibility to trade variation tolerance to power dissipation. The restriction $L_{\max}$ helps in restricting the delay between the mesh buffers and the clock sinks. This, in turn, helps in keeping the skew low.

The buffers are sized with the assumption that they are driving a complete mesh. Hence, the constraints in mesh reduction will be satisfied in mesh buffer sizing as well. After mesh reduction, the buffers could be driving a load that is significantly lesser than that of a complete mesh. Hence, as a postprocessing step, we compute the new load and downsize the buffers accordingly. This procedure will be detailed in Section IV.

III. Simultaneous Mesh Buffer Placement and Sizing via Set Cover

A. Algorithm Description

The *Set Cover* problem can be stated as follows: given a set universe $\mathcal{U}$ and a collection $\mathcal{S}$ of subsets of $\mathcal{U}$, find a minimum size subset $\mathcal{C} \subset \mathcal{S}$ such that $\mathcal{C}$ covers $\mathcal{U}$. The previous definition can be modified to weighted set cover problem by assigning weights of each set in $\mathcal{S}$. In this section, we shall show that the mesh buffer placement/sizing problem can be formulated as an instance of the set cover problem.

For each node in the mesh, define a covering region as follows.

Definition—Covering region: The covering region of the node for a particular buffer is defined as the set of nodes around the node in the 2-D mesh such that the total capacitance of the nodes included in the covering region (including the mesh capacitance as well as the nodes that the mesh drives) is less than the maximum capacitance that the buffer can drive.

Fig. 1 depicts the covering region of a buffer placed at a point in a 2-D mesh grid. The dashed lines indicate the regions of the mesh that can be driven by the buffer without violating the maximum load constraint of the buffer. If any more edges in the mesh are added, then the capacitance of the region will be greater than the maximum load that the buffer can drive.

Let CR^j_i denote the covering region of node $i \in I$ while driven by buffer $j \in B$, i.e., $\mathrm{CR}^j_i \subset I$ for each $i \in I$ and $j \in B$. Let SCR denote the super set of covering regions. We

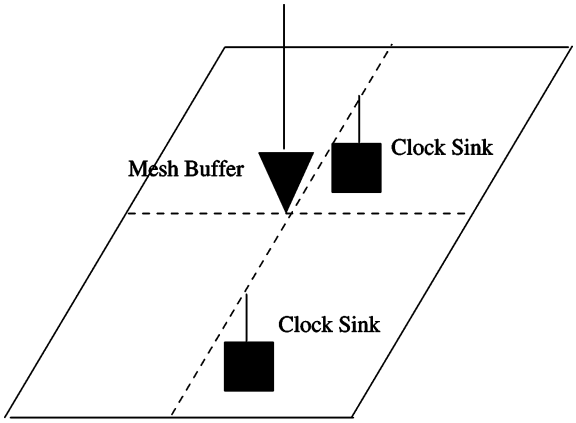


Fig. 1. Example of covering region.

can draw the parallels between the previously defined variables and the instance of set cover defined earlier.

1) The set of I of node locations can be considered as the universe $\mathcal{U}$.
2) The covering regions CR_i^j form the collection of subsets $\mathcal{S}$.
3) If the buffer size b_j denotes the weight of a subset CR_i^j, then the objective is to "pick" minimum weighted sum of subsets such that each node has at least one subset covering it.

In other words, the mesh buffering problem is identical to the set cover problem with $I \Leftrightarrow \mathcal{U}$ and $\mathrm{SCR} \Leftrightarrow \mathcal{C}$. *If* CR_i^j *is picked, then node* $i \in I$ *is driven by buffer* $j \in B$.

To motivate the solution approach, we shall now state the same problem in mathematical terms using indicator variables. Let x_i^j denote the indicator variable that is set to 1 if CR_i^j is picked in the solution. Then, the problem can be stated as

$$\text{minimize} \sum_{j \in B} \sum_{i \in I} b_j x_i^j \cup_{(i,j):x_i^j=1} \mathrm{CR}_i^j \supseteq I. \qquad (2)$$

Note that irrespective of the approach toward solving the set cover problem, it is possible that the algorithm may return two buffers for the same location, which is not a feasible solution. However, such a situation can be easily avoided by using the observation and lemma stated next.

Observation 1: For any node $i \in S, \mathrm{CR}_i^j \supseteq \mathrm{CR}_i^l$ if $b_j > b_l$. The observation comes from the fact that a bigger buffer size can drive a bigger load.

Lemma 1: In any optimal solution Φ, for any node i, there can be at most one buffer j such that $x_i^j = 1$.

Proof: Direct consequence of *Observation 1*. If there exists two buffers j and l such that $x_i^j = 1$ and $x_i^l = 1$ and $b_j \geq b_l$, then CR_i^l can be removed from Φ without loss of feasibility. This implies that Φ is not optimal, and hence, a contradiction.

Corollary 1: In any solution Φ, for any node i, if there are more than one buffer driving a node, one can pick the biggest buffer without losing feasibility.

At the end of algorithm, the solution is pruned using Corollary 1. Although there are several ways to implement the set cover algorithm, we shall implement it using the greedy algorithm. The greedy algorithm can be stated as follows: *Pick the set that covers the most nodes and then throw away the nodes that are covered. Repeat this process until all nodes are covered*. The algorithm is very fast in practice (within few seconds a test case with more than 1700 sinks) and produces good results.

Greedy set cover for mesh buffer placement/sizing.
Input : $SCR = \cup\, CR_i^j$ for each $i \in I$ and $j \in B$
Output : M = set of covering regions that are picked
1. $M \leftarrow \phi$ 2. While M does not cover I do 2.1 For each unpicked covering region CR_i^j define $C_{eff} = \frac{b_j}{\lvert CR_i^j - M \rvert}$ 2.2 Pick set C with least C_{eff}. 2.3 $M \leftarrow M \cup C$ 3. For each node $i \in I$, if there exists $j \in B$ and $l \in B$ both CR_i^j and CR_i^l are picked and $b_j > b_l$ drop CR_i^l from the solution.

Fig. 2. Greedy set cover for mesh buffer placement/sizing.

B. Complexity Analysis and Near-Continuous Sizing

Let α denote the minimum size of a subset. In other words, α denotes the minimum number of nodes in the mesh covered by the minimum buffer size in the library. Let N denote the number of nodes in the mesh and β the size of the buffer library. Since each iteration of the algorithm (in Fig. 2) covers at least α nodes, it is easy to see that the complexity of the algorithm is $O((N\beta)/(\alpha))$. Essentially, the algorithm is linear with the mesh size and number of buffer types. Fig. 3 shows the plot of CPU time verses the number of buffer types for the test case s35932. The figure illustrates the following aspects of our buffer placement/sizing algorithm.

