Novel Binary Linear Programming for High Performance Clock Mesh Synthesis

Minsik Cho, David Z. Pan* and Ruchir Puri
IBM T. J. Watson Research Center, Yorktown Heights, NY 10598
*Dept. of ECE, The University of Texas at Austin, Austin, TX 78712
minsikcho@us.ibm.com, dpan@ece.utexas.edu, ruchir@us.ibm.com

***Abstract*— Clock mesh is popular in high performance VLSI design because it is more robust against variations than clock tree at a cost of higher power consumption. In this paper, we propose novel techniques based on binary linear programming for clock mesh synthesis for the first time in the literature. The proposed approach can explore both regular and irregular mesh configurations, adapting to non-uniform load capacitance distribution. Our synthesis consists of two steps: mesh construction to minimize total capacitance and skew, and balanced sink assignment to improve slew/skew characteristics. We first show that mesh construction can be analytically formulated as binary polynomial programming (a class of nonlinear discrete optimization), then apply a compact linearization technique to transform into binary linear programming, significantly reducing computational overhead. Second, our balanced sink assignment enables a sink to tap the least loaded mesh segment (not the nearest one) with another binary linear programming which reduces both slew and skew. Experiments show that our techniques improve the worst skew and total capacitance by 14% and 15% over the state-of-the-art clock mesh algorithm [19] on ISPD09 benchmarks.**

I. Introduction

With aggressive technology scaling to sub $32nm$ nodes, the impacts of process variation, power supply noise, and temperature fluctuation on clock network are becoming more critical [2], [14], [17], [24]. Such impacts result in larger clock skew, directly decreasing the performance of a VLSI system. Therefore, various design techniques to build more robust clock network have been introduced [6], [11]–[13], [15], [16], [18], [19], [22]. Among them, clock mesh with global clock tree in Fig. 1 has been shown to be highly effective in minimizing the impacts of PVT (process, voltage, and temperature) variations on clock skew [20], [21], thanks to the redundant current paths from a clock source to a sink.

The key challenge in clock mesh design is to optimize two conflicting objectives, power dissipation and worst skew (e.g., variation tolerance) [2]. The redundant mesh structure makes clock skew less sensitive to the variations, but substantially increases total capacitance (leading to higher power consumption). In order to build more robust and power efficient clock mesh, various algorithms have been proposed. [4] proposed wire sizing for an existing clock mesh to reduce capacitance without degrading clock skew. [22] proposed set-covering based buffering and mesh reduction for a given mesh configuration to reduce power consumption while keeping similar variation tolerance. [19] suggested an iterative technique to determine a regular mesh configuration, followed by buffering and mesh reduction techniques both extended from [22].

However, most of existing approaches are either incrementally tuning a given clock mesh or lacking global planning. [7], [19] do not take non-uniform sink distribution and different clock domains into consideration upfront, resorting to post-processing step (e.g., mesh reduction), which may result in a sub-optimal clock mesh. In all previous work, a sink taps the nearest clock mesh segment, which is not necessarily the best decision when taking uneven sink capacitance distribution and buffering blockages into the account.

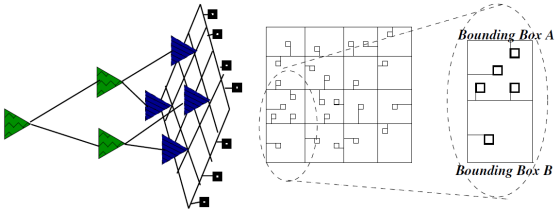


Fig. 1. Illustration of a clock mesh driven by a buffered global clock tree, and clock sinks tapping the clock mesh through stub wires [19].

Lastly, the impact of clock mesh design on global clock tree has not been comprehensively analyzed/reported before.

In this paper, we propose two novel binary linear programming formulations for clock mesh synthesis. The first one explores various (both regular and irregular) clock mesh configurations. It automatically determines the mesh dimension (rows and columns) and physical wire locations with lower skew at less power consumption, adapting to non-uniform load capacitance distribution (e.g., sink distribution or blockages). Once a clock mesh configuration is determined, the second binary linear programming is applied to assign each sink to the least loaded mesh segment in order to improve slew and skew. To our best knowledge, this is the first time that clock mesh synthesis is formulated in a rigorous mathematical programming manner, which 1) searches for the best trade-off between capacitance and skew and 2) targets for more uniform load capacitance distribution. The major contributions of this paper include the following:

- We develop a simple/linear but high-fidelity skew bound model for clock mesh synthesis, which is incorporated into our clock mesh synthesis framework.
- We propose a novel binary polynomial programming formulation for clock mesh synthesis, which is further linearized into binary linear programming based on [1] for lower computational overhead. Our formulation can explore both regular and irregular mesh configurations to determine the optimal mesh dimension (e.g., row/column) and find the best trade-off between skew and capacitance (or power).
- We propose balanced sink assignment using binary linear programming in order to balance effective capacitance seen by each buffer, which will result in smaller slew and skew. We also take buffer blockages into consideration.

The rest of the paper is organized as follows. Section II provides notations/assumptions in this work, and Section III presents our proposed algorithm. Experimental results are in Section IV, followed by the conclusion in Section V.

