Low-Power Clock Tree Synthesis for 3D-ICs

TIANTAO LU and ANKUR SRIVASTAVA, University of Maryland, College Park

We propose efficient algorithms to construct a low-power clock tree for through-silicon-via (TSV)-based 3D-ICs. We use shutdown gates to save clock trees' dynamic power, which selectively turn off certain clock tree branches to avoid unnecessary clock activities when the modules in these tree branches are inactive. While this clock gating technique has been extensively studied in 2D circuits, its application in 3D-ICs is unclear. In 3D-ICs, a shutdown gate is connected to a control signal unit through *control* TSVs, which may cause placement conflicts with existing *clock* TSVs in the layout due to TSV's large physical dimension. We develop a two-phase clock tree synthesis design flow for 3D-ICs: (1) 3D abstract clock tree generation based on K-means clustering and (2) clock tree embedding with simultaneous shutdown gates' insertion based on simulated annealing (SA) and a force-directed TSV placer. Experimental results indicate that (1) the K-means clustering heuristic significantly reduces the clock power by clustering modules with similar switching behavior and close proximity, and (2) the SA algorithm effectively inserts the shutdown gates to a 3D clock tree, while considering *control* TSV's placement. Compared with previous 3D clock tree synthesis techniques, our K-means clustering-based approach achieves larger reduction in clock tree power consumption while ensuring zero clock skew.

CCS Concepts: • **Hardware** → **3D integrated circuits**; **Clock-network synthesis**;

Additional Key Words and Phrases: 3D-ICs, TSV, clock tree synthesis, clock gating, optimization

ACM Reference Format:

Tiantao Lu and Ankur Srivastava. 2017. Low-power clock tree synthesis for 3D-ICs. *ACM Trans. Des. Autom. Electron. Syst.* 22, 3, Article 50 (April 2017), 24 pages.
DOI: <http://dx.doi.org/10.1145/3019610>

1. INTRODUCTION

The three-dimensional integrated circuit (3D-IC) has emerged as one of the most promising solutions to continue the scaling trajectory predicted by Moore's Law, as the traditional device scaling is approaching its physical limit. The 3D-IC stacks multiple chips vertically, and the communication between different chips is established by TSVs. TSVs substantially shorten the communication distance; thus, the 3D-IC has significant improvement in performance, power, and area compared to the conventional 2D circuit.

However, power consumption has become one of the major concerns for 3D-IC designers. Besides the reason that some 3D-ICs' applications are resource restricted (portable devices using batteries, for example), there are two other major factors: the immature development in 3D-ICs' power delivery network and the heat removal technique. On the one hand, in a 3D-IC, limited power pins are available and a more significant IR drop effect is expected as more chips stack together, and a high power density raises a

This work is supported by the National Science Foundation, under grant CCF-0917057.

Authors' addresses: T. Lu and A. Srivastava are with the Department of Electrical and Computer Engineering, University of Maryland, College Park, MD, 20742 USA; emails: {ttl, ankurs}@umd.edu.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.

© 2017 ACM 1084-4309/2017/04-ART50 \$15.00

DOI: <http://dx.doi.org/10.1145/3019610>

serious challenge to the power delivery network [Lu et al. 2016a]. On the other hand, the 3D-IC requires bonding materials between neighboring chips, which are poor heat conductors, and high power density creates thermal hotspots, which deteriorates 3D-ICs' reliability.

The clock tree is one of the largest and most frequently switched networks in a 3D-IC, making it a major contributor to the chip's total power in high-performance VLSI circuits (can take up to 70% of the chip's total power [Shelar and Patyra 2010]). Naturally, we are interested in designing a low-power clock tree for 3D-ICs. One of the well-known techniques for low-power clock tree design is clock gating. Clock gating exploits the fact that instructions are not executed with even frequency, causing spatial and temporal variations in the "on" and "off" states of the sequential logic. The clock gating technique applies a control signal at certain intermediate clock tree nodes to shut down all its descendants' (wires and sequential logic cells) clock signals when its downstream sequential logic cells are inactive, thereby reducing the dynamic power dissipated by wires, buffers, and sequential logic cells.

Clock gating for 2D clock trees has been extensively studied in the literature [Farrahi et al. 2001; Oh and Pedram 2001; Donno et al. 2003; Li et al. 2004; Chao and Mak 2008; Lu et al. 2012; Shen et al. 2010; Benini et al. 1999; Bolzani et al. 2009]. Instruction stream was introduced to calculate the switching probability for each tree node, and tree nodes with similar activity patterns were clustered greedily with higher priority [Farrahi et al. 2001; Oh and Pedram 2001]. A follow-up work integrated the register-transfer-level (RTL) clock gating approach into industry synthesis tools [Donno et al. 2003]. Benini et al. [1999] treated the circuit as a Finite-State Machine (FSM) and calculated the Observability Don't-Care (ODC) conditions using a formal mathematical formulation. Bolzani et al. [2009] provided a physical synthesis method to integrate the clock gating and power gating techniques. A deterministic clock gating design was proposed by Li et al. [2004], and it controls the gating circuitry based on a circuit block's actual usage during runtime. Chao and Mak [2008] considered clock gating simultaneously with buffer insertion to ensure zero clock skew. Lu et al. [2012] discussed a slew-aware clock gating method. Shen et al. [2010] optimized the placement of clock sinks to boost the power saving brought by clock gating.

Recently, with the emergence of 3D-ICs, 3D clock design has become an important research topic for 3D digital/mixed signal circuits. Generally the 3D clock tree synthesis is a two-step procedure: first the 3D abstract clock tree generation and then the 3D clock tree embedding. The 3D abstract clock tree is a binary tree structure, representing the connectivity from a centralized clock source to all the sequential logic cells. Currently the 3D abstract clock tree is designed purely from a wire length minimization standpoint. For example, Zhao et al. [2011] implemented a cutting-based approach for abstract clock tree generation, such that the 3D-IC was recursively partitioned in the X, Y, or Z direction. The location of the cutting line was determined based on the median value of the logic cells' coordinates. Another example is the NN-3D algorithm proposed by Kim and Kim [2011]. The authors greedily picked the closest sequential pairs and recursively formed the abstract clock tree. During the 3D clock tree embedding step, the physical locations of the intermediate clock tree nodes as well as the locations of *clock* TSVs are determined such that the clock tree's total wire length and clock skew are minimized. For example, Kim and Kim [2011] extended the 2D deferred-merge-embedding (DME) algorithm [Chao et al. 1992] to the 3D space in order to decide the length of clock tree edges and the locations of *clock* TSVs to ensure minimum wire length and zero-skew in 3D clock tree. In addition, Yang et al. [2011] investigated the impact of the TSV-induced stress on timing corners and optimized the location of clock buffers to minimize the stress-induced clock skew variation. Lu and Srivastava [2015a] incorporated TSV's lifetime due to stress migration into the 3D clock tree

synthesis flow. Lung et al. [2013] considered the reliability issues of *clock* TSVs and used a TSV fault-tolerant unit (TFU) to enhance clock trees' robustness against TSV failures. Further tuning techniques for the TFU and associated fault-tolerant 3D clock tree, such as better control of clock slew, TSV count, and clock slew, were proposed by Park and Kim [2015].

The primary concern of this article is the design of the 3D low-power clock tree using the clock gating technique. We notice that the conventional 2D clock gating approach cannot be applied directly to a 3D-IC, for the following reason: In a gated clock tree, the shutdown network provides enable signals for the shutdown gates. The shutdown network is a planar *Star* network (one centralized control center delivers the enable signals to all shutdown gates through separated wires). In 2D-IC, the wiring overhead of the shutdown network is usually ignored, and the authors of the 2D clock gating papers assume they can always deliver the enable signals to wherever needed [Farrahi et al. 2001; Oh and Pedram 2001; Donno et al. 2003; Li et al. 2004; Chao and Mak 2008; Lu et al. 2012; Shen et al. 2010; Benini et al. 1999; Bolzani et al. 2009]. Therefore, these works insert a control gate anywhere as long as there is power saving. However, the assumption that designers can insert shutdown gates at every tree edge and that the wiring overhead of the *Star* network is negligible is no longer valid for a 3D clock tree.

In a 3D clock tree, *clock* TSVs deliver clock signals from the clock source to each of the clock sinks, while *control* TSVs provide shutdown signals from a centralized control center to all shutdown gates [Lu and Srivastava 2014]. Since clock synthesis is usually performed after cell placement, layout whitespace for TSVs is limited. However, both TSVs (*clock* and *control*) occupy a large placement area and the reliability [Jung et al. 2012; Xie et al. 2016] and signal integrity [Liu et al. 2011] requirements enforce TSVs to maintain certain keep-out areas, both of which constrain the usage of TSVs. In addition, although TSVs offer short and fast vertical connections, excessive usage of TSVs increases the manufacturing overhead and makes the system's reliability questionable, due to the degradation of TSVs over time. The restriction that limited numbers of *clock* TSVs and *control* TSVs are available to designers changes the way that the 3D clock tree should be synthesized and how the clock gating technique should be applied.