1) The run time increases linearly with number of buffer types.
2) We ran the experiment with 41 buffer types. The minimum and maximum buffer sizes were set to 20× and 60× that of a minimum-size inverter. In other words, we had a buffer size granularity equal to the size of a minimum inverter. Our setup is equivalent to *near-continuous sizing*. Even in such a scenario, the run time of our algorithm (as indicated in Fig. 3) was about half a minute. Hence, we can conclude that our algorithm is expandable for large library sizes as well as continuous sizing—if necessary.

IV. Mesh Reduction and Postprocessing

A. Mesh Reduction

Once we locate the position of the mesh buffers and their sizes (from the buffer library), the next task is to reduce the size of the mesh. This is done by removing edges such that a certain level of redundancy is still maintained. The exact definition of the problem was stated in Section II.

We propose to solve the mesh reduction problem using similar solutions used in the design of robust communication networks. The communication networks are prone to frequent failures. Survivability makes the network functional even in the

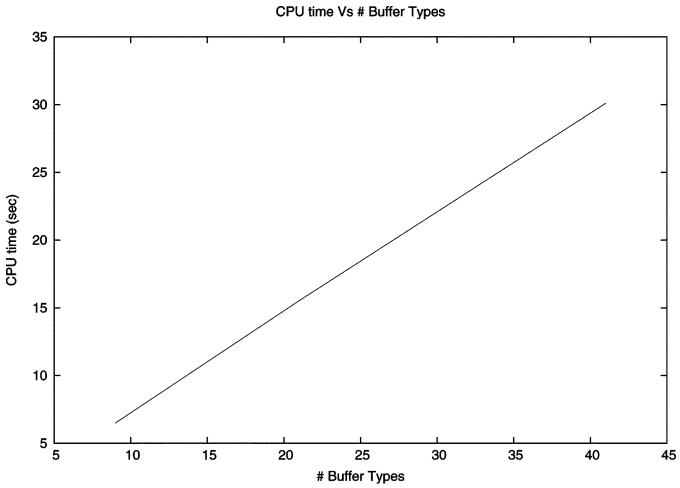


Fig. 3. CPU time versus # buffer types for s35932.

presence of link failures. This is often done by creating redundant paths that are edge disjoint (thereby increasing the chances of at least one path being active in the presence of failures). This concept has striking parallels to the clock mesh that is designed with redundancy to account for tolerance to variations. The *Steiner network problem* and its variants have been used in the design of survivable networks. In its generalized form, the Steiner network problem can be stated as follows.

Given: 1) graph $G = (V, E)$, 2) a cost function c for the edges, and 3) a connectivity requirement function $r : V^2 \rightarrow Z^+$, find a minimum cost subgraph in G such that there exists at least $r(u, v)$ edge disjoint paths for every ordered pair $u, v \in V$.

(Z^+ denotes the set of positive integers). Note that one may set $r(u, v) = 0$ for those vertex pairs that do not have a connectivity requirement. One could also set an upper bound on the number of times an edge is used in the network. The Steiner network problem generalizes the Steiner forest problem, which, in turn, generalizes the Steiner tree problem. Hence, it is NP-hard. Interested reader may refer to [17], [19], and [20] for details about the Steiner network problem and survivable networks. We shall abstract the problem of mesh reduction into Steiner network problem.

Three parameters k, l, and $L_{\max}$ define an instance of the mesh reduction problem (refer to Section II for definition of the parameters). For the sake of simplicity, we shall first assume that there is no constraint on $L_{\max}$ or path length. We shall later show that the constraint is taken care of implicitly in our formulation. We transform the mesh reduction problem into Steiner network by the following procedure.

1) Let the mesh be represented by a graph $G = (V, E)$.
2) Set connectivity requirement function $r(u, v) = 0$ for all $u, v \in V$.
3) For each clock sink $s_i \in S$, identify k closest mesh buffer locations (say) $T_i = (t_1, t_2, \ldots, t_k)$.
4) Set $r(i, j) = l$ for all $s_i \in S$ and T_i.

Now, one may use any Steiner network optimization algorithm (like [19]) on the previous instance. Since we identify the k closest buffers in the connectivity requirement, the short path constraint (by means of $L_{\max}$) is implicitly taken care of. For example, if k equals 2 and $L_{\max}$ equals 4, then assuming that a feasible solution exists, the two closest buffers should be at the distance of 4 or lesser (else relax the constraints on $L_{\max}$). This is due the fact that if these closest buffers do not satisfy the $L_{\max}$ requirement, it is easy to see that there exist no other buffer locations that can satisfy the constraint, and $L_{\max}$ requirement should be relaxed. Further, because of the connectivity requirement, edges in the shortest pairs will be retained. To solve the Steiner network problem, we use a simple greedy heuristic. Other complicated approaches like LP-rounding [19] or path length constrained network approaches [21] can also be used. But we found that the one detailed earlier produces good results with a very low run time.

Greedy Steiner Network for Mesh Reduction
Input : $G = (V, E)$, and connectivity requirements
Output : $E' \subset E$ satisfies connectivity requirements
1. For each $e \in E$, set $c(e) = 1$. 2. Initialize $E' \leftarrow \phi$ 3. For each sink $s_i \in S$ 3.1 Find k closest buffers locations 3.2 Identify l minimum cost disjoint paths (denoted by P_i) between s_i and identified buffer locations 3.3 For each $e \in P_i$, 3.3.1 $E' \leftarrow E' \cup e$, $c(e) = 0$. 4. Output E'.

Fig. 4. Greedy mesh reduction.

Post processing of buffer sizes after mesh reduction
Input : Mesh buffer sizes and reduced mesh
Output : New buffer sizes for the reduces mesh
1. For each $i \in I$ (set of buffer nodes) with $b_i > 0$ 1.1 Find minimum buffer size (say b_k) that can drive $cap_r(CR_i^j)$. 1.2 If $b_k < b_i$, $b_i \leftarrow b_k$

Fig. 5. Postprocessing after mesh reduction.

An overview of our algorithm for Steiner network minimization is shown in Fig. 4. The algorithm starts with initializing the cost of all edges to unity. This is followed by identifying edge disjoint paths between clock sinks and the closest k mesh buffers. It is worthy to mention three points about identifying these paths: 1) the disjoint path requirement is between a clock sink and a *particular* mesh buffer and not across all the k assigned buffers. For example, if a sink a is assigned to buffers at locations b and c, then we need to identify l disjoint paths between a to b (say P_{ab}) and a to c (say P_{ac}). While the paths within P_{ab} and P_{ac} are edge disjoint, they are allowed to share edges across each other; 2) since it is cost driven, the cost of an added edge is set to zero and the algorithm tries to maximize the usage of edges that improves the quality of the solution; and (c) since the mesh graph has a very regular structure (planar grid), it is easy to identify the paths. Experimental results indicate that our algorithm runs *very fast* achieving run time within *few seconds* even for a test case with more than 1700 sinks.