II. Notations & Assumptions

The notations in this paper are listed in Table I. l_m^s is to measure distance between a sink s and a mesh candidate m during the mesh

978-1-4244-8194-1/10/$26.00 ©2010 IEEE

TABLE I
THE NOTATIONS IN THIS PAPER.

r_o	unit wire resistance
c_o	unit wire capacitance
S	a set of the sinks (indexed by s)
t_s	latency to the sink s
C_s	load capacitance of the sink s
V	a set of vertical mesh candidates (indexed by v)
H	a set of horizontal mesh candidates (indexed by h)
L_m	the length of the mesh candidate m
l_m^s	the minimum distance from the sink s to mesh candidate m
W	the stub (a wire from a sink to the clock mesh) length limit
V^s	a set of mesh candidates $\in V$ within W from the sink s
H^s	a set of mesh candidates $\in H$ within W from the sink s
l^s	the stub length from the sink s to the (final) clock mesh
l_v^s	the vertical distance from the sink s to the clock mesh
l_h^s	the horizontal distance from the sink s to the clock mesh
B	the set of potential buffer locations (indexed by b) which are on the intersections of $v \in V$ and $h \in H$
b_s	the nearest buffer location $b \in B$ from the sink s
m_b^s	the Manhattan distance from the sink s to b_s
G	a set of mesh segments from the final mesh (indexed by g)
S_g	the length of a mesh segment g
n_g	the number of unblocked buffer locations on a mesh segment g ($0 \leq n_g \leq 2$)

construction step in Section III-C, and l^s denotes stub length between a sink and *the final clock mesh* which is built with a sub-set of $V \cup H$. If the stub from a sink s runs vertically to touch the final clock mesh, $l^s = l_v^s$, otherwise $l^s = l_h^s$.

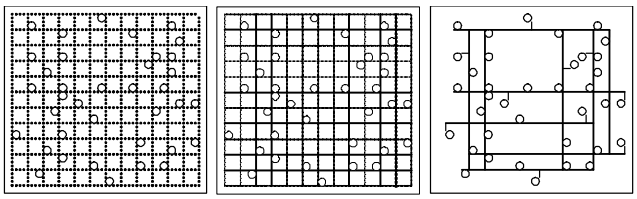
We assume that the clock buffers driving a clock mesh will be located at the cross-sections of mesh candidates [19], [22], as long as the location is unblocked [10] (e.g., the leaf level buffers in Fig. 1). These clock buffers become sinks to the global clock tree in Fig. 1. We additionally assume zero-skew and identical violation-free slew from a global clock tree to the inputs of these clock buffers. Also, while exploring various clock mesh configurations in our algorithm, we assume that a nominal strength buffer is placed at $\forall b \in B$. These assumptions on buffering are only to make our mesh synthesis less complex. Hence, more accurate buffer sizing (even removal) can be done after our flow. Even though we use a single wire type in this work (r_o, c_o), our techniques can be readily extended to handle various wire types.

III. ALGORITHM

In this section, we present our clock mesh synthesis. We begin with the overview in Section III-A. We propose a linear skew bound model for clock mesh in Section III-B, and then introduce our algorithm, which consists of mesh construction in Section III-C and balanced sink assignment in Section III-D.

A. Overview

Fig. 2 illustrates our algorithm flow. For a given input sink distribution, we create a set of horizontal/vertical clock mesh candidates as in Fig. 2 (a). Then, the binary linear formulation linearized from the binary polynomial formulation in Section III-C selects a set of mesh candidates to determine a mesh configuration as in Fig. 2 (b), minimizing total wire capacitance and skew bound based on Section III-B. Our formulation will not only automatically configure the dimensions of the clock mesh but also adjust the spacing between mesh wires from a given set of mesh candidates, accommodating non-uniform load capacitance distribution. Once a clock mesh is constructed, we connect sinks to the clock mesh by solving the binary linear programming formulation in Section III-D while trying to uniformly distribute load capacitance. After removing any dangling wire segments, we will obtain the final clock mesh in Fig. 2 (c).



(a) Sink distribution with mesh candidates in dashed lines. (b) Mesh candidates selected by mesh construction. (c) Final clock mesh after sink assignment and clean-up.

Fig. 2. Overall flow of the proposed algorithm.

B. Skew Bound Modeling

Due to non-tree structure, an accurate analytical model for skew in clock mesh is currently unknown. Therefore, a skew bound is used in [19] to select the right mesh configuration. However, the skew bound in [19] is too complex to be used in an analytical optimization framework because it is nonlinear and requires full knowledge of mesh configuration. Hence, we propose a simpler/linear skew bound model to enable efficient analytical optimization of mesh configuration under the assumptions in Section II.