Aiming at a low-power 3D clock tree that accounts for the TSV usage constraints, we believe that both the 3D abstract clock tree design and the 3D clock tree embedding procedure need to be optimized. First, the 3D abstract clock tree needs to account for not only the total wire length but also the similarities in sequential logic cells' "on" and "off" behavior, so that the maximum amount of power saving can be achieved with a limited number of shutdown gates. Second, during the clock tree embedding step, we need to decide at which tree edge to put shutdown gates and the physical locations of *clock* TSVs and *control* TSVs such that all of the TSVs can fit into the layout whitespace, clock skew is zero, and the power saving is maximum.

To summarize, this article focuses on the problem of designing a low-power clock 3D clock tree using the clock gating technique, while satisfying the layout whitespace constraint and zero clock skew. The designs of the 3D abstract clock tree, the shutdown clock network, and how the shutdown network affects the design of the clock tree embedding step are investigated and optimized. In the following sections, we show our methodology to smartly generate the 3D abstract tree that is preferable for shutdown gate insertion, and select and allocate shutdown gates such that all the TSVs can be legally placed inside layout whitespace, clock skew is zero, and clock tree power is minimized. More specifically, this article makes the following contributions:

- We identify the problem of layout-constrained shutdown network design for 3D-ICs and discuss its significant difference from 2D clock gating techniques.

- We propose a two-step design flow for low-power 3D clock tree synthesis, which comprises the abstract clock tree generation step and the clock tree embedding step.
- During the abstract clock tree generation step, we propose a K-means clustering [Lloyd 1982]-based algorithm to cluster nodes that are close to each other and have similar activity patterns.
- During the clock tree embedding step, we develop an SA-based heuristic to find at which tree edge to insert shutdown gates and to decide both the *clock* and *control* TSVs' locations by a force-directed TSV placer such that the overall power consumption is minimized.

The organization of this article is as follows. We provide an overview of related preliminary studies on a 3D clock tree's power model in Section 2. The follow-up section, Section 3, focuses on optimizing the 3D abstract tree topology. Section 4 focuses on optimizing the clock tree embedding procedure, where we formulate and solve an optimization problem to decide at which tree edge to put shutdown gates and the placement of *clock* TSVs and *control* TSVs. Section 5 analyzes the design complexity (i.e., number of clock TSVs) for a multilayer control scheme. Experimental results are presented in Section 6, and Section 7 concludes the article.

2. PRELIMINARY

Some basic models and concepts are introduced in this section. First, we briefly cover the 3D clock tree's power models. Then we illustrate the basic idea of obtaining an activity pattern for each clock tree node. After that we introduce the TSV's legal placement problem in 3D-IC and show that constrained layout whitespace impacts the decision of gate insertion and thus completely changes the clock tree design flow. Finally, an example of a 3D gated clock tree is shown to shed light on the influence of a 3D-IC's layout whitespace constraint on the clock gating strategy and power consumption.

2.1. Power Model

Clock tree power consists of two components: dynamic power and static power. Dynamic power is the summation of the following three: clock sink (module's) power P_m^T , clock tree's wiring power P_w^T , and controller network's wiring power P_w^C . Static power is consumed by the buffers along a long signal or power wire and any synchronized CMOS circuit. Since the estimation of static power along power wire needs thorough investigation of how the chip's PDN is synthesized, and CMOS's static power is largely decided by device technology, we focus on the dynamic power of the clock network in this article.

The dynamic power of a clock sink (P_m^T) is the weighted sum of its *active circuit power* P_A , when the clock is applied and the clock sink is active; *standby circuit power* P_S , when the clock is applied and the clock sink is idle; and *idle circuit power* P_I , when the clock is not applied and the clock sink is idle. The weight for each power component is the probability that the clock sink is in an active, standby, and idle state, respectively. The clock tree and controller network's wiring powers are caused by charging and discharging the wire capacitance. We ignore the controller network's wiring power (P_w^C) in this article because the signal in the controller network switches far less frequently than the clock signal, which turns on/off twice within one cycle. However, our power model can easily handle ad hoc designs that require frequent switching of the control signal as well. Detailed formulation for P_w^T is similar as in Oh and Pedram [2001] and thus is omitted for brevity.

2.2. Activity Pattern

In this section, we show the methodology for obtaining the activity pattern of each sink module and clock tree edge. We assume each sink module has registers with clock signal on its input. The activity pattern of a sink module is a Boolean string where “1” indicates the clock signal is applied and “0” means it’s shut down. When it is on “0,” the sink module is gated and it only consumes *idle circuit power* P_I . On the contrary, when it is on “1,” the sink module consumes *active circuit power* P_A or *standby circuit power* P_S , depending on whether data are fed or not. In other words, P_S is the amount of power dissipated on the capacitive components inside the module when the clock is still “on” even though the module is not doing any computation.

If the clock tree is completely gated (clock gate is inserted at every clock tree node), the activity pattern of an internal clock tree node (A_i) can be obtained by ORing (bit-wise OR) the activity patterns of all its descendant clock sinks ($S_1, S_2, \dots, S_k$). This procedure (Phase 1) is illustrated in Equation (1):

$$A_i = S_1 \bigcup S_2 \bigcup \dots \bigcup S_k. \quad (1)$$

If the clock tree is partially gated, then after obtaining the activity pattern for a completely gated clock tree (Phase 1), by preorder traversal, from the clock source to all clock sinks, a clock tree node inherits its parent’s pattern if gating is not applied between that node and its parent; otherwise, it stays unchanged. We call this procedure Phase 2, as shown in Equation (2):

$$\hat{A}_i = (\hat{A}_p * \bar{G}_i) \bigcup A_i, \quad (2)$$

where $\hat{A}_p$ is the activity pattern of i ’s parent clock tree node, A_i is i ’s activity pattern obtained from Phase 1, and G_i is a Boolean variable, which is 1 when the clock gate is applied between node i and its parent and 0 otherwise.

2.3. An Example: Minimizing Clock Power with TSV Placement Constraint

Figure 1(a) and Figure 1(b) are two 3D clock tree designs. Each design contains one clock source, seven nodes ($V_1 - V_7$), and four clock sink modules ($M_1 - M_4$). “AND” gate is used as the shutdown gate. Each module is associated with one activity pattern, recording the module’s activity over time. A module is considered as active when it is clocked and performing calculation (its activity pattern is filled/black). Otherwise, it is in standby mode when it’s clocked but performs no calculation (activity pattern is gray), or in idle mode if it is unclocked and performs no calculation (activity pattern is unfilled/white). The definition of the internal node’s activity pattern is slightly different from the module’s. The activity pattern for the internal node is filled for “1” (active) or unfilled for “0” (idle). Activity patterns for internal nodes are calculated by Phase 1 and Phase 2, described in Section 2.2. Figure 1(a) shows Phase 1, where the clock tree is completely gated, meaning shutdown gates are inserted at every clock tree node. Each internal node’s activity pattern is obtained by ORing the activity patterns of its children. Since we ignore the power consumption of the controller network, complete clock gating will achieve the optimal power saving. However, a complete clock gating might not be feasible for 3D-ICs due to its high usage of *control* TSVs.

Figure 1(b) shows another design that uses fewer *control* TSVs. Clock tree edges e_4 and e_5 are not gated. According to Phase 2 in Section 2.2, node V_5 inherits V_7 ’s pattern, but node V_6 ’s pattern stays unchanged (compared to V_6 ’s pattern in Figure 1(a)) because of the shutdown gate on edge e_6 . Meanwhile, the fourth state in M_4 ’s pattern is gray, which corresponds to the situation where a module is clocked but is not doing any computation.

Fig. 1. Suppose the control center that sends out enabling signals is at layer $Z_0 = 0$ and the maximum number of TSVs allowed is five. Design (a) has seven "AND" gates and seven TSVs, which is infeasible to place. Design (b) uses only five TSVs while still achieving considerable power saving.