B. Postprocessing for Mesh Reduction

The buffers are placed and sized with the assumption that they are driving a complete mesh. However, this may not be the case

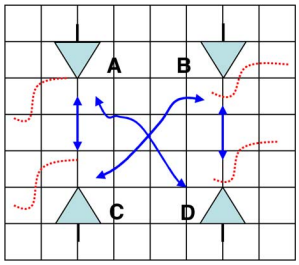


Fig. 6. Interaction between multiple mesh drivers.

after mesh reduction. Hence, there is a potential to reduce the size of the buffers after mesh reduction. In postprocessing, we find the new capacitance of the covering regions and undersize the buffers if necessary. Let R denote the set of edges that were removed from the mesh, $\mathrm{cap}(e)$ denote the capacitance of edge e, and $\mathrm{cap}(\mathrm{CR}_i^j)$ denote the capacitance of a covering region. This capacitance includes capacitance of the mesh edges as well as the input capacitance of the clock sinks, and $\mathrm{cap}_r(\mathrm{CR}_i^j)$ denote the capacitance of the covering region in the *reduced mesh*, which is computed using

$$\mathrm{cap}_r\left(\mathrm{CR}_i^j\right) = \mathrm{cap}\left(\mathrm{CR}_i^j\right) - \sum_{e \in R, e \in \mathrm{CR}_i^j} \mathrm{cap}(e). \quad (3)$$

Let b_i denote the size of the buffer at node i.

Fig. 5 gives the description of the postprocessing algorithm. We first compute the capacitance of the covering regions after accounting for the deleted edges. Each buffer in the library is characterized by the maximum capacitance that it can drive. If a certain node in a mesh has a buffer assigned to it, we then assign the minimum size buffer in the library than can drive the capacitance of the covering region—after accounting for the removed edges.

V. Driver Modeling

For clock meshes and nontree clocks, nonlinear driver modeling presents new modeling challenges. On one hand, due to the fact that a large number of clock drivers may be employed in clock meshes, efficient driver modeling is essential for increasing the analysis efficiency. Simulating large clock meshes at the transistor level is prohibitively expensive. Adopting transistor-level analysis during the optimization iterations will be very difficult, if not impossible. To this end, it is critical to develop efficient and yet accurate driver models to reduce the complexity of simulation and control the overall cost of optimization and verification.

To see the modeling issues brought by interactions between multiple drivers in a mesh, in Fig. 6, a mesh structure with four mesh drivers is shown. Since the four drivers are driving the same mesh, each of them interacts with others at the output node via the mesh network. Under this multidriver context, it is not only the case that the clock signal at any node of the mesh is fed by more than one driver, what is also true is that for each nonlinear driver the load it drives does not appear to be passive anymore. In fact, all the drivers interact with each other in a complex and nonlinear fashion. Since the input signals of these mesh drivers may differ in terms of arrival time and slew, driver output signal waveforms may be fairly complex due to the nonlinear coupling between all the drivers. Most traditional driver models are characterized under certain passive/capacitive output loading conditions, and therefore not applicable for handling the nonlinear signal interactions in clock mesh.

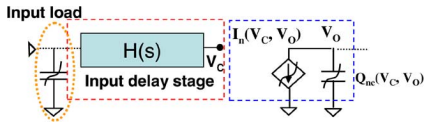


Fig. 7. Proposed clock driver model.

TABLE I
Comparison of Driver Model With HSPICE

Case	HSPICE CPU (sec)	Driver Model CPU (sec)	Speed Up	Max Error(%)	Avg Error (%)
s9234	10.8	0.2	47.1	4.3	1.5
s5378	3.9	0.4	9.5	2.8	1.4
s13209	145.6	4.6	31.7	7.2	2.0
s15850	84.8	4.9	17.5	4.5	1.3
s38584	590.9	12.8	46.4	3.6	1.0
s35932	934.4	15.1	62.0	5.9	1.5
Ave	295.1	6.3	35.7	4.7	1.5

To accurately capture the complex signal interactions in multidriver clock meshes, it is evident that robust driver models that are accurate even for nondigital, highly complex and nonmonotonic input/output signals are desired. In this paper, we achieve this goal by adopting driver models that are characterized in a waveform independent fashion. For a nonlinear driver, possibly consisting of multiple stages, we model it using the compact driver model shown in Fig. 7. The driver model we employed in this paper is based on an extension of the work presented in [26].

As shown in Fig. 7, the driver model consists of three basic components. First, either a linear or a nonlinear input capacitance is included at the input pin to model the input loading effect. Since the input capacitance is primarily contributed by the gate capacitances of the transistors at the first stage of the driver, it is parametrized in transistor channel lengths in order to account for its variability. The next component of the model consists of a linear transfer function block $H(s)$ that is used to model the signal transfer from the input pin to an internal controlling node voltage V_c. $H(s)$ is specified by two pole/residue pairs. Essentially, it is precharacterized to capture the "intrinsic" internal delay of the driver, especially for multistage drivers. The last component of the model characterizes the output stage of the driver. It consists of a voltage-controlled current source and a nonlinear output capacitance both of which are specified using lookup tables (LUTs). The current LUT is a 2-D table indexed by two voltages: V_c and V_o. It models the nonlinear dc current driving capability of the driver under various V_c and V_o combinations. The LUT for the nonlinear output capacitance is in form of a 2-D charge table indexed by the same two voltages. It is employed to model the nonlinear capacitive parasitics effects at the output. Since the complete gate model is characterized without making any assumption on the input/output

TABLE II
RESULTS FOR THE BUFFER PLACEMENT/SIZING ALGORITHM

Case	Area	WL (μm)	Pow (mW)	$skew_{nom}$ (ps)	μ_{skew} (ps)	σ_{skew} (ps)	$skew_{max}$ (ps)	SV (%)	P_{delay} (ps)	CPU (sec.)
s9234	1140	30366	7.8	33.0	37.2	6.3	56.1	6.4	943.9	0.1
s5378	1350	32290	8.4	29.1	44.0	5.5	60.1	0.0	939.5	0.1
s13207	4870	153450	31.3	22.9	46.9	17.0	97.8	0.0	902.2	0.7
s15850	5370	164670	34.2	21.8	31.4	10.0	61.4	0.0	938.6	0.7
s38584	11410	371900	80.2	33.0	66.1	14.8	110.6	1.0	889.4	4.3
s35932	12660	427900	94.9	36.2	65.4	11.6	100.0	7.7	900.0	4.7
Ave	6135	196763	42.8	29.3	48.5	10.9	81.1	2.5	918.9	1.8