Consider Fig. 3 where a sink i is right below a buffer b and sink j is connected to b through wire whose length is l. C_b is a parasitic/output capacitance in b and C_e represents any visible capacitance from b (e.g., other sinks and mesh segments in the proximity). Since i is right below b, we can regard that i is fully and solely driven by b. Then, the delay to i can be approximated as

$$t_i \approx R_b(C_b + C_e + C_i + c_o l + C_j) \tag{1}$$

Meanwhile, j has some distance to b and may get partially driven by other buffers in the proximity. Therefore, by ignoring the currents from other buffers, we can compute the upper bound of t_j as follows:

$$t_j \leq R_b(C_b + C_e + C_i + c_o l + C_j) + r_o l(\frac{c_o l}{2} + C_j) \tag{2}$$

Therefore, the worst latency difference in the proximity of b can be expressed as

$$\begin{aligned} K_b &= max(\{t_j - t_i | b_i = b_j = b\}) && (3)\\ &\leq max(\{r_o m_b^s(\frac{c_o m_b^s}{2} + C_s) | b_s = b\}) && (4)\\ &\leq max(\{r_o m_b^s(c_o W + C_s) | b_s = b\}) && (5) \end{aligned}$$

In detail, the latency difference between the sinks i and j is $r_o l(\frac{c_o l}{2} + C_j)$ from Eq. (1) and (2). Thus, if we replace l with the longest m_b^s (a Manhattan distance from a sink s to the buffer $b = b_s$), the skew in the proximity of b can be bounded by Eq. (4). However, as Eq. (4) is quadratic, it is expensive for optimization and cannot be used in a linear programming model. Since $m_b^s \leq 2W (\forall s \in S, \forall b \in B)$

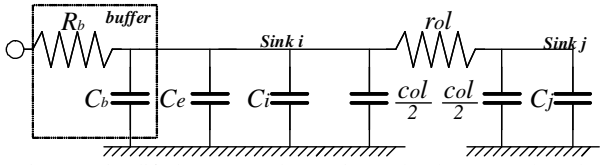


Fig. 3. Simplified RC network to estimate the skew upper bound.

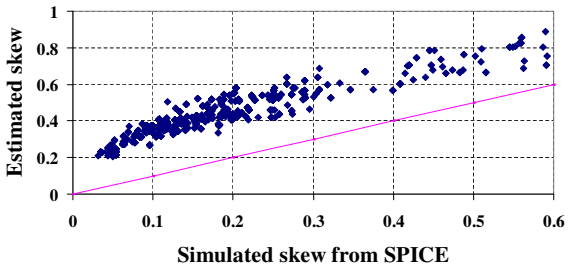
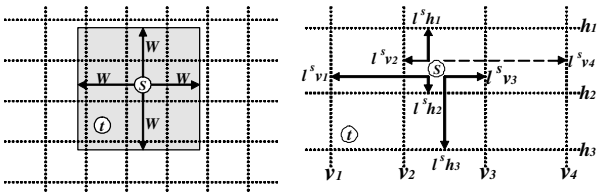


Fig. 4. Skew bound estimation based on Eq. (6).



(a) A set of mesh candidates for a sink s determined by W.

(b) 3 vertical and 3 horizontal mesh candidates and their distances from s, when zoomed-in from (a).

Fig. 5. Example of mesh candidates around the sink s within distance W.

due to Eq. (12) and (13), we can find a linearized skew bound in the proximity of b as in Eq. (5) (see Section III-C for details).

m_b^s is the key physical parameter which heavily affects the clock skew bound, and Eq. (5) implies that the longest distance from any sink to its nearest buffer has the first-order impact on the skew. m_b^s can be computed as in Eq. (11), even when a clock mesh is being constructed. Then, we take the following as our skew bound:

$$K = max(\{K_b | \forall b \in B\}) \tag{6}$$

This is not the exact definition of skew ($max(\{t_s | s \in S\}) - min(\{t_s | s \in S\})$), as our scope is limited to the proximity of each buffer. However, our estimation is to obtain a skew bound rather than the real skew value, and we found that our skew bound model is practically sufficient enough for optimization. Fig. 4 shows the fidelity of our skew bound estimation based on 740 different clock mesh configurations. Our skew estimation always overestimates the skew from SPICE simulation and shows high-fidelity ($r = 0.94$).

C. Mesh Construction

In this section, we present our binary linear programming formulation for clock mesh construction. The inputs are a set of sinks and a set of vertical/horizontal mesh candidates as discussed in Fig. 2 (a), and the output is a set of mesh candidates selected under capacitance/skew consideration. Although we used a set of regularly spaced candidates in our implementation, a designer can add custom candidates if needed. For example, when a custom block or IP require a special clock pin access, some design-specific candidates can be prepared to meet the requirements.

Let x_m be a binary variable set to 1 if the mesh candidate m is selected ($\overline{x}_m = 1 - x_m$). Then, we can formulate mesh construction as follows:

$$\textbf{min :} \quad c_o(\textstyle\sum_{m \in V \cup V} L_m + \sum_{s \in S} l^s) + \alpha K \tag{7}$$