Table I compares the power consumptions for the designs in Figure 1(a) and Figure 1(b). We assume the module's active power $P_A = 30$, standby power $P_S = 5$, and idle power $P_I = 3$. We also assume the wiring power is 100 for each wire. "AND" gate and TSV consume two- and eight-unit power, respectively. The power values for the TSVs and the wires are scaled based on TSV capacitance values ($50 fF$) reported by

Table I. Illustrative Power Consumption

Power Contribution	Complete Gating Figure 1(a)	Partial Gating Figure 1(b)	No Gating
Planar wiring	400.0	466.7	700.0
TSV	9.3	13.3	24.0
Gate	12.0	8.0	0.0
Module	57.0	57.3	61.7
Total	478.3	545.3	785.7
Saving(%)	39.1	30.6	0.0

previous works [Sematech 2010; Zhao et al. 2011; Kim and Kim 2010] and global wire capacitance reported by ITRS roadmap (assuming $1,000\mu\text{m}$ long global wire) [ITRS 2010], respectively.

First, we calculate the overall power consumption for both designs and we find that Figure 1(b) uses fewer TSVs but still achieves considerable power saving compared to Figure 1(a), as shown in Table I.

Second, we show that blindly inserting gates may end up with infeasible TSV placement. For illustration purposes, we assume we can fit at most five TSVs into the layout whitespace. (In practice, we use a force-driven placer to legalize all TSVs.) Both designs in Figure 1(a) and Figure 1(b) use three *clock* TSVs. Figure 1(a) inserts shutdown gates at every possible location, resulting in four *control* TSVs (on gates G_3, G_4, G_5, G_6), which becomes an infeasible design under the layout whitespace constraint. On the other hand, Figure 1(b) only uses two *control* TSVs, which makes it a feasible design. The main observation we have from this example is that clock gating for 3D clock trees is quite different from 2D-ICs' case: because of the area overhead of TSVs, the design of the shutdown network in a 3D clock tree must be reconsidered along with the layout information. There is a tradeoff between TSV layout whitespace (or the usage of TSVs) and maximum power saving, and deciding where to insert the shutdown gate is the key to address this problem.

3. ABSTRACT 3D CLOCK TREE GENERATION

Abstract clock tree generation is the first step of the clock tree synthesis. An abstract tree is a tree topology whose leaves represent all sequential modules (clock sinks), and the root represents the clock source. The clock signal flows from the clock source, through the edges, to all clock sinks. In this article, we assume the abstract clock tree is fully binary.

Most existing 3D abstract clock tree generation methods focus on minimizing the total wire length and total TSV usage, while the clock skew is minimized in the subsequent clock tree embedding step (see Section 4). There are three existing algorithms to generate a 3D abstract clock tree: 3D-MMM (method of means and medians) [Zhao et al. 2011], MMM-3D [Kim and Kim 2010], and NN-3D (nearest neighbor selection for 3D-ICs) [Kim and Kim 2011]. The 3D-MMM method is an extension of the 2D-MMM algorithm proposed by Jackson et al. [1990]. For a given TSV bound, the 3D-MMM algorithm partitions all the clock sinks horizontally (X/Y-cut) or vertically (Z-cut). During an X/Y-cut, the clock sinks are projected to a 2D plane and then separated evenly, while in a Z-cut, the clock sinks are partitioned such that the sinks from the same chip belong to the same subset, and a TSV is inserted between adjacent chips. The MMM-3D method [Kim and Kim 2010] uses a designer-specified parameter to control the partitioning direction (X/Y-cut or Z-cut) between sinks. The NN-3D method extends Edahiro's 2D bottom-up approach [Edahiro 1993]. NN-3D recursively selects a pair of

subtrees or clock sinks and then merges that pair. The selection criterion is based on a cost function that is a weighted sum of 2D Euclidean distance, TSV number, and capacitive load.

While these three algorithms are promising, they are designed primarily for wire length minimization purposes, not for clock power. In addition, they are not capable of handling the unique challenge in 3D clock tree design that the shutdown gates can no longer be inserted at every tree edge. As we have explained in Section 1, this is because the TSV placement whitespace is limited during the clock synthesis step, and excessive usage of TSVs brings extra manufacturing cost and raises TSV's reliability concerns [Lu et al. 2016b, 2016c]. Therefore, we aim to generate a 3D abstract clock tree such that the minimum number of shutdown gates is needed while simultaneously reducing the wire length and the power consumption of the clock tree. To achieve this goal, it is desirable that (1) clock sinks with similar activity patterns are clustered together (belong to the same subtree), and (2) clock sinks that are next to each other are clustered together. It's preferable to cluster sinks with similar activity patterns because a shutdown gate can turn off the entire subtree when none of the clock sinks in that subtree is active. It stays on (thus the entire subtree consumes dynamic power) even though there's only one clock sink in that subtree that is active. Clustering similar activity patterns ensures maximum shutdown rate and also reduces the need for the shutdown gates and therefore potentially reduces the number of *control* TSVs. Meanwhile, clustering clock sinks that are next to each other minimizes the total wire length, thus minimizing the wire's dynamic power.

To summarize, the 3D-MMM and MMM-3D algorithms only consider total wire length as an objective, but neither of them considers the dynamic power consumption during the charging and discharging behaviors happening on the capacitive loads. NN-3D models the capacitive load; however, it may consume more power for a highly frequently switched load with small capacitance than a large capacitance that switches less frequently. As far as the clock gating technique is concerned, we believe the quality of a gated low-power 3D clock tree heavily depends on its abstract clock tree. Therefore, in the following subsection, we propose a K-means clustering-based algorithm to generate a low-power 3D abstract clock tree, which produces a high-quality abstract clock tree that is suitable for the subsequent low-power clock tree embedding.

3.1. K-Means Clustering

As mentioned earlier, a low-power 3D clock tree geared toward the clock gating technique should capture both the similarities between activity patterns and the physical proximity between clock sinks. These "similarities" in activity patterns and physical locations can be captured by clustering algorithms, and we select the well-studied K-means clustering technique to cluster the clock sinks. There are other clustering algorithms such as Massberg and Vygen [2008] and Shelar [2012]. However, the K-means clustering algorithm captures the similarity of a set of points within a group (the distance function between any point to the group's centroid), but Massberg and Vygen [2008] and Shelar [2012] define the distance function pairwise. Capturing the activity pattern similarity within a group of clock sinks is one of the essences of the article. The K-means clustering algorithm divides up a set of points into K clusters, such that the sum of the within-cluster dissimilarity is minimized. In our binary abstract clock tree's case, $K = 2$. The within-cluster dissimilarity is defined as the average distance between two points within a cluster. We also associate each clock sink S_i with a tuple: $(x_i, y_i, z_i, C_i, A_i)$, where x_i, y_i, z_i are the location, and C_i and A_i are sink S_i 's capacitive load and activity pattern, respectively.

Specifically, our K-means clustering algorithm alternates between two steps: the assignment step and the update step.

3.1.1. Assignment Step. The assignment step assigns each clock sink to the closest clustering center. During the assignment step, the distance function that accounts for activity patterns' similarity and physical proximity between two clock sinks i and j (or two subtrees' roots) is defined as the dynamic power consumption of a subtree if node i and j are children of the same subtree's root. This formulation is better than clustering clock sinks purely based on their activity patterns, or physical locations, because these two might be contradictory objectives. For example, clock sinks that are far away might possess highly similar activity patterns, and clock sinks that are next to each other possibly have distinct activity patterns. Our proposed distance function combines the *best* of both worlds and therefore is used in this article. Specifically, the distance function d_{ij} is defined as in Equation (3):

$$d_{ij} = [c_w(|x_i - x_j| + |y_i - y_j|) + c_v|z_i - z_j| + C_i + C_j] \cdot \alpha_{ij}, \quad (3)$$

where $x_{i,j}$, $y_{i,j}$, and $z_{i,j}$ are i, j 's location, and c_w , c_v , $C_{i,j}$, and $\alpha_{i,j}$ denote the wire's unit capacitance, TSV's unit capacitance, node i, j 's loading capacitance, and switching activity similarity, respectively. Particularly, α_{ij} is defined as the percentage of the same switching behavior during the sampling period:

$$\alpha_{ij} = \frac{\text{sum}(\bar{A}_i \text{ XNOR } \bar{A}_j)}{m}, \quad (4)$$

where XNOR is the bitwise logical operation that is the complement of the exclusive OR operation, m is the length of the activity pattern, and $\bar{A}_i$ is the estimated activity pattern of node i , which is the average between two extreme cases (as shown in Equation (5)): a completely gated clock tree and a clock tree with no clock gating. Since at the 3D abstract clock tree's generation stage we know neither the physical implementation of the clock tree nor the exact location of shutdown gates, and therefore we don't know the exact activity pattern of each clock sink, we believe that taking the average of an ungated clock tree and a fully gated case to represent node i 's activity pattern, as shown in Equation (5), is a good estimation:

$$\bar{A}_i = \frac{1}{2} \cdot (A_i + \hat{A}_i), \quad (5)$$

where A_i is node i 's activity pattern when the clock tree is completely gated (shutdown gates inserted at every tree edge), and Figure 1(a) is an example of a fully gated clock tree. A_i can be obtained by simply ORing (bitwise OR) the activity patterns of all its descendant clock sinks (refer to Phase 1's Equation (1) for more details). $\hat{A}_i$ is node i 's activity pattern when the clock tree is not gated (no clock gating at all). In that case, according to Phase 2's Equation (2), every G_i is zero, which means every node inherits its parent's activity pattern and therefore every node's activity pattern is identical to the activity pattern of the tree root. The root's activity pattern can be obtained by bitwise ORing all the clock sinks' activity patterns.