TABLE III
RESULTS FOR UNIFORM SIZING—SMALLEST SIZE

Case	Smallest Buffer										
	$skew_{nom}$	μ_{skew}	σ_{skew}	$skew_{max}$	SV (%)	Area	Ratio	Pow	Ratio	P_{delay}	Ratio
s9234	31.5	29.6	7.8	53.0	25.00	980	0.86	7.9	1.01	947.0	1.00
s5378	19.6	23.6	3.2	33.2	17.20	999	0.74	8.1	0.96	966.8	1.03
s13207	16.3	34.1	9.9	63.8	10.55	3847	0.79	31.4	1.00	936.2	1.04
s15850	16.4	32.9	12.5	70.3	18.54	3866	0.72	33.9	0.99	929.7	0.99
s38584	36.0	63.8	15.1	109.0	47.49	7531	0.66	78.8	0.98	891.0	1.00
s35932	50.7	69.4	10.3	100.4	77.05	7723	0.61	92.1	0.97	899.6	1.00
Ave	28.4	42.2	9.8	71.6	32.64	4158	0.73	42.03	0.98	928.4	1.01

TABLE IV
RESULTS FOR UNIFORM SIZING—MEDIUM BUFFER SIZE

Case	Medium Buffer										
	$skew_{nom}$	μ_{skew}	σ_{skew}	$skew_{max}$	SV (%)	Area	Ratio	Pow	Ratio	P_{delay}	Ratio
s9234	25.6	25.3	7.0	46.2	0.00	1642	1.44	8.4	1.08	953.8	1.01
s5378	21.5	23.0	3.4	33.2	0.00	1661	1.23	8.7	1.04	966.8	1.03
s13207	14.6	35.6	9.9	65.2	0.00	6380	1.31	32.7	1.04	934.8	1.04
s15850	14.9	40.7	13.6	81.4	0.00	6390	1.19	35.2	1.03	918.6	0.98
s38584	28.3	61.3	14.8	105.7	0.00	12551	1.10	81.5	1.02	894.4	1.01
s35932	39.3	68.7	13.6	109.3	13.23	12787	1.01	95.3	1.00	890.7	0.99
Ave	24.0	42.4	10.4	73.5	2.21	7442	1.21	43.63	1.03	926.5	1.01

TABLE V
RESULTS FOR UNIFORM SIZING—LARGEST BUFFER SIZE

Case	Largest Buffer										
	$skew_{nom}$	μ_{skew}	σ_{skew}	$skew_{max}$	SV (%)	Area	Ratio	Pow	Ratio	P_{delay}	Ratio
s9234	23.2	24.8	5.9	42.6	0.00	1972	1.73	8.8	1.13	957.5	1.01
s5378	17.7	21.2	3.2	30.7	0.00	1998	1.48	9.1	1.08	969.3	1.03
s13207	13.0	38.6	4.2	51.2	0.00	7695	1.58	34.1	1.09	948.8	1.05
s15850	11.9	44.8	14.6	88.7	0.00	7679	1.43	36.6	1.07	911.3	0.97
s38584	24.7	62.7	14.8	107.0	0.00	15175	1.33	75.3	0.94	893.0	1.00
s35932	34.9	69.4	13.7	110.6	0.00	15319	1.21	91.1	0.96	889.4	0.99
Ave	20.9	43.6	9.4	71.8	0.00	8955	1.46	42.50	1.05	928.2	1.01

signal waveforms, it models the intrinsic nonlinear dynamic behavior of the driver. As a result, it can provide near-HSPICE accuracy even when the signal shapes deviate significantly from ramp-like shapes. In our experiments, we have observed that the proposed modeling technique and the associated simulation can achieve up to *62X* run time speedups compared with HSPICE without any significant sacrifice of accuracy for clock mesh simulation.

VI. EXPERIMENTAL RESULTS

A. Experiment Setup

The algorithms were implemented in C++ and simulations were run on a Linux Work Station with 2 GB RAM. The proposed techniques were verified using experiments performed on the ISCAS89 benchmark circuits. Interested reader may refer to [15] for benchmark characteristics as well as the computer-aided design (CAD) flow used to synthesize them. All driver model results were compared with HSPICE using 65-nm process model cards from Berkeley Predictive Technology Model (BPTM).[1] The per unit resistance and capacitance used were 0.391 Ω/μm and 0.155 fF$/\mu$m, respectively [28]. We use Pi-model to model the interconnect. The following notations will be used in the tables presented: *WL* denotes the wire length (in micrometers), ($\mathrm{skew_{nom}}$) denotes the nominal skew (measured when parameters are set to ideal values), and $\mu_{\mathrm{skew}}(\sigma_{\mathrm{skew}})$ denotes the mean (standard deviation) of the skew due to variations. $\mathrm{skew_{max}}$ equals $\mu_{\mathrm{skew}} + 3\sigma_{\mathrm{skew}}$. All skew measurements are presented in picoseconds. *Pow* denotes power dissipation measured in milliwatts. The power dissipation is measured using HSPICE simulations that measure the current drawn by the devices in an entire clock cycle and computing the area under the voltage versus current curve. Hence, this power includes both dynamic leakage and crowbar

[1][Online]. Available: http://www.eas.asu.edu/ptm/

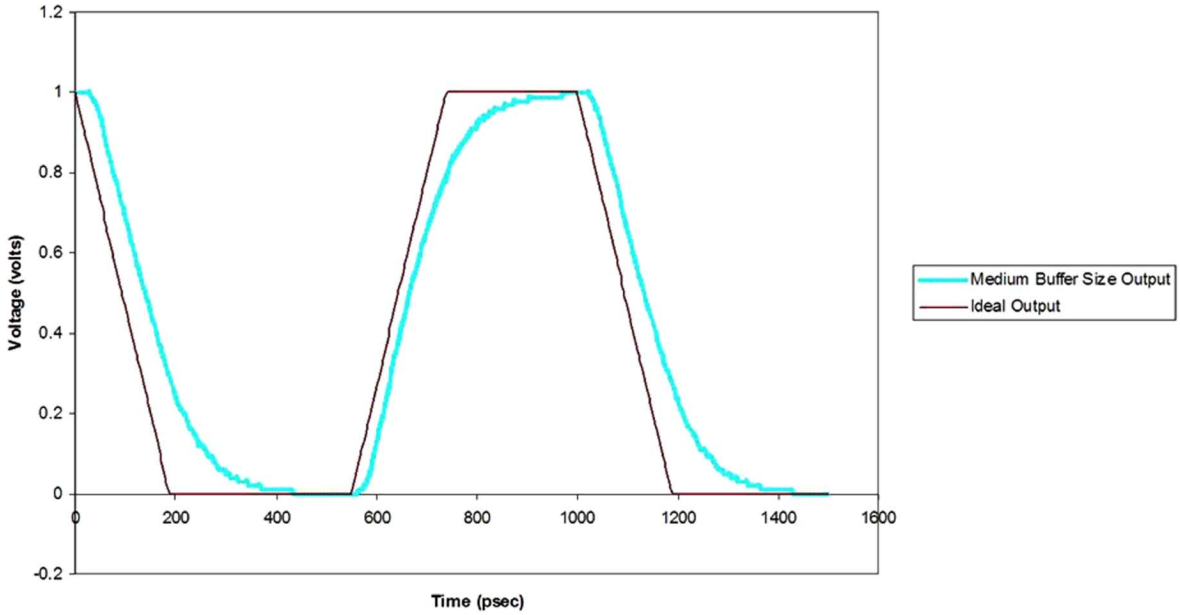


Fig. 8. Worst-case output waveform with medium buffer size.

power dissipation. Slew violation (*SV*) is the maximum positive deviation at all the clock sink locations from the user-specified value, i.e., if slew_r denotes the required slew and $\text{slew}_{\max}$ denotes the maximum slew among the sink nodes, then

$$\text{SV} = 100 \times \frac{(\text{slew}_{\max} - \text{slew}_r)}{\text{slew}_r}. \tag{4}$$

If the slew violation is negative, then it is set to zero. The slew_r is set to 150 ps. P_{delay} measures the maximum permissible delay as defined by (1). P_{delay} measurements are presented in picoseconds.