$$\textbf{s.t :} \quad l^s = P(V_s \cup H_s, s) \qquad \forall s \in S \tag{8}$$

$$l_v^s = P(V_s, s) \qquad \forall s \in S \tag{9}$$

$$l_h^s = P(H_s, s) \qquad \forall s \in S \tag{10}$$

$$m_b^s = l_v^s + l_h^s \qquad \forall s \in S \tag{11}$$

$$\textstyle\sum_{v \in V_s} x_v \geq 1 \qquad \forall s \in S \tag{12}$$

$$\textstyle\sum_{h \in H_s} x_h \geq 1 \qquad \forall s \in S \tag{13}$$

$$K \geq K_b \qquad \forall b \in B \tag{14}$$

$$x_m \in \{0, 1\} \qquad \forall m \in V \cup H$$

The objective in Eq. (7) is to minimize weighted summation of total wire capacitance and skew bound (α is a weight parameter). The total wire capacitance consists of two factors: one from the mesh candidates (L_m) and the other from stubs (l^s) between the sinks and the clock mesh. l_s^v and l_s^h represent the vertical/horizontal distance to the final clock mesh. When connecting a sink s to a mesh, we only consider the mesh candidates whose distance to s is shorter than W as in Fig. 5 (a), which form V_s and H_s depending on direction. In order to guarantee the final result forms a mesh, we need to pick at least one vertical and horizontal candidates around any sink within W as in Eq. (12) and (13) which explains $m_b^s \leq 2W$ in Section III-B along with the buffer location assumption in Section II. Eq. (8) is to compute the shortest distance to any of selected mesh candidates in the proximity of a sink s. Eq. (11) is to compute m_b^s, a Manhattan distance from a sink to the nearest buffer location, which will be used to compute skew bound based on Eq. (6) as in Eq. (14).

The most challenging part of the formulation is how to compute l_s^v in Eq. (9), l_s^h in in Eq. (10), and l^s in Eq. (8) which are the vertical/horizontal/shortest distances to the clock meshes, when the final clock mesh is *unknown* yet (we are in the process of selecting mesh candidates to construct the clock mesh). Since Eq. (9), (10), and (8) share the same goal (computing distance to the final mesh), we express these distances in a compact mathematical form covering all possible mesh candidate selections using a polynomial function $P(U, s)$ without loss of generality, which is defined as follows:

$$P(U, s) = \sum_{u \in U} (l_u^s \prod_{\{i | l_i^s < l_u^s\}} \overline{x}_i x_u) \tag{15}$$

which returns the shortest distance to any mesh candidate $u \in U$ from a sink s. For example, $P(V_s, s)$ in Eq. (9) returns the shortest distance from s to any (vertical) mesh candidates in V_s, and $P(V_s \cup H_s, s)$ returns the shortest distance from s to any in $V_s \cup H_s$.

Eq. (15) is essentially to encode all possible mesh candidate combinations as a one-hot priority encoder, since we are interested in only the shortest distance from s to $\forall u \in U$. Fig. 5 (b) illustrates how to compute $l^s = P(V_s \cup H_s, s)$ where $V_s = \{v_1, v_2, v_3\}$, $H_s = \{h_1, h_2, h_3\}$ using the priority encoder idea in Eq. (15) ($v_4 \notin V_s$, because $l_{v_4}^s > W$). Note that l_h^s and l_v^s can be computed by providing V_s and H_s respectively to Eq. (15). First, we can sort all the candidates in $V_s \cup H_s$ in the ascending order of distance from s, which leads to $l_{h_2}^s < l_{v_2}^s < l_{h_1}^s < l_{v_3}^s < l_{h_3}^s < l_{v_1}^s$. Accordingly, we can assign higher priority to a mesh candidate with a shorter distance. The reason for prioritization is that once a mesh candidate with higher priority is selected, then other candidates with lower priorities can be ignored. Consider Table II which shows the value of l_s depending on the selection of mesh candidates in $V_s \cup H_s$. For instance, if h_2 is not selected but v_2 is selected (the second row in Table II), then $l^s = l_{v_2}^s$ regardless of the remaining candidates, making them again as *don't-cares*, which is the exact behavior of priority encoder. By leveraging a binary representation of a priority encoder, we can express l^s for the case in Fig. 5 (b) as following:

$$\begin{aligned} l^s = P(V_s \cup H_s, s) \quad &= \quad l_{h_2}^s x_{h_2} \\ &+ \quad l_{v_3}^s \overline{x}_{h_2} x_{v_3} \end{aligned}$$

$$
\begin{aligned}
&+ \quad l^s_{h_1}\overline{x}_{h_2}\overline{x}_{v_3}x_{h_1} \\
&+ \quad l^s_{v_3}\overline{x}_{h_2}\overline{x}_{v_3}\overline{x}_{h_1}x_{v_3} \\
&+ \quad l^s_{h_3}\overline{x}_{h_2}\overline{x}_{v_3}\overline{x}_{h_1}\overline{x}_{v_3}x_{h_3} \\
&+ \quad l^s_{v_1}\overline{x}_{h_2}\overline{x}_{v_3}\overline{x}_{h_1}\overline{x}_{v_3}\overline{x}_{h_3}x_{v_1} \qquad (16)
\end{aligned}
$$

Due to the nature of one-hot scheme, each term in Eq. (16) (also, $\prod_{\{i|l^s_i<l^s_u\}}\overline{x}_i x_u$ in Eq. (15)) is mutually exclusive. This is because some mesh candidates in $V_s \cup H_s$ can be selected by neighboring sinks such as t in Fig. 5 (b). For example, even when h_3 is selected for t (which means $l^t = l^t_{h_3}$), l^s will not be impacted if any mesh candidate with higher priority than h_3 is selected for s. Consequently, l_s in Eq. (15) is determined as l^s_u when $\prod_{\{i|l^s_i<l^s_u\}}\overline{x}_i x_u = 1$, which means $x_u = 1$ and any candidate i which has higher priority than u (e.g., $\{i|l^s_i < l^s_u\}$) is not selected. The priority encoder representation of l^s is the most compact expression of the shortest distance from a sink to the final clock mesh, covering all possible mesh candidate combinations. We can guarantees that l^s in Eq. (15) will have a valid distance, because each term in Eq. (15) is mutually exclusive and we have the Eq. (12) and (13) as constraints.