3.1.2. Update Step. The update step replaces each clustering center by the average of clock sinks in its cluster. During the update step, the new cluster center is generated by averaging out the tuples belonging to the clock sinks in the same cluster: $(x_c, y_c, z_c, cap_c, a_c) = (\bar{x}_i, \bar{y}_i, \bar{z}_i, \bar{cap}_i, \text{round}(\bar{a}_i))$. The bitwise average of activity patterns represents the most frequently appeared activity pattern among the clock sinks in the same cluster.

3.2. Our Abstract Tree Generation Algorithm

The abstract clock tree generation algorithm is summarized in Algorithm 1. The inputs of the algorithm are the clock source's layer index (z_{src}), TSV bound (B), and a set (S)

ALGORITHM 1: Abstract Tree Generation**Input:**

1. TSV bound B .
2. Clock source's layer index z_{src} .
3. A set (S) representing clock sinks, which contains:
 4. Physical location of each clock sink (x, y, z);
 5. Activity pattern of each clock sink (A);
 6. Capacitive load of each clock sink (C).
7. Clock source's z index (z_{src})

Output:

8. Binary abstract tree.
9. AbstractTree3D (B, S)
10. **if** $|S| = 1$ // only one clock sink in S
11. return root (S)
12. **else if** $B = 1$ and $sizeof(Z) > 1$
13. **if** $\min(z) \leq z_{src} \leq \max(z)$
14. $S_1 = \{s | z_i \geq z_{src}\}, S_2 = \{s | s \notin S_1\}$
15. **else**
16. $S_1 = \{s | |z_i - z_{src}| = \min(|z_i - z_{src}|)\}$
17. $S_2 = \{s | s \notin S_1\}$
18. **end**
19. $B_1 = B_2 = 1$
20. **else**
21. $[S_1, B_1, S_2, B_2] = \text{K-means clustering}(S)$
22. **end**
23. AbstractTree3D(B_1, S_1)
24. AbstractTree3D(B_2, S_2)
25. $LeftChild(root(S)) = root(S_1)$
26. $RightChild(root(S)) = root(S_2)$
27. return $root(S)$.

representing the clock sinks. There are two terminating conditions: (1) when there's only one clock sink in S (line 10, $|S|$ means the number of sinks in set S), and the algorithm returns with this clock sink (line 11), and (2) when there's only one TSV per layer available ($B = 1$) and the clock sinks in S span over more than two dies ($sizeof(Z) > 1$), and the algorithm bipartitions S into two subtrees S_1 and S_2 . If z_{src} is contained in the range of $[\min(z), \max(z)]$, then S_1 contains all the clock sinks that have a layer index larger than z_{src} (line 14) while S_2 contains the rest. Otherwise, S_1 contains all the clock sinks that are located at the layer that is next to the z_{src} (line 16), while S_2 contains all other clock sinks. When neither of these two terminating conditions is satisfied, the K-means clustering algorithm bipartitions S and B to minimize the clock tree power and wire length (line 21). Then our function "AbstractTree3D" is called recursively on (B_1, S_1) and (B_2, S_2) and the roots of S_1 and S_2 become the current tree node, $root(S)$'s children.

Figure 2 shows a simple example for 3D abstract clock tree generation. In Figure 2, four modules (M_1 to M_4) are distributed on a two-layer 3D-IC. Each module has its activity pattern. When the TSV bound is 1, both the proposed K-means clustering-based method and the 3D-MMM algorithm generate the same abstract clock tree: the modules on the same layer belong to the same subtree. However, when the TSV bound is more than 1, for instance, 2 in Figure 2, the 3D-MMM algorithm only clusters the nearest modules, while the K-means method takes both the location and the activity pattern's information into account. The K-means method generates a 3D abstract clock tree that is suitable for shutdown gate insertion, which we explain in Section 4.

Fig. 2. The 3D abstract tree generated by our K-means clustering-based algorithm and the 3D-MMM algorithm [Zhao et al. 2011] for a given TSV bound. (a) 3D view where thick lines represent TSVs, each clock sink has a layer index (black dot indicates upper layer, and white dot is for lower layer), and each clock sink is associated with an activity pattern (black square is “on,” while white square is “off”). (b) The resulting 3D abstract trees where the black rectangles represent TSVs.

Another recent 3D clock tree synthesis heuristic is a greedy algorithm called NN-3D [Kim and Kim 2011]. NN-3D iteratively picks and connects a pair of clock sinks (or subtree roots) from clock sink set S with the lowest cost, defined as in Equation (6):

$$c_{ij} = c_w \cdot (|e_i| + |e_j|) + \beta_{TSV} \cdot c_v \cdot |z_i - z_j| + \alpha \cdot (C_i + C_j), \quad (6)$$

where c_w and c_v are wire and TSV capacitance, e_i and e_j are wire length from i and j to their parent node, and C_a and C_b are node i and j 's downstream capacitance, respectively. α and β_{TSV} are user-defined parameters that determine the weight of capacitance and TSV usage. In NN-3D, $\beta_{TSV} = 1$.

Then these two sinks (or subtree roots) are deleted from S and their parent node with an updated location, and capacitive load is returned to S . Similar iterations carry on until there is only one subtree root in S .

In this article, we are interested in how limited TSV placement whitespace affects the 3D clock tree synthesis decision (e.g., number of *clock* TSVs used). Therefore, for a given design benchmark, we vary β_{TSV} to obtain designs with different *clock* TSV numbers. These designs generated by NN-3D are used as baseline designs when analyzing the performance of the K-means clustering-based abstract clock tree generation heuristic.

4. CLOCK TREE EMBEDDING

For a given 3D abstract clock tree, the second phase of 3D clock tree synthesis, the wire embedding, determines the physical locations of the intermediate clock tree nodes, in order to minimize objective functions such as total wire length, clock skew, clock power, and so forth. In this section, we introduce the conventional clock skew and wire length-centered embedding method, and then we propose a new problem for 3D low-power clock tree generation, under certain placement whitespace constraints. We ask where are the best locations for shutdown gate insertion, and how the 3D shutdown network affects the design of the 3D clock tree network.

4.1. Conventional Embedding Method

A prevailing objective for 3D clock tree embedding is to minimize the total wire length while ensuring zero clock skew. Kim and Kim [2011] proposed a DME-3D (deferred-merge-embedding) algorithm to produce a 3D clock tree with near-optimal wire length.

The DME-3D algorithm is an extension of the 2D-DME work proposed by Chao et al. [1992]. In both DME algorithms, following a bottom-up fashion, the location of each intermediate clock node is investigated. With zero clock skew as a constraint and minimum wire length as an objective, the DME algorithm finds the solution space, namely, the “merging segments” (MSs) for each of the intermediate tree nodes from clock sinks all the way to the clock tree root. After that, starting from the clock tree root, a preorder traversal decides the exact location of each intermediate tree node such that the Manhattan distance from the intermediate node to its parent node is minimized. Then the algorithm inserts *clock* TSVs when a tree node is not at the same layer as its parent node. It is provable that the *clock* TSV should be placed at the same location as its parent node to achieve minimum wire length. In a word, given a 3D abstract clock tree, the DME-3D algorithm [Kim and Kim 2011] generates a 3D clock tree with low wire length and ensures zero clock skew. One other notable clock tree embedding technique includes the sDMBE algorithm [Zhao et al. 2011], which performs simultaneous clock buffering to control the clock slew during the DME-3D procedure.