B. Experiment Design

In this paper, we propose the following techniques: 1) simultaneous mesh buffer placement/sizing; 2) mesh reduction; and 3) driver model for fast simulation of clock mesh. In order to measure the effectiveness of the aforementioned techniques, we conducted the following experiments.

1) Compare our mesh buffer placement/sizing algorithm with uniform sizing. By uniform sizing, we mean placing buffer of single size uniformly distributed throughout the mesh. Buffers are added at symmetric locations. We do the experiments for small, medium, and large buffer sizes from the library.
2) Compare our mesh reduction with that of complete mesh.
3) Compare the run time results for our driver model with HSPICE.

For the first two sets of experiments mentioned before, we make comparisons in terms of wire length, power dissipation, nominal skew, and variation skew. For measuring the impact of variation on clock skew, we constructed a zero-skew buffered clock tree to drive the clock mesh. Clock tree construction follows the techniques presented in [5].

For Monte Carlo simulations, the following parameters were varied: 1) buffer channel length; 2) interconnect wire width; 3) VDD; and 4) sink load capacitance. The previous parameters are varied with mean as the nominal value and the σ value set to 5% of the nominal value. This implies that the worst-case value (generally equated to 3σ) is 15%. Spatial correlation among all the variations was accounted by using the principal component analysis [29]. The Monte Carlo simulations were done using HSPICE.

C. Results

Table I compares the results of the driver model with HSPICE. The second and third columns indicate the CPU time while running HSPICE and the driver model, respectively. The fourth column indicates the speedup measured as a fraction of HSPICE run time to the model run time. The delay is measured at all the clock sinks using both HSPICE and the driver model. Using HSPICE delay as the base value, we then measure the maximum and average percentage error at the sinks and report it in the last two columns. It is easy to see that the model achieves a *high level of accuracy* (within 7% in all the cases) with an average speedup above 35×. In fact, the largest test case showed a speedup of *62*× with less than 6% maximum error. Such a tool is especially helpful in running statistical Monte-Carlo-based analysis (which is very accurate). Since running Monte Carlo simulations on HSPICE becomes impractical, the proposed model could be used.

The results for our simultaneous mesh buffer placement/sizing algorithm (henceforth referred to as *sizing algorithm*) are shown in Table II. The second, third, and fourth columns indicate the total buffer area, wire length, and power dissipation, respectively. Buffer area is reported in terms of the buffer area of a minimum-size inverter (i.e., if the buffer area is 100, then the actual buffer area is 100× the active area of a minimum-size inverter). In the next column, we report the nominal skew. This is followed by a set of three columns that denote mean, standard deviation, and maximum skew due to variations (obtained by running 1000 Monte Carlo

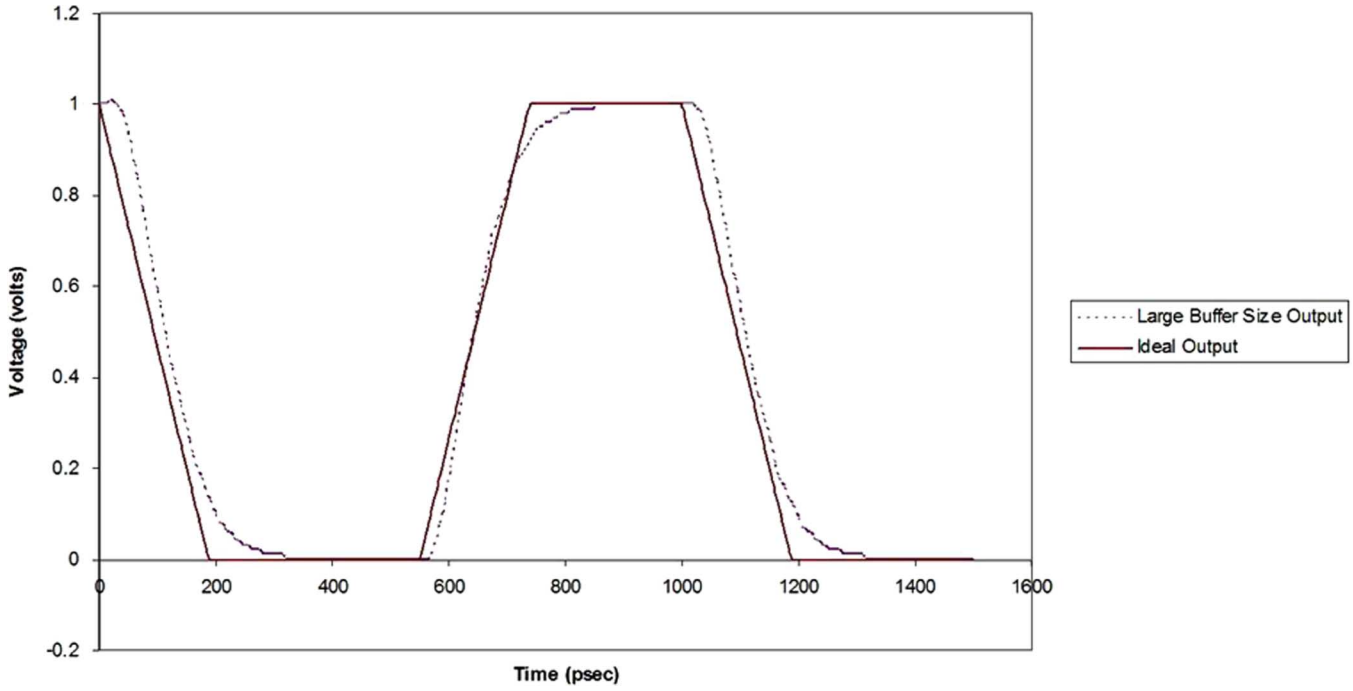


Fig. 9. Worst-case output waveform with large buffer size.

simulations). The last three columns denote the slew violation [computed using (4)], maximum permissible delay, and CPU time, respectively. P_{delay} is computed by subtracting the clock period (assuming 1 GHz clock) with $\text{skew}_{\max}$. The following conclusions can be drawn from Table II.