So far, we show that a stub length (l^s) can be expressed in a compact mathematical manner. However, as Eq. (15) makes our formulation binary polynomial programming which belongs to a class of discrete nonlinear optimization, the computational complexity is extraordinarily high. Therefore, we need to lower the complexity and we accomplished it by linearizing Eq. (15) with the transformation technique in [1]. It is known that binary linear programming is computationally much more affordable than binary polynomial programming [1], [5]. Mathematically, any binary polynomial $\prod_{i\in I} x_i$ where $x_i \in \{0,1\}$, can be linearized with one *continuous* auxiliary variable and $|I|$+1 linear constraints as following:

$$
\begin{aligned}
\prod_{i\in I} x_i \quad &\equiv \quad z \qquad (17)\\
z \quad &\leq \quad min(\{x_i | i \in I\}) \\
z \quad &\geq \quad max(\sum_{i\in I} x_i - |I| + 1, 0)
\end{aligned}
$$

Note that the auxiliary variable z can be declared as a continuous variable (helping the solver), since it will naturally be either 0 or 1 due to the linear constraints. Hence, based on Eq. (17), we can redefine a linearized $P(U,s)$ with additional linear constraints for the optimization in Eq. (7) as below:

$$
\begin{aligned}
P(U,s) \quad &= \quad \sum_{u\in U} l_u z_u \qquad (18)\\
z_u \quad &\leq \quad min(\{\overline{x}_i | l^s_i < l^s_u\} \cup \{x_u\}) \\
z_u \quad &\geq \quad max(\sum_{\{i|l^s_i<l^s_u\}} \overline{x}_i + x_u - |\{i|l^s_i < l^s_u\}|, 0)
\end{aligned}
$$

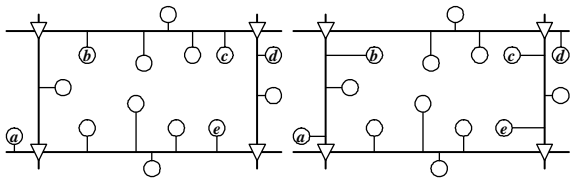
The key benefit of our analytical formulation over [19] is that it can explore various regular/irregular mesh configurations to find the best trade-off between total wire capacitance and skew bound by adapting to non-uniform capacitance distribution (either due to blockages or sink distribution).

D. Balanced Sink Assignment

During mesh construction in Section III-C, we assume that each sink would tap the clock mesh with minimum stub length as in Eq. (8). However, we observe that assigning a sink to the nearest mesh segment as in [19] is not necessarily the best in terms of slew/skew, despite it will minimize the capacitance from stubs. In fact, the larger skew or more slew violations caused by poorly balanced load distribution (e.g., from the nearest sink assignment scheme) may eventually result in even larger total capacitance due to larger overhead in buffering. Therefore, once a mesh configuration is determined based on Section III-C, we further explore the trade-off between capacitance and slew/skew during sink assignment by balancing load capacitance distribution.

TABLE II
TRUTH TABLE TO COMPUTE l_s FOR FIG. 5 (B).

l_s	x_{h_2}	x_{v_2}	x_{h_1}	x_{v_3}	x_{h_3}	x_{v_1}
$l^s_{h_2}$	1	*	*	*	*	*
$l^s_{v_2}$	0	1	*	*	*	*
$l^s_{h_1}$	0	0	1	*	*	*
$l^s_{v_3}$	0	0	0	1	*	*
$l^s_{h_3}$	0	0	0	0	1	*
$l^s_{v_1}$	0	0	0	0	0	1



(a) Sink assignment to the nearest mesh segment in [19]. (b) Sink assignment to the least loaded mesh segment.

Fig. 6. Example of sink assignment.

Fig. 6 illustrates the basic idea of our balanced sink assignment where Fig. 6 (a) is from the nearest assignment and (b) is from our balanced assignment. We can see that horizontal clock mesh segments are far more loaded than the vertical ones in Fig. 6 (a), which may lead to far longer latency and slew for b, c, e than d. Whereas, Fig. 6 (b) has more balanced capacitance distribution by assigning the sinks a, b, c, and e to the less loaded vertical mesh segments.

Additionally, we need to consider the existence of buffers in the proximity (The ISPD09 clock contest rule does not allow a buffer in a blocked region), which can influence the driving strength. For example, when a mesh segment is driven by a single buffer, it should be less loaded than one with two buffers. Such buffer consideration is important in our framework, as it can complement the formulation in Section III-C where blockage is not modeled.