4.2. Clock Gating for 3D Low-Power Embedding

In this section, we formulate a problem for 3D low-power clock tree embedding. Let $M = \{M_1, M_2, \dots, M_n\}$ represent clock sinks (modules). The location of each clock sink is (x_i, y_i, z_i) . Another input is the 3D abstract clock tree, which is generated using the K-means clustering-based algorithm proposed in Section 3.1. Let $V = \{v_1, v_2, \dots, v_{2n-1}\}$ denote the nodes of the abstract clock tree, where $v_1, v_2, \dots, v_n$ refer to leaf nodes (clock sinks) and the rest are internal tree nodes. We call the tree edge between v_i and its parent e_i . Each edge e_i is a potential location of a shutdown gate, G_i , and G_i is able to shut down the subtree rooted at v_i when necessary. If G_i is inserted at e_i , we always insert G_i at the end of the edge, next to v_i 's parent, so that we can save e_i 's power consumption when necessary. We assume one control unit is located at the center of layer Z_0 . The control signal paths form a *Star* network (each control gate connects to a centralized control unit through a separate wire). It's worth mentioning that the *Star* network is one simple but effective connecting structure. More sophisticated structures can be applied to avoid excessive wiring capacitance. For example, a designer may exploit the opportunities that multiple shutdown gates with the same enabling signals could share a common control signal path. Or one can add multilayer control units in multiple layers and the “enable” signals are issued from the control units to the clock sinks in the same layer. The comparison between a single-control-unit design and multiple-control-unit design is elaborated in Section 5. The following section assumes the control signal network is a *Star* network. We investigate the following simultaneous clock gating and TSV placement problem.

Problem formulation: Simultaneous shutdown gate insertion with TSV placement

Given a 3D ungated abstract clock tree generated by a conventional wire embedding technique (DME-3D [Kim and Kim 2011], for instance), we investigate how to construct a low-power 3D clock tree with simultaneous gate insertion under the constraint that the *clock* TSVs and *control* TSVs need to be legally placed inside the layout whitespace. Meanwhile, we also need to minimize the total wire length and clock skew.

The decision making is twofold. On the one hand, we need to determine where to insert the shutdown gates among all $2n - 1$ gating candidates (n = total number of clock sinks). As the shutdown signal requires *control* TSVs' connection to reach the control center and the TSV occupies a large placement area, inserting a shutdown gate at every tree edge is impractical in 3D-ICs. In addition, some tree edges don't need shutdown gates. For example, in Figure 3, if two child nodes possess the same activity patterns, there's no need to insert shutdown gates at these edges because these children

Fig. 3. M_1 and M_2 have identical activity patterns, and thus the shutdown gates G_1 and G_2 are unnecessary.

nodes switch “on” and “off” simultaneously. Another example could be in a situation where certain nodes are always “on” (activity patterns are mostly 1s). In that case, clock gating is also ineffective.

On the other hand, subject to a TSV placement whitespace constraint, we need to decide the number and the location for both *clock* and *control* TSVs, as they compete for placement whitespace resources. *Clock* and *control* TSVs need to be legally placed without overlap. Increasing the number of *clock* TSVs might reduce the total wire length and power for an ungated clock tree; however, when clock gating is considered, increasing the *clock* TSV number also implies that fewer *control* TSVs (therefore fewer shutdown gates) can be used.

4.3. Methodology

In this section, we present our methodology. We first develop a force-directed TSV placer to place the TSVs within the given placement whitespace and then propose an SA-based approach to solve the shutdown gate insertion problem. As the SA procedure could be time-consuming, we discuss approaches to speed up the heuristic.

4.3.1. TSV Placement. The number of *control* TSVs, which carry the shutdown gates’ control signals, is proportional to the number of shutdown gates used. However, since clock tree synthesis is usually performed after the placement of blocks/standard cells/IO pins, the layout whitespace left for TSVs is limited. Moreover, TSVs have larger dimensions than standard cells; for example, a TSV could occupy a $5\mu\text{m} \times 5\mu\text{m}$ to $30\mu\text{m} \times 30\mu\text{m}$ area Lu et al. [2016b, 2016c], depending on different fabrication technologies, as compared to less than $1\mu\text{m} \times 1\mu\text{m}$ standard cells at a sub-microtechnology node. In addition, TSVs have to maintain a certain distance from each other, due to mechanical [Jung et al. 2012; Lu et al. 2016b] and signal integrity [Liu et al. 2011] considerations. The TSV placement problem is to place the *clock* TSVs and *control* TSVs such that each TSV is located inside the layout area and outside other TSVs’ keep-out-zones (KOZs) [Lu and Srivastava 2015b; Lu et al. 2016c].

We adopt the force-directed placement idea in Viswanathan and Chu [2005] to solve the TSV placement problem. Overlap between TSVs and overlap between TSVs and placement boundaries produce forces to move TSVs to the overlapping-free regions

Fig. 4. Force-directed placer to place *control* TSVs inside the TSV layout whitespace.

during each iteration. The magnitude of the force is proportional to the area that overlaps. The direction of force is the combination of two kinds of overlap: (1) the overlap between TSVs and other TSVs' KOZs and (2) the overlap between TSVs and the boundary region. Figure 4 illustrates these two forces. Both *clock* and *control* TSVs are movable.

4.3.2. Shutdown Gate Insertion with Simultaneous TSV Placement. The core idea is to use an SA framework, which aims to find a high-quality selection of *control* TSVs that could fit into the current chip layout while minimizing the total clock power. We use the force-directed TSV placer that iteratively removes overlaps.

An exhaustive search approach to decide at which tree edge to put shutdown gates takes at least $O(2^n)$, where n is the number of clock sinks. For a modern synchronizing processor, n varies from several hundreds to several millions, depending on the granularity of the design. In this article, we propose an efficient heuristic based on SA, as shown in Figure 5. It starts with an estimation *control* TSV number that can fit into the whitespace. Then a set of shutdown gates is randomly selected to form the initial state. Iteration begins as we try to fit the selected *control* TSVs into the current layout whitespace, using the force-directed placer. If the placement fails, we reject the current selection immediately. Then we remove one *control* TSV in a high overlapping area, hoping the remaining *control* TSVs can be legally placed in the next iteration. Otherwise, if the placement is successful, the SA process probabilistically accepts or rejects the current selection, based on the clock tree's total power consumption. If the current selection is accepted, we add one more *control* TSV, hoping it'll bring us more power saving in the next iteration. If the selection is rejected due to an increase in clock power, we substitute one *control* TSV with one unused *control* TSV. We update the annealing temperature afterward. When the annealing procedure converges (no significant improvement is seen in several consecutive iterations), a high-quality set of shutdown gates is chosen, which achieves maximum power saving, and all the TSVs can be legally placed within the layout whitespace.

It is worth mentioning that after each successful TSV placement, the clock tree itself needs to be resynthesized to meet the zero-skew constraint. The time complexity of DME-based clock tree synthesis is bounded by $O(n)$, where n is the number of clock sinks [Kim and Kim 2011]. So this extra synthesis step won't increase the time complexity of our SA-based approach asymptotically.

Fig. 5. Flowchart of SA-based approach for simultaneous clock gating and TSV placement.

4.3.3. Setting Up Initial State. SA's solution quality and runtime heavily rely on the initial state. In order to speed up our SA-based algorithm, we develop a snippet to set up the initial state before SA begins. We start with an ungated routed zero-skew clock tree and estimate the *control* TSVs' count that can fit into the current layout whitespace, and then we iteratively select the gate with maximum power saving. The selection continues until the *control* TSV count reaches our early estimation, or all the gating candidates have been selected. The procedure is summarized in Algorithm 2.

ALGORITHM 2: Setting Up Initial State for SA

0. $S = \emptyset$; $U = \{G_1, G_2, \dots, G_{2n-1}\}$;
 1. m = estimation of available *control* TSV numbers;
 2. Construct a zero-skew ungated clock tree using DME-3D algorithm [Kim and Kim 2011];
 3. **Repeat**
 4. Calculate the power saving when inserting each one of the gates in set U ;
 5. Pick the gate insertion candidate G_i that has the largest power saving;
 6. Add G_i to S ;
 7. Remove G_i from U ;
 8. if *control* TSV is needed when inserting G_i , update m ;
 9. **Until** $m = 0$ or $U = \emptyset$.
-

5. EXPLORATION OF MULTILAYER CONTROL SCHEME

In this section, we explore the possibility of implementing multilayer control units.

As shown in Figure 6(a), the single control unit analyzes the shutdown condition of each clock tree branch and issues the “enable” signals to each of the shutdown gates through separate channels. These channels form a “star” network, in which shutdown gates at different layers are reached via *control* TSVs. For convenience, we assume an N -layer 3D-IC in which n_g shutdown gates are distributed among layers, and the i th layer contains n_i shutdown gates ($i = 1, 2, \dots, N$). The single control unit is located at the first layer. Therefore, a fully gated clock tree (e.g., Figure 1(a)) requires n_2 *control*

Fig. 6. (a) Single control unit distributes “enable” signals through a “star” control TSV network. (b) Multi-layer control units employ internal buses to achieve interlayer synchronization.