1) Our sizing algorithm meets the slew specifications (within *2.5%* on an average).
2) The run time of our algorithm is within a few seconds and is therefore largely inconsequential.

We compare our approach with the one where identical buffers were placed at symmetric nodes in the mesh. Tables III–V present the nominal, variation skew area, power, and P_{delay} values when the smallest, medium size, and largest buffer sizes are used to drive the mesh at symmetric nodes, respectively. The tables also provide the ratio comparison (of area, power, and P_{delay}) between uniform sizing and our sizing algorithm (results in Table II). It is worth noting that the skew results (both nominal and variation) from our sizing algorithm are not very different (within 15 ps) from the ones obtained by using uniform sizing. The following inferences can be made.

1) Using the smallest buffer uniformly results in an area reduction of 37% compared to our sizing algorithm. However, it also leads to 33% slew violation on an average and up to 77% in some cases. P_{delay} values are identical for almost all the approaches.
2) Medium buffer sizes satisfy slew constraints in all but one case and large buffer size satisfies the slew constraints in all the cases. But, this comes at an area penalty of 21% (46%) for medium (large) buffer size.

Figs. 8 and 9 gives the worst-case output signal (signal at the sink that has the worst slew) while using medium and large buffer sizes. Fig. 10 gives the corresponding plot while using our sizing algorithm. In each of the aforementioned plots, the ideal signal output (with signal slew of 150 ps) is also shown for comparison. The plots were obtained for the test case s35932. Note that using large buffer type as well as using our sizing algorithm gives a waveform that closely resembles the ideal output. However, using large buffer size will result in 46% increase in buffer area. Using medium buffer sizes gives a signal waveform that deviates significantly from the slew specification.

Next we compare the results of the mesh reduction algorithm. We compare both resource consumption and tolerance to variation. The results are indicated in Table VI. We present the wire length and power reduction when compared to the complete mesh. Table VI also presents the nominal skew, mean, standard deviation, maximum skew value, and the maximum delay (P_{delay}) obtained on running Monte Carlo simulations run with the setup described earlier. The results can be summarized as follows.

1) Mesh leads to a *wire length reduction of 25.9% and power savings of 18.5% on an average*.
2) In some test cases, the power savings can be as high as *28%*.
3) These savings do have an impact on the tolerance to variations. However, it can be seen that the delay penalty is less than 3.3% on all the cases. In fact, the delay penalty is *less than 1%* for the two largest test cases. The nominal skew results are identical to those obtained without reduction. Hence, mesh reduction preserves the nominal skew.

In order to find the effectiveness of our greedy heuristics, we implemented the LP-rounding technique [19]. The results for mesh reduction (relative to the greedy heuristic) are shown in Table VII. We found that the results are identical in most test cases with higher run time for linear programming. So, the rest of the analysis is done using the greedy algorithm.

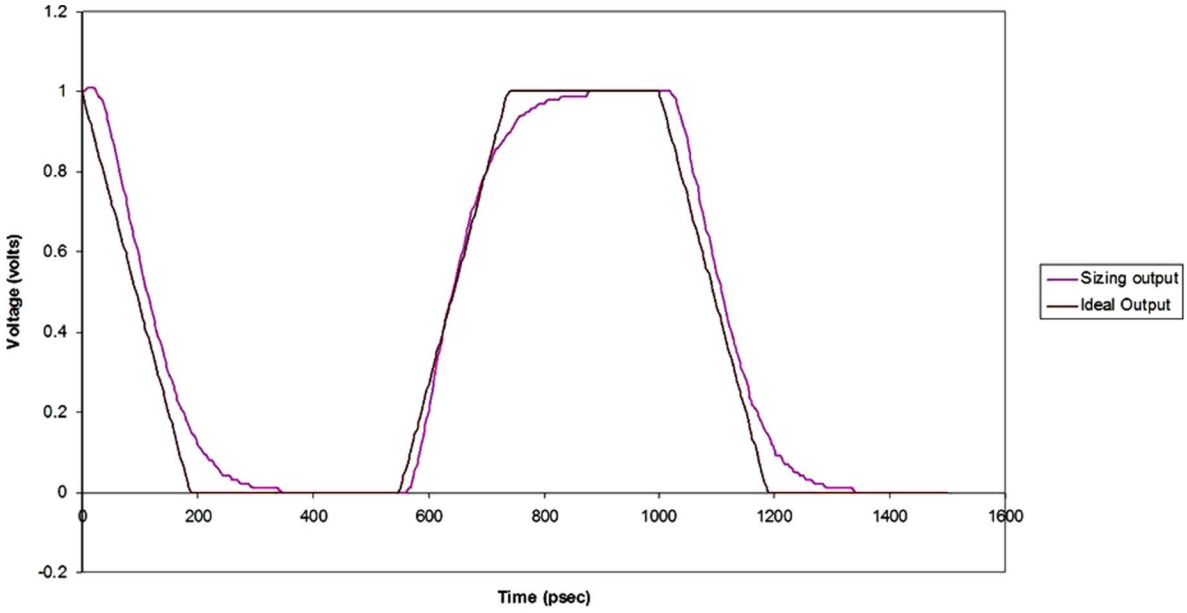


Fig. 10. Worst-case output waveform with sizing algorithm.

TABLE VI
RESULTS FOR MESH REDUCTION

Case	Wire Length		Pow		$skew_{nom}$	μ_{skew}	σ_{skew}	$skew_{max}$	P_{delay}	
	μm	% Imp.	mW	% Imp	(ps)	(ps)	(ps)	(ps)	(ps)	% Red
s9234	27177	10.5	6.7	6.1	33.0	45.4	7.8	68.7	931.3	1.3
s5378	24911	22.9	6.7	14.0	29.1	44.7	6.4	63.8	936.2	0.3
s13207	109538	28.6	23.8	21.5	22.9	60.0	22.5	127.5	872.5	3.3
s15850	100778	38.8	23.8	28.1	21.8	37.5	11.3	71.2	928.8	1.0
s38584	262528	29.4	60.9	22.0	33.0	70.5	16.0	118.5	881.5	0.9
s35932	321293	24.9	74.3	19.6	36.2	71.2	12.3	108.1	891.9	0.9
Ave	131981	25.9	32.7	18.5	29.7	54.9	12.7	93.0	907.0	1.3

TABLE VII
MESH REDUCTION USING LP-ROUNDING, RELATIVE TO GREEDY HEURISTIC

Case	Wire Length	Pow	CPU
s9234	1.000	1.000	1.20
s5378	1.000	1.000	1.20
s13207	0.992	0.995	1.80
s15850	1.000	1.000	1.60
s38584	0.994	1.000	2.10
s35932	0.995	1.000	2.50
Ave	0.997	0.999	1.73

Fig. 11 gives the power versus maximum skew tradeoff curve for test case s9234. The *Y*-axis measures the power dissipation and *X*-axis measures the $\mathrm{skew}_{\max}$ value after measuring the skew using Monte Carlo simulations. Each point in the curve is obtained by setting a different value of k. A higher value would imply more paths, more wire length, more power dissipation, lower skew, and higher P_{delay} and vice versa for lower k (k is the number of buffers each sink should be connected to with paths of short length as defined in Section II). Since our algorithm is very fast, the user can potentially run it for different values of k and pick the one that meets the design specifications. Thus, it provides a *framework for tradeoff* between skew and power dissipation.