Our balanced sink assignment can be formulated as a binary linear programming. Let Y^g_v and Y^g_h be the set of vertical and horizontal stub candidates touching the mesh segment g, respectively. Furthermore, suppose y^s_v and y^s_h are binary variables (set 1 if selected), representing a vertical and a horizontal stub candidate for a sink s, respectively. Then, our balanced sink assignment can be formulated as follows:

$$
\begin{aligned}
\textbf{min :} \quad & c_o(\textstyle\sum_{s\in S} l^s) + \beta E & & (19)\\
\textbf{s.t :} \quad & y^s_v + y^s_h = 1 & \forall s \in S \quad & (20)\\
& y^s_{\{v,h\}} \in \{0,1\} & \forall s \in S \quad & (21)\\
& E \geq \frac{c_o S_g + \sum_{s\in Y^g_v} y^s_v (c_o l^s_v + C_s)}{2^{(n_g-1)}} & \forall g \in G \quad & (22)\\
& E \geq \frac{c_o S_g + \sum_{s\in Y^g_h} y^s_h (c_o l^s_h + C_s)}{2^{(n_g-1)}} & \forall g \in G \quad & (23)\\
& l^s = l^s_v y^s_v + l^s_h y^s_h & \forall s \in S \quad & (24)
\end{aligned}
$$

The objective in Eq. (19) is to minimize the weighted summation of total stub capacitance and maximum relative load on a mesh

TABLE III
COMPARISON WITH MESHWORKS [19] ON ISPD09 CLOCK BENCHMARKS.

benchmark	MeshWorks [19]									Ours								
	mesh	tree cap [a]		mesh cap [a]		total	skew (ps)		CPU	tree cap [a]		mesh cap [a]		total	skew (ps)		CPU (sec)	
	size	wire	buf[c]	wire	buf	cap [a]	worst	norm[b]	(sec)	wire	buf	wire	buf	cap [a]	worst	norm[b]	CPLEX	total
ispd09f11	8x5	19.9	13.4	33.4	23.2	90.0	45.1	41.4	196.5	16.4	11.1	31.5	12.5	71.5	37.0	33.3	83.4	231.3
ispd09f12	6x6	17.9	12.0	29.7	16.7	76.2	39.3	36.1	142.2	14.5	9.7	28.2	12.2	64.6	30.2	26.4	227.2	365.3
ispd09f21	7x6	23.4	15.8	37.4	24.5	101.1	43.4	39.3	211.9	16.1	10.9	36.2	20.5	83.7	37.1	32.4	1898.9	2099.7
ispd09f22	4x6	10.9	7.3	18.7	13.9	50.9	43.9	40.2	100.6	8.2	5.5	18.3	9.8	41.8	34.1	30.0	276.5	352.2
ispd09f31	9x11	45.1	30.4	65.5	36.5	177.5	39.8	36.6	426.5	36.7	24.7	64.7	34.6	160.7	37.5	33.8	3040.8	3444.2
ispd09f32	10x10	37.5	25.3	55.1	24.3	142.2	38.0	34.8	383.4	29.7	20.0	51.2	27.7	128.7	36.6	32.6	2388.5	2697.8
ispd09f33	11x8	36.7	24.8	52.5	32.3	146.3	36.3	32.8	344.8	29.8	20.1	50.7	28.2	128.9	34.8	30.7	1701.4	1994.4
ispd09f34	8x11	30.8	20.8	44.0	26.8	122.5	42.4	39.2	284.2	27.2	18.3	44.1	27.3	116.9	34.4	31.0	749.2	1041.7
ispd09f35	11x11	37.7	25.4	50.5	46.7	160.3	40.8	37.0	496.2	26.6	17.9	45.3	32.3	122.1	38.7	35.2	709.1	1069.1
ispd09fnb1	7x4	1.3	0.9	6.9	7.0	16.2	40.8	36.9	65.4	1.0	0.7	8.5	5.3	15.6	37.6	32.5	406.3	461.6
ispd09fnb2	5x9	5.6	3.8	17.6	16.9	43.8	54.4	50.4	245.5	4.3	2.9	18.3	18.1	43.6	48.2	44.9	537.2	708.6
sum		266.8	179.9	411.3	268.8	1126.8	464.2	424.7	2897.2	210.5	141.9	397.1	228.4	978.0	406.2	362.8	12018.5	14465.9
ratio		1.27	1.27	1.04	1.18	1.15	1.14	1.17	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	4.15	4.99

[a] All the capacitances in pF.
[b] Also known as CLR (clock latency range) in ISPD09 clock contest.
[c] Buffer.

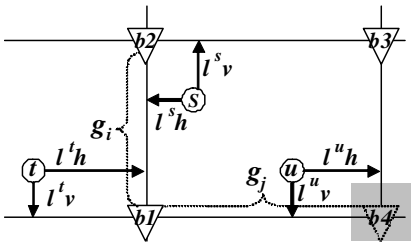


Fig. 7. Example for the balanced sink assignment.

segment (β is a weight parameter). As in Eq. (20), we consider two possible stubs and do not allow redundant tapping for all the sinks. The formulation can be extended for more possible stub candidates, but we stick to only two choices in order to keep m_b^s unchanged which impacts the skew bound model in Section III-B. Eq. (22) and (23) are to compute E, and Eq. (24) is to computed l^s which is the stub length expressed with y_v^s and y_h^s. Note that since all the mesh candidates are known at this time, l_v^s and l_h^s are known constants.

Fig. 7 shows an example of sink assignment where the sinks s, t, and u have two possible stub candidates, vertical and horizontal ones. The vertical mesh segment g_i along with the buffers b_1 and b_2 may drive s and t, if both choose the horizontal stubs ($y_h^s = y_h^t = 1$). Therefore, $Y_h^{g_i} = \{s, t\}$ and $Y_v^{g_i} = \phi$, thus the constraint corresponding to Eq. (23) can be shown as follows:

$$E \geq \frac{c_o S_{g_i} + y_h^s(c_o l_h^s + C_s) + y_h^t(c_o l_h^t + C_t)}{2^{(2-1)}(=2)}$$

The first term is the wire capacitance from g_i and the remaining terms are the conditional capacitance from stub and sink load. Since $n_{g_i} = 2$ due to b_1 and b_2, it is divided by 2. For the horizontal mesh segment g_j, the buffer location b_4 is blocked. Therefore, its driving strength should be penalized accordingly.