TSVs to reach the shutdown gates at the second layer, $2n_3$ control TSVs for the third layer, and $(N - 1)n_N$ control TSVs for the N^{th} layer. The total usage of control TSVs equals $\sum_{i=2}^N (i - 1)n_i$.

An implementation of multilayer control units is shown in Figure 6(b). In a multilayer control scheme, the master control unit encodes all the shutdown gates’ indexes and sends the encoded bits to the slave control units at other layers. The slave control units receive and decode the signal, and then distribute the “enable” signal through planar wiring. Although no control TSV is needed, signal TSVs are required between the master control unit and the slave control units, and the encoding and decoding circuitries yield extra overhead. A binary stream can be used to represent indexes of the selected shutdown gates. For example, “111000” represents that shutdown gates with indexes 1, 2, and 3 are “on” and gates with indexes 4, 5, and 6 are “off.”

Let’s consider the problem of how many TSVs are needed to reliably transmit the binary string from the master control unit to the slave control units without loss. First, we calculate the TSV count from the master control unit to the slave control unit at the second layer. The master control unit needs to encode the shutdown gates’ indexes at the second, third, ... and N^{th} layer, with a total count of $\sum_{i=2}^N n_i$. Therefore, there are $2^{\sum_{i=2}^N n_i}$ possible combinations. Assuming the appearance of each binary stream is a binary random variable, with the probability of p_i , according to the Shannon entropy theory, the entropy of the interface between the master control unit and the first slave control unit, H_1 , is defined as follows [Shannon 2001]:

$$H_1 = - \sum_{i=1}^{2^{\sum_{i=2}^N n_i}} p_i \log_2 p_i. \quad (7)$$

The entropy H_1 represents the minimum channel width to reliably transmit the binary stream without loss [Shannon 2001]. The upper bound of H_1 is $\log(2^{\sum_{i=2}^N n_i}) = \sum_{i=2}^N n_i$ when all the p_i s are equal. Similarly, the entropy of the interface between the first slave control unit and the second slave control unit $H_2 \leq \sum_{i=2}^N n_i$, and $H_{N-1} \leq \sum_{i=N-1}^N n_i$. Therefore, the upper bound of TSV usage in a multilayer control scheme is

$$H = \sum H_i \leq \sum_{j=1}^{N-1} \sum_{i=j+1}^N n_i = \sum_{i=2}^N (i - 1)n_i. \quad (8)$$

Fig. 7. Binary stream's entropy increases linearly with the number of shutdown gates.

The entropy from the simulation results are reported as follows. In order to compare the single control unit with the multilayer control units, we calculate the system's entropy value based on a completed gated design (e.g., Figure 1(a)). We execute a queue of instructions, and for each instruction, we analyze the indexes of the shutdown gates that need to be turned on. We count the appearance of each bit stream and divide it by the total number of instructions to obtain its probability (p_i).

For a two-layer 3D-IC in which shutdown gates are evenly distributed among layers, we show the total number of TSVs needed for a single-layer control and multilayer control scheme, respectively, with respect to the total number of shutdown gates, n_g , in Figure 7.

Figure 7 shows that the TSV required in a two-layer control scheme in practice scales approximately with $n_g/4$. This is because the appearance of each binary stream is not of equal probability. For example, when a parent node in a clock tree is clock gated, it will be unnecessary to clock gate any of its descendant nodes.

In summary, a multilayer control scheme uses fewer TSVs for delivering the control signal, at the cost of the overhead of encoding and decoding circuitries.

6. EXPERIMENTAL RESULT

Our 3D clock tree synthesis flow is implemented using Matlab and run on a Windows machine by an Intel Core i5 3.1GHz CPU and 12GB RAM. We evaluate the performance of our clock gating algorithm based on the benchmark circuits from the ISPD clock network synthesis contest [ISPD 2009]. These benchmarks contain up to 273 clock sinks. Since the benchmark circuits are originally designed for a 2D chip (with an area of A), we randomly partition all the clock sinks into N layers, with an area of $\frac{A}{N}$ on each layer. In this article, we focus on two-layer 3D-ICs ($N = 2$); however, our algorithms can be applied to any layer number as well. We randomly generate several whitespaces for TSVs, where TSVs are located. For each of the six ISPD benchmarks, we fix the total chip area and impose three sets of layout whitespace constraints (for TSV placement), small = 0.1%, medium = 0.15%, and large = 0.2%. As more placement whitespace is allocated for TSVs, more *clock* and *control* TSVs can be placed. When clock gating is applied, more shutdown gates are available to a designer, which help reduce the overall power dissipation.

The diameter of the TSV we use for simulation is $5\mu\text{m}$, and the minimum keep-out distance between centers of neighboring TSVs is $12\mu\text{m}$ [ITRS 2010]. The technology

Fig. 8. Clock power trends for benchmark f21 when sweeping *clock* TSV count. Too many *clock* TSVs increase clock power due to increased total wire length and TSV's capacitance.

parameters are based on the 45nm predictive technology [ITRS 2010], the 2009 ISPD clock network synthesis contest [ISPD 2009], and previous work on clock tree synthesis [Zhao et al. 2011; Kim and Kim 2011]: $r_w = 0.1\Omega/\mu m$, $c_w = 0.2 fF/\mu m$, $r_v = 0.035\Omega$, and $c_v = 50 fF$.

In addition, we follow the work of Oh and Pedram [2001] to generate random activity patterns for clock sinks that have the following characteristics: 10% of the instructions are made to appear 50% of the time in the instruction stream; the remaining 90% of the instructions make up the remaining 50% of the instruction stream. The average number of used modules per instruction is 20% of the total number of clock sinks.

6.1. Abstract Tree Generation

In this section, we compare the quality of our K-means clustering-based abstract tree generation algorithm (Algorithm 1) to the 3D-MMM method as well as NN-3D on a two-layer 3D-IC. We generate abstract clock trees using these three methods and then embed the resulting clock trees using the DME-3D algorithm assuming no clock gating. The DME algorithm ensures zero clock skew. We implement slew-aware buffer insertion (sDMBE) [Zhao et al. 2011] simultaneously with DME. Buffers are inserted into clock wiring wherever its loading capacitance exceeds a predefined threshold. A similar buffer insertion scheme was applied in Zhao et al. [2011].

For each benchmark, we sweep the *clock* TSV count to reveal the relation between *clock* TSV and total clock power. For K-means and 3D-MMM, this is done by controlling the TSV bound parameter (B) in Algorithm 1. For NN-3D, we sweep the *clock* TSV number by controlling β_{TSV} in Equation (6). The relation between *clock* TSV count and clock power generated by K-means for a representative benchmark f21 is shown in Figure 8.

The blue curve in Figure 8 shows that reducing the *clock* TSV count to a certain point saves clock power. However, increasing the *clock* TSV count beyond the minimum point increases the clock power. The clock power trend change is caused by not only the increase in TSV capacitance but also the increase in clock trees' overall wire length (shown as the red curve). When we sweep the clock TSV count, the clock tree

Table II. Comparison of 3D **Ungated** and **Buffered** Zero-Skew Clock Trees' TSV Count (#TSV), Wire Length (WL, in μm), Power Consumption (Pw, in W), Activity Similarity, and Simulation Runtime (RT, in s) Between K-means Clustering, 3D-MMM, and NN-3D Algorithms

Ckt.	K-Means						3D-MMM						NN-3D					
	Clk.	TSV	WL	Pw	Act. Sim.	RT	Clk.	TSV	WL	Pw	Act. Sim.	RT	Clk.	TSV	WL	Pw	Act. Sim.	RT
f11	36	120,276	0.133	0.710	2.04		36	159,222	0.165	0.639	0.17		41	114,364	0.127	0.625	1.56	
f12	32	100,643	0.115	0.668	1.05		34	154,039	0.159	0.622	0.17		7	126,169	0.129	0.648	0.88	
f21	22	120,522	0.128	0.734	1.01		40	152,703	0.158	0.607	0.16		30	132,666	0.138	0.601	0.79	
f22	10	72,938	0.083	0.763	0.86		28	97,999	0.107	0.675	0.19		33	66,432	0.081	0.633	0.64	
f31	26	315,267	0.327	0.705	1.07		57	428,124	0.426	0.639	0.31		39	327,971	0.333	0.627	3.51	
f32	27	204,226	0.216	0.696	6.11		54	298,797	0.300	0.621	0.22		51	240,363	0.248	0.655	1.84	
Avg.	26	155,645	0.167	0.713	2.02		43	215,147	0.219	0.634	0.20		34	167,994	0.176	0.632	1.54	
Ratio	—	1.00×	1.00×	1.00×	—		—	1.38×	1.31×	0.89×	—		—	1.08×	1.05×	0.89×	—

Fig. 9. Wire length comparison between 3D-MMM and K-means.

topology changes. This means a different clock TSV count changes not only the total TSV capacitance but also the total wire length and therefore total wiring capacitance.