Table VIII gives the buffer area, power, and P_{delay} improvement for the postprocessing technique detailed in Section IV-B.

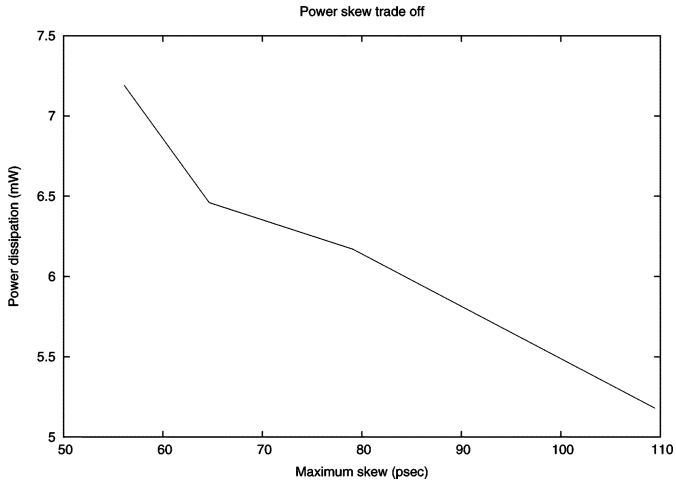


Fig. 11. Power versus $\mathrm{skew}_{\max}$ tradeoff for s9234.

The table compares the results with those shown in Table VI. The following inferences can be drawn from Table VIII. Postprocessing technique can be treated as a *fine-tuning* technique and not an optimization procedure of its own. However, it does result in *3.8%* reduction in buffer area (and could be as high *9.9%*) on an average. Such a reduction though small could be significant in high performance designs. Postprocessing does

TABLE VIII
REDUCTION IN BUFFER AREA AND POWER AFTER POSTPROCESSING

Case	Area Imp(%)	Pow Imp(%)	P_{delay} Imp(%)
s9234	3.5	0.68	0.0
s5378	0.0	0.00	0.0
s13209	2.7	0.41	0.0
s15850	9.9	2.12	0.0
s38584	4.7	0.88	0.0
s35932	1.8	0.24	0.0
Ave	3.8	0.72	0.0

not have much of an impact on power dissipation and permissible delay.

VII. CONCLUSION

In this paper, we presented two combinatorial algorithms used for fast clock mesh optimization. The first one does a simultaneous buffer placement and sizing that satisfies the signal slew constraints while minimizing the total buffer size. The second one reduces the mesh by deleting certain edges, thereby trading off skew tolerance for low power dissipation. Since our techniques are fast, it offers the flexibility to optimize clock mesh with different design objectives. We also present gate models that achieve near-SPICE accuracy with significant speedup. Our techniques indicate up to *28%* reduction in power with less than 3.3% increase in maximum permissible delay.

REFERENCES

[1] Y. Liu, S. R. Nassif, L. T. Pileggi, and A. J. Strojwas, "Impact of interconnect variations on the clock skew of a gigahertz microprocessor," in *Proc. DAC*, 2000, pp. 168–171.

[2] J. P. Fishburn, "Clock skew optimization," *IEEE Trans. Comput.*, vol. 39, no. 7, pp. 945–950, Jul. 1990.

[3] T.-H. Chao, Y.-C. Hsu, J.-M. Ho, K. D. Boese, and A. B. Kahng, "Zero skew clock routing with minimum wirelength," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 39, no. 11, pp. 799–814, Nov. 1992.

[4] A. Rajaram, J. Hu, and R. Mahapatra, "Reducing clock skew variability via cross links," in *Proc. ACM/IEEE Des. Autom. Conf.*, 2004, pp. 18–23.

[5] G. Venkataraman, N. Jayakumar, J. Hu, P. Li, S. Khatri, A. Rajaram, P. McGuinness, and C. Alpert, "Practical techniques to reduce skew and its variations in buffered clock networks," in *Proc. IEEE/ACM Int. Conf. Comput. Aided Des.*, 2005, pp. 592–596.

[6] N. A. Kurd, J. S. Barkatullah, R. O. Dizon, T. D. Fletcher, and P. D. Madland, "A multigigahertz clocking scheme for the pentium 4 microprocessor," *IEEE J. Solid-State Circuits*, vol. 36, no. 11, pp. 1647–1653, Nov. 2001.

[7] P. J. Restle, T. G. McNamara, D. A. Webber, P. J. Camporese, K. F. Eng, K. A. Jenkins, D. H. Allen, M. J. Rohn, M. P. Quaranta, D. W. Boerstler, C. J. Alpert, C. A. Carter, R. N. Bailey, J. G. Petrovick, B. L. Krauter, and B. D. McCredie, "A clock distribution network for microprocessors," *IEEE J. Solid-State Circuits*, vol. 36, no. 5, pp. 792–799, May 2001.

[8] N. Bindal, T. Kelly, N. Velastegui, and K. L. Wong, "Scalable sub-10 ps skew global clock distribution for a 90 nm multi-GHz IA microprocessor," in *Proc. IEEE Int. Solid-State Circuits Conf.*, 2003, pp. 346–355.

[9] G. Northrop, "609 MHz G5 S/399 microprocessor," in *Proc. ISSCC*, 1999, pp. 88–89.

[10] P. J. Restle, "The clock distribution of the power4 microprocessor," in *Proc. ISSCC*, 2002, pp. 144–145.

[11] R. Heald, "Implementation of a 3rd-generation SPARC V9 64 b microprocessor," in *Proc. ISSCC*, 2000, pp. 412–413.

[12] C. Yeh, "Clock distribution architectures: A comparative study," in *Proc. 7th Symp. Quality Electron. Des.*, 2006, pp. 85–91.

[13] M. G. R. Thomson, P. J. Restle, and N. K. James, "A 5 GHz duty-cycle correcting clock distribution network for the power6 microprocessor," in *Proc. Int. Conf. Solid State Circuits*, 2006, pp. 1522–1529.

[14] M. Donno, E. Macii, and L. Mazzoni, "Power-aware clock tree planning," in *Proc. ISPD*, 2004, pp. 138–147.

[15] G. Venkataraman, Z. Feng, J. Hu, and P. Li, "Combinatorial algorithms for fast clock mesh optimization," in *Proc. ICCAD*, 2006, pp. 563–567.

[16] H. Su and S. Sapatnekar, "Hybrid structured clock network construction," in *Proc. ICCAD*, 2001, pp. 333–336.

[17] V. V. Vazirani, *Approximation Algorithms*. New York: Springer-Verlag, 2001.