IV. EXPERIMENTAL RESULTS

We implemented our clock mesh synthesis in C++ and adopted CPLEX [23] as our discrete solver. We used the ISPD09 clock benchmarks [10] and modified the evaluation script (eval2009v10.pl) for SPICE simulation such that we can deliver the zero-skew clock and 100ps slew signal to the inputs of clock buffers driving a mesh, as global clock tree is not the main contribution of this work. However, we used BST [3], [9] with zero-skew option to estimate clock tree wire capacitance, and multiplied the average buffer-to-wire ratio of three winning clock tree algorithms from ISPD09 contest (which is 0.674) to predict required buffer capacitance on clock tree (e.g., non-leaf level buffers in Fig. 1). We used ngSPICE 18.0 [8] for our experiments with 45nm library from ISPD09 contest, and ran all the experiments on a 2.4GHz Linux machine with 4G RAM. As a post-optimization for both ours and [19], we performed buffer sizing based on iterative SPICE simulation to satisfy the slew constraint (100 ps) and minimize skew. Along with buffer sizing, we also applied mesh optimization techniques in [19], [22] to remove under-loaded mesh segments in all the experiments. We set W maximally 1.5mm in our experiments, which matches well with the data in [20].

For comprehensive study, we have not only implemented [19], but further extended it such that it can exhaustively search the entire solution space through iterative SPICE simulations (e.g., enumerate all the possible meshes and evaluate accurately through SPICE simulations). Therefore, each result for [19] in this section is our best possible solution. Since there is a trade-off between capacitance and skew, our objective for [19] is to pick a Pareto-optimal solution with least total capacitance without any slew violation. For our case, we used a default β but adjusted α such that our total mesh wire capacitance (which is only controllable during mesh construction) gets as close as possible to that from [19].

Table III comprehensively compares the performance of our techniques with [19]. The best mesh configuration we found for [19] is disclosed right next to the benchmark name. In all our optimizations, we considered 30 vertical/horizontal mesh candidates (in total 60). Overall, our approach yields 15% less capacitance with 14% less

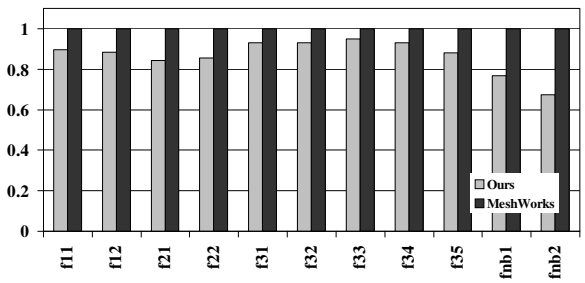
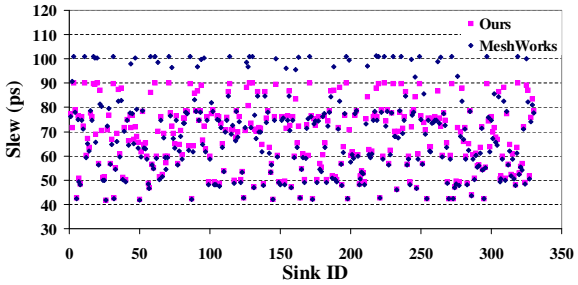
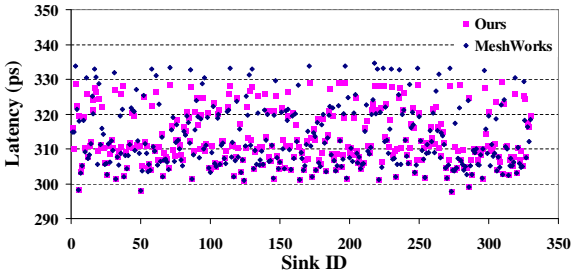


Fig. 8. Comparison of the HPWL of bounding boxes of mesh-driving clock buffers.



(a) Slew distribution comparison



(b) Latency distribution comparison

Fig. 9. Comparison of our balanced skew assignment with the one in [19] on ispd09fnb1.

worst skew at 5x computational overhead. However, note that the cpu time for [19] is to build one *specific/best* mesh configuration (e.g., 8x5 for ispd09f11) without including the total time spent in exploring/evaluating various other solutions, while ours includes all the time required to obtain the reported solution. CPLEX to solve the formulations in Eq. (7) and (19) accounts for 83% of our runtime. Regarding total capacitance, we can see considerable capacitance reduction from global clock tree distribution which is 27%. The reason is two fold. First, our approach produces a smaller bounding box of mesh-driving clock buffers (or sinks to the global clock tree), which will lead to generally a smaller global clock tree. Fig. 8 compares the HPWL of bounding boxes from ours with those from [19] where ours shows on average 13% smaller bounding boxes. The other reason is the distribution of clock buffers. [19] tends to distribute the clock buffers uniformly, meanwhile ours makes them locally clustered.