For each benchmark circuit, we sweep the *clock* TSV count and pick the design that results in the lowest clock power. The results for the **ungated** and **buffered** clock tree are shown in Table II.

First, let's first focus on the comparison between K-means and 3D-MMM. As shown in Table II, our K-means clustering-based algorithm produces 3D abstract clock trees with shorter total wire length and less power consumption, in every ISPD benchmark. For an ungated clock tree, our clustering algorithm considers capacitive load, which improves the total wire length and saves the clock tree's dynamic power. On average, for buffered and ungated clock trees, compared to the 3D-MMM algorithm, the total wire length reduction achieves 28% ($1.00\times$ v.s. $1.38\times$), and power reduction is 23.7% ($1.00\times$ v.s. $1.31\times$).

The wire length and power reduction occur for two reasons. First, 3D-MMM groups the clock sinks based on the median value of their x (or y) coordinates, while the K-means clustering clusters the clock sinks that are close to each other. Figure 9 reveals the wire length advantage of the K-means clustering over 3D-MMM.

In Figure 9, there are six clock sinks (M_1 to M_6). M_4 is physically close to M_1 , M_2 , and M_3 , and thus M_1 , M_2 , M_3 , and M_4 should intuitively be grouped together. However, when we apply 3D-MMM, as Figure 9(a) shows, M_1 , M_2 , and M_3 form one subtree and M_4 , M_5 , and M_6 form another. Grouping M_4 , M_5 , and M_6 inevitably increases the total wire length. Figure 9(b) applies K-means clustering, and the physically closed sinks, M_1 to M_4 form one subtree, while M_5 and M_6 form another.

The second reason is that after abstract tree generation, it is necessary to ensure zero clock skew during embedding (i.e., DME algorithm). When two children nodes have different capacitive loads, the DME algorithm introduces extra wiring (wire snaking) at the node with smaller capacitance. Therefore, a high-quality abstract clock tree should be able to balance the capacitive loads between two children nodes. The 3D-MMM method only considers the clock sinks' physical location while ignoring the capacitive loads of the clock sinks. This leads to significant wiring overhead in order to achieve zero clock skew. On the contrary, our K-means clustering method inherently considers the clock sink's capacitive load, thereby significantly reducing extra wiring overhead.

Second, when comparing K-means with NN-3D, on average, our K-means clustering-based algorithm produces buffered and ungated 3D clock trees with slightly shorter total wire length and less power consumption. We notice that NN-3D has better wire-length and power numbers in the f11 and f22 benchmarks. This is because for an ungated clock tree, adding activity similarity into the cost function leads to suboptimal power. In other words, the benefit for penalizing a pair of sinks with low activity similarity does not appear in an ungated clock tree.

However, including the activity similarity in the cost function brings more than 20% power reduction after shutdown gates are inserted. More details are presented in Section 6.2.

The runtime of K-means clustering is $\mathcal{O}(nkdI)$, where n is the total number of clock sinks, k is the number of clusters ($k = 2$ in our case), d is the dimension ($d = 3$ in our case), and I is the total number of iterations. K-means clustering converges when the centroid assignments no longer change, and it has been proved that if k and d are fixed values, the upper bound of time complexity is less than or equal to $\mathcal{O}(n^{dk+1} \log n)$ [Inaba et al. 1994]. Therefore, the upper bound for the K-means-based abstract clock tree generation algorithm is less than or equal to $\mathcal{O}(n^{dk+1} \log n \log n)$, since we need $\mathcal{O}(\log n)$ partitions. However, in our simulation, the number of iterations is often small until convergence (roughly $\mathcal{O}(n)$), and the total time complexity is roughly $\mathcal{O}(n \log n)$. The runtime is similar to 3D-MMM, which is also $\mathcal{O}(n \log n)$.

6.2. Clock Tree Embedding

The second phase of 3D clock tree synthesis, the clock tree embedding step, determines the locations of intermediate clock tree nodes and the *clock* TSVs. For a gated 3D clock tree, the location of shutdown gates and the locations of the *control* TSVs are also calculated in this step.

In this section, we construct the low-power gated 3D clock tree based on the abstract tree generated by the K-means clustering and NN-3D algorithms. Again, we sweep the *clock* TSV count to generate a set of abstract clock tree designs (setting the TSV bound (B) parameter for K-means and the TSV weight parameter, β_{TSV} , in Equation (6) for NN-3D). For each of the abstract clock trees, we use the DME-3D algorithm [Kim and Kim 2011] with slew-aware buffer insertion (sDMBE) [Zhao et al. 2011] to generate an ungated baseline design. The baseline design with the lowest clock power is selected and recorded, which is referred to as "Ungated Tree" in the table.

We apply the SA and the TSV placer introduced in Section 4.3 to solve the simultaneous shutdown gate insertion and TSV placement problem for each of the abstract clock tree designs. After clock gating, the optimal (in terms of clock power) design is referred to as "Gated Tree" in the table. Three sets of TSV whitespace area are investigated. A larger TSV whitespace area can accommodate more *clock* and *control* TSVs. The results for the baseline design and the optimized design are summarized in Table III.

Table III indicates that on average, clock gating achieves 37% and 21% power reduction for clock trees generated by K-means and NN-3D, respectively. When a larger TSV placement whitespace is given, more *control* TSVs can be legally placed, which allows more shutdown gates and more significant power saving.

Table III. Summary of the *Clock* TSVs' Count (Clk. TSV), Total Wire Length (WL, in μm), Total Power Consumption (Pw, in W), and Number of Buffers Used (Bufs) Before Clock Gating, and the *Control* TSVs' Count (Ctrl. TSV), Total Power Consumption (Pw, in W), and Simulation Runtime (RT, in s) After Clock Gating

Ckt.	TSV White-space	K-Means							NN-3D						
		Ungated Tree				Gated Tree			Ungated Tree				Gated Tree		
		Clk. TSV	WL	Pw	Bufs	Ctrl. TSV	Pw	RT	Clk. TSV	WL	Pw	Bufs	Ctrl. TSV	Pw	RT
f11	small	18	130,246	0.138	89	15	0.095	60	32	118,827	0.132	91	2	0.110	78
	medium	18	130,317	0.138	91	26	0.086	59	37	115,848	0.130	86	8	0.106	89
	large	18	130,317	0.138	91	37	0.082	73	39	113,322	0.128	87	17	0.102	93
f12	small	3	114,117	0.121	84	28	0.073	107	6	117,150	0.124	83	25	0.096	78
	medium	3	114,117	0.121	84	37	0.070	112	6	117,150	0.124	83	35	0.091	81
	large	3	115,304	0.122	83	48	0.068	151	6	117,150	0.124	83	45	0.090	84
f21	small	12	132,388	0.137	94	25	0.081	106	2	136,551	0.138	94	34	0.107	75
	medium	13	132,613	0.137	94	36	0.078	96	2	136,551	0.138	94	47	0.102	99
	large	12	132,388	0.137	94	49	0.077	104	2	136,551	0.138	94	60	0.100	91
f22	small	10	72,938	0.083	55	13	0.055	52	3	79,874	0.087	58	21	0.068	51
	medium	10	72,938	0.083	55	20	0.052	55	3	79,874	0.087	58	27	0.065	57
	large	10	72,938	0.083	55	28	0.050	57	3	79,874	0.087	58	36	0.065	63
f31	small	23	315,022	0.326	223	29	0.221	305	46	297,928	0.317	224	6	0.262	218
	medium	26	315,267	0.327	219	43	0.215	328	46	297,928	0.317	224	23	0.255	305
	large	23	315,022	0.326	223	63	0.208	231	57	297,059	0.318	221	29	0.252	254
f32	small	27	206,439	0.218	150	24	0.145	164	40	245,990	0.255	170	12	0.207	129
	medium	27	209,421	0.221	150	41	0.140	212	55	228,217	0.243	161	14	0.200	147
	large	27	204,226	0.216	151	59	0.133	171	53	228,776	0.244	167	32	0.198	149
Average		16	162,001	0.171	116	35	0.107	136	24	163,590	0.174	119	26	0.138	119
Ratio		—	—	1.00×	—	—	0.63×	—	—	—	1.00×	—	—	0.79×	—

When comparing **ungated** 3D clock trees generated by K-means to NN-3D, we find the total wire length, number of buffers used, and total power are similar (K-means average WL: 162,001 μm , buffers: 116, power: 0.171W; NN-3D average WL: 163,590 μm , buffers: 119, power: 0.174W). However, after shutdown gate insertion, on average K-means reduces the total clock power to 0.107W, much lower than NN-3D (power: 0.138W).