[18] U. Feige, "A threshold of ln n for approximating set cover," *J. ACM*, vol. 45, no. 4, pp. 634–652, 1998.

[19] K. Jain, "A factor 2 approximation algorithm for the generalized Steiner network problem," in *Proc. IEEE Symp. Found. Comput. Sci.*, 1998, pp. 448–457.

[20] H. Kerivin and A. R. Mahjoub, "Design of survivable networks: A survey," *Networks*, vol. 46, no. 1, pp. 1–21, Apr. 2005.

[21] W. B. Ameur, "Constrained length connectivity and survivable networks," *Networks*, vol. 36, no. 1, pp. 17–23, Aug. 2000.

[22] J. Qian, S. Pullela, and L. Pillage, "Modeling the 'effective capacitance' of RC interconnect," *IEEE Trans. Comput.-Aided Des.*, vol. 13, no. 12, pp. 1526–1535, Dec. 1994.

[23] F. Dartu, N. Menezes, J. Qian, and L. Pillage, "A gate-delay model for high speed CMOS circuits," in *Proc. IEEE/ACM Des. Autom. Conf.*, June 1994, pp. 576–580.

[24] R. Arunachalam, F. Dartu, and L. Pileggi, "CMOS gate delay models for general RLC loading," in *Proc. IEEE ICCD*, Oct. 1997, pp. 224–229.

[25] J. Croix and D. Wong, "Blade and razor: Cell and interconnect delay analysis using current-based models," in *Proc. IEEE/ACM Des. Autom. Conf.*, June 2006, pp. 386–389.

[26] P. Li and E. Acar, "A waveform independent gate model for accurate timing analysis," in *Proc. IEEE ICCD*, Oct. 2005, pp. 363–365.

[27] I. Keller, K. Tseng, and N. Verghese, "A robust cell-level crosstalk delay change analysis," *Proc. IEEE ICCAD*, pp. 147–154, Nov. 2004.

[28] A. B. Kahng and B. Liu, "Q-Tree: A new iterative improvement approach for buffered interconnect optimization," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI*, Feb. 2003, pp. 183–188.

[29] H. Chang and S. S. Sapatnekar, "Statistical timing analysis considering spatial correlations using a single PERT-like traversal," in *Proc. IEEE/ACM Int. Conf. Comput. Aided Des.*, 2003, pp. 621–625.

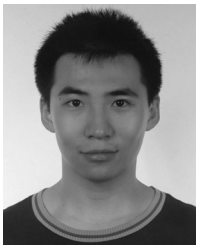
[30] E. M. Sentovich, K. J. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. R. Stephan, R. K. Brayton, and A. L. Sangiovanni-Vincentelli,, "SIS: A system for sequential circuit synthesis," Univ. California at Berkeley, Berkeley, CA, Memo. ERL M92/41, May 1992.

[31] "CPMO-constrained placement by multilevel optimization," Comput. Sci. Dept., UCLA, Los Angeles, CA, 2009. [Online]. Available: http://ballade.cs.ucla.edu/cpmo/

Ganesh Venkataraman (S'02–M'07) received the B.E.(Hons.) degree in electrical and electronics engineering and the M.Sc.(Hons.) degree in physics from Birla Institute of Technology and Science, Pilani, India, both in 2000, the M.S. degree in electrical engineering from the University of Iowa, Iowa City, in 2003, and the Ph.D. degree in computer engineering from Texas A&M, College Station, in 2007.

Between 2000 and 2001, he was a Design Engineer at Cypress Semiconductor. He is currently a Member of the Consulting Staff at Magma Design Automation, San Jose, CA, where his responsibilities include aspects of physical synthesis and optimization. His current research interests include VLSI physical design, variation tolerant clock distribution, low power, graph theory, and combinatorial optimization.

Dr. Venkataraman was a recipient of the Graduate Merit Fellowship at the Texas A&M University.



Zhuo Feng received the B.Eng. degree in information engineering from Xian Jiaotong University, Xian, China, in 2003, and the M.Eng. degree in electrical engineering from the National University of Singapore, Singapore, in 2005. He is currently working toward the Ph.D. degree in electrical and computer engineering at Texas A&M University, College Station.

His current research interests include hardware acceleration of circuit simulations, fast power gird analysis, model order reduction, variational VLSI circuit modeling/optimization, and statistical timing analysis.

Jiang Hu (SM'07) received the B.S. degree in optical engineering from Zhejiang University, Hangzhou, China, in 1990, the M.S. degree in physics and the Ph.D. degree in electrical engineering from the University of Minnesota, Minneapolis, in 1997 and 2001, respectively.

From January 2001 to June 2002, he was with International Business Machines Corporation (IBM) Microelectronics. He is currently an Associate Professor in the Department of Electrical and Computer Engineering, Texas A&M University, College Station. His current research interests include computer-aided design for VLSI circuits, especially on interconnect optimization, clock network synthesis, variation tolerance technology, and design for manufacturability.

Dr. Hu received the Best Paper Award at the Association for Computing Machinery (ACM)/IEEE Design Automation Conference in 2001 and the IBM Invention Achievement Award in 2003. He has been a member of technical program committees of the Design Automation Conference (DAC), the International Conference on Computer-Aided Design (ICCAD), the International Symposium on Physical Design (ISPD), the International Symposium on Quality Electronic Design (ISQED), the International Conference of Computer Design (ICCD), the Design Automation and Test in Europe (DATE), and the International Symposium on Circuits and Systems (ISCAS). He is currently an Associate Editor of the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN (CAD).

Peng Li (S'02–M'04) received the B.Eng. degree in information engineering and the M.Eng. degree in systems engineering from Xian Jiaotong University, Xian, China, in 1994 and 1997, respectively, and the Ph.D. degree in electrical and computer engineering from Carnegie Mellon University, Pittsburgh, PA, in 2003.

Since August 2004, he has been an Assistant Professor at Texas A&M University, College Station. His current research interests include the general area of VLSI design and computer-aided design (CAD) with an emphasis on analog/RF optimization and test, circuit simulation, parallel CAD algorithms, design and analysis of power and clock distribution networks, interconnect and timing analysis, statistical circuit analysis, and optimization.

Dr. Li received two Semiconductor Research Corporation (SRC) Inventor Recognition Awards in 2001 and 2004, two Design Automation Conference (DAC) Best Paper Awards in 2003 and 2008, the Microelectronics Advanced Research Corporation (MARCO) Inventor Recognition Award in 2006, and the National Science Foundation CAREER Award in 2008. He is an Associate Editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS II. He has been a member of the committees of a number of international conferences and workshops including the International Conference on Computer-Aided Design (ICCAD), the International Symposium on Quality Electronic Design (ISQED), the International Symposium on Circuits and Systems (ISCAS), and the TAU. He is the recipient of the Association for Computing Machinery (ACM) Outstanding Ph.D. Dissertation Award in Electronic Design Automation.