To see the effectiveness of our balanced sink assignment, we compared it against [19] starting from the identical mesh configuration, and plotted slew/latency distributions in Fig. 9. We can observe that our scheme not only minimizes the worst case slew/skew, but also narrows down the distribution itself. For example, ours has $\sigma = 13.5$ and 7.8, while [19] has $\sigma = 16.8$ and 8.9 for Fig. 9 (a) and (b), respectively.

V. Conclusion

We propose the first analytical approach for clock mesh synthesis. Our mesh construction based on linearized binary polynomial programming finds the right mesh configuration while minimizing both capacitance and skew. Our novel sink assignment based on binary linear programming further improves slew/skew by balancing the load on each mesh segment/buffer. Our analytical algorithm can significantly improve total capacitance (including global clock tree) and skew which will enhance overall VLSI system performance.

References

[1] W. P. Adams, R. J. Forrester, and F. W. Glover. A simple recipe for concise mixed 0-1 linearizations. *Operation Research Letters*, 33:55–61, Jan 2005.

[2] A. Chakraborty, P. Sithambaram, K. Duraisami, A. Macii, E. Macii, and M. Poncino. Thermal Resilient Bounded-Skew Clock Tree Optimization Methodology. In *Proc. Design, Automation and Test in Eurpoe*, 2006.

[3] J. Cong, A. B. Kahng, C.-K. Koh, and C.-W. A. Tsao. A clock distribution network for microprocessors. *ACM Trans. on Design Automation of Electronics Systems*, 4, Jan 1999.

[4] M. P. Desai, R. Cvijetic, and J. Jensen. Sizing of clock distribution networks for high performance CPU chips. In *Proc. Design Automation Conf.*, 1996.

[5] F. W. Glover and E. Woosley. Converting the 0-1 Polynomial Programming Problem to 0-1 Linear Program. *Operation Research*, 2:180–182, Jan 1974.

[6] M. R. Guthaus, D. Sylvester, and R. B. Brown. Process-induced skew reduction in nominal zero-skew clock trees. In *Proc. Asia and South Pacific Design Automation Conf.*, 2006.

[7] M. R. Guthaus, G. Wilke, and R. Reis. Non-Uniform Clock Mesh Optimization with Linear Programming Buffer Insertion. In *Proc. Design Automation Conf.*, 2010.

[8] http://ngspice.sourceforge.net.

[9] http://vlsicad.ucsd.edu/GSRC/bookshelf/Slots/BST.

[10] http://www.sigda.org/ispd/contests/09/ispd09cts.html.

[11] D. Lee and I. L. Markov. Contango: Integrated Optimizations for SoC Clock Networks. In *Proc. Design, Automation and Test in Eurpoe*, 2010.

[12] J. Long, J. C. Ku, S. O. Memik, and Y. Ismail. A self-adjusting clock tree architecture to cope with temperature variations. In *Proc. Int. Conf. on Computer Aided Design*, 2007.

[13] P. Mahoney, E. Fetzer, B. Doyle, and S. Naffzigern. Clock distribution on a dual-core multi-threaded Itanium-family processor. *IEEE Int. Solid-State Circuits Conf. Digest of Technical Papers*, 2005.

[14] J. Minz, X. Zhao, and S. K. Lim. Buffered clock tree synthesis for 3D ICs under thermal variations. In *Proc. Asia and South Pacific Design Automation Conf.*, 2008.

[15] M. Mondal, A. J. Ricketts, S. Kirolos, T. Ragheb, G. Link, N. Vijaykrishnan, and Y. Massoud. Thermally robust clocking schemes for 3d integrated circuits. In *Proc. Design, Automation and Test in Eurpoe*, 2007.

[16] S. Pullela, N. Menezes, and L. T. Pileggi. Post-processing of clock trees via wiresizing and buffering for robust design. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 15(6):691–701, June 1996.

[17] A. Rajaram, J. Hu, and R. Mahapatra. Reducing Clock Skew Variability via Crosslinks. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 25(6):1176–1182, Jun 2006.

[18] A. Rajaram and D. Z. Pan. Variation Tolerant Buffered Clock Network Synthesis with Cross Links. In *Proc. Int. Symp. on Physical Design*, Apr 2006.

[19] A. Rajaram and D. Z. Pan. MeshWorks: an efficient framework for planning, synthesis and optimization of clock mesh networks. In *Proc. Asia and South Pacific Design Automation Conf.*, 2008.

[20] 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*, 36:792–799, May 2001.

[21] S. Tam, S. Rusu, U. N. Desai, R. Kim, J. Zhang, and I. Young. Clock generation and distribution for the first IA-64 microprocessor. *IEEE J. Solid-State Circuits*, 35:1545–1552, Nov 2000.

[22] G. Venkataraman, Z. Feng, J. Hu, and P. Li. Combinatorial Algorithms for Fast Clock Mesh Optimization. In *Proc. Int. Conf. on Computer Aided Design*, 2006.

[23] www.ibm.com/software/websphere/ilog-migration/cplex.com.

[24] H. Yu, Y. Hu, C. Liu, and L. Heu. Minimal Skew Clock Embedding Considering Time Variant Temperature Variation with Automatic Correlation Extraction. In *Proc. Int. Symp. on Physical Design*, 2007.