The K-means heuristic brings us more power benefit because it considers the sink switching activity during the abstract clock tree construction phase. We have already shown in Table II that on average, clock trees generated by K-means have higher (0.713) activity similarity than those generated by either NN-3D (activity similarity: 0.632) or 3D-MMM (0.634).

Figure 10 illustrates the interaction between *clock* TSVs and *control* TSVs. For a fixed TSV whitespace area constraint, increasing the number of *clock* TSVs shrinks the available whitespace for the *control* TSVs. There exists an optimal allocation for *clock* and *control* TSVs such that the clock power is minimized. An optimal ungated 3D clock tree design (i.e., minimum power) does not yield an optimal gated 3D clock tree. In Figure 10, for example, for an **ungated** clock tree, the optimal clock TSV number is 22. However, blindly inserting shutdown gates into this clock tree does not produce the optimal solution. In fact, when clock gating is applied, the optimal clock tree is the one with only 13 clock TSVs. Figure 10 shows that the impact of *control* TSVs needs to be considered during the 3D clock tree synthesis stage in order to obtain the optimal power saving.

At last, we present the comparison between placement-unaware 3D clock tree design and our placement-aware design. We use a completely gated clock tree as an example of TSV whitespace-unaware design and show the power consumption, TSV usage, and

Fig. 10. The clock power trend for benchmark f21 when a fixed TSV whitespace area is applied. Increasing the *clock* TSV count reduces the *control* TSV count. The 3D clock tree that is optimal for clock gating is different from the ungated clock tree of optimal power. The impact of *control* TSVs needs to be considered during the 3D clock tree synthesis stage in order to obtain the optimal power saving.

Table IV. Comparison of **Ungated**, **Gated**, and **Completely Gated** 3D Clock Trees

Ckt.	TSV Whitespace	Ungated Tree		Partially Gated Tree		Completely Gated		
		Clk. TSV	Pw	Ctrl. TSV	Pw	Ctrl. TSV	Pw	Violation
f11	small	18	0.138	15	0.095	241	0.075	6.8×
	medium	18	0.138	26	0.086	241	0.074	4.9×
	large	18	0.138	37	0.082	241	0.074	3.7×
f12	small	3	0.121	28	0.073	233	0.063	6.6×
	medium	3	0.121	37	0.070	233	0.063	4.9×
	large	3	0.122	48	0.068	233	0.063	3.6×
f21	small	12	0.137	25	0.081	233	0.072	5.6×
	medium	13	0.137	36	0.078	233	0.072	4.0×
	large	12	0.137	49	0.077	233	0.072	3.0×
f22	small	10	0.083	13	0.055	181	0.043	7.3×
	medium	10	0.083	20	0.052	181	0.043	5.4×
	large	10	0.083	28	0.050	181	0.044	4.0×
f31	small	23	0.326	29	0.221	545	0.173	9.9×
	medium	26	0.327	43	0.215	545	0.178	7.3×
	large	23	0.326	63	0.208	545	0.173	5.6×
f32	small	27	0.218	24	0.145	379	0.118	7.0×
	medium	27	0.221	41	0.140	379	0.119	5.0×
	large	27	0.216	59	0.133	379	0.116	3.7×

placement violation in Table IV. All clock trees in Table IV are generated using our K-means-based algorithm, with buffer insertion.

Table IV shows that a completely gated clock tree achieves better power compared to a partially gated clock tree; however, its TSV placement violation varies from 3.6×

to $9.9\times$, depending on the TSV whitespace availability. Therefore, in designs in which TSV placement whitespace constraints are applied, completely gated clock trees are not practical.

7. CONCLUSION

In this article, we propose a design flow for 3D low-power clock tree synthesis, using the clock gating technique. Our design flow consists of two phases: (1) first, we propose a K-means clustering-based algorithm to construct a 3D abstract tree topology that is suitable for shutdown gate insertion, and (2) second, we use simulated annealing and a force-directed TSV placer to select the shutdown gates, while ensuring all the *clock* TSVs and *control* TSVs can be legally placed in the placement whitespace. We coplace *clock* TSVs and *control* TSVs to enhance the usage of the TSV placement area, so that more *control* TSVs can be placed and better power savings are achieved. Our experimental results show that our algorithms can effectively generate a low-power 3D abstract tree and also decide at which tree edge to put shutdown gates and the TSV locations, which saves clock power while ensuring zero clock skew.

REFERENCES

- Luca Benini, Giovanni de Micheli, Enrico Macii, Massimo Poncino, and Riccardo Scarsi. 1999. Symbolic synthesis of clock-gating logic for power optimization of synchronous controllers. *ACM Trans. Des. Autom. Electron. Syst.* 4, 4 (Oct. 1999), 351–375.
- Leticia Bolzani, Andrea Calimera, Alberto Macii, Enrico Macii, and Massimo Poncino. 2009. Enabling concurrent clock and power gating in an industrial design flow. In *Proc. Des. Autom. Test in Euro.* 334–339.
- Ting-Hai Chao, Yu-Chin Hsu, Jan-Ming Ho, and A. B. Kahng. 1992. Zero skew clock routing with minimum wirelength. *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.* 39, 11 (Nov. 1992), 799–814.
- Wei-Chung Chao and Wai-Kei Mak. 2008. Low-power gated and buffered clock network construction. *ACM Trans. Des. Autom. Electron. Syst.* 13, 1 (Feb 2008), 20:1–20:20.
- Monica Donno, Alessandro Ivaldi, Luca Benini, and Enrico Macii. 2003. Clock-tree power optimization based on RTL clock-gating. In *Proc. 40th Design Automation Conf. (DAC'03)*.
- Masato Edahiro. 1993. A clustering-based optimization algorithm in zero-skew routings. In *Proc. 30th Design Automation Conf. (DAC'93)*. 612–616.
- Amir H. Farrahi, Chunhong Chen, Gustavo Tellez, and Majid Sarrafzadeh. 2001. Activity-driven clock design. *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.* 20, 6 (Nov. 2001), 705–714.
- Mary Inaba, Naoki Katoh, and Hiroshi Imai. 1994. Applications of weighted Voronoi diagrams and randomization to variance-based K-clustering: (Extended abstract). In *Proc. 10th Annual Symp. Comput. Geom. (SCG'94)*. ACM, New York, NY, 332–339. DOI: <http://dx.doi.org/10.1145/177424.178042>
- ISPD. 2009. Ispd 2009 clock network synthesis contest. Retrieved from <http://ispd.cc/contests/09/ispd09cts.html>.
- ITRS. 2010. ITRS. 2010. International technology roadmap for semiconductors. Retrieved from <http://www.itrs.net/>.
- Michael A. B. Jackson, Arvind Srinivasan, and E. S. Kuh. 1990. Clock routing for high-performance ICs. In *Proc. 27th Design Automation Conf. (DAC'90)*. 573–579.
- Moongon Jung, J. Mitra, D. Z. Pan, and Sung Kyu Lim. 2012. TSV stress-aware full-chip mechanical reliability analysis and optimization for 3-D IC. *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.* 31, 8 (Aug. 2012), 1194–1207.
- Tak-Yung Kim and Taewhan Kim. 2010. Clock tree embedding for 3D ICs. In *Proc. 15th Asia South Pac. Des. Autom. Conf. (ASP-DAC'10)*. 486–491.
- Tak-Yung Kim and Taewhan Kim. 2011. Clock tree synthesis for TSV-based 3D IC designs. *ACM Trans. Des. Autom. Electron. Syst.* 16, 4 (Oct. 2011), 48:1–48:21.
- Hai Li, S. Bhunia, Yiran Chen, K. Roy, and T. N. Vijaykumar. 2004. DCG: Deterministic clock-gating for low-power microprocessor design. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* 12, 3 (Mar. 2004), 245–254.
- Chang Liu, Taigon Song, Jonghyun Cho, Joohee Kim, Joungho Kim, and Sung-Kyu Lim. 2011. Full-chip TSV-to-TSV coupling analysis and optimization in 3D IC. In *Proc. 48th Design Automation Conf. (DAC'11)*. 783–788.

- Stuart Lloyd. 1982. Least squares quantization in PCM. *IEEE Trans. Inf. Theor.* 28, 2 (1982), 129–137.
- Jingwei Lu, Wing-Kai Chow, and Chiu-Wing Sham. 2012. Fast power- and slew-aware gated clock tree synthesis. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* 20, 11 (Nov. 2012), 2094–2103.
- Tiantao Lu, Caleb Serafy, Zhiyuan Yang, and Ankur Srivastava. 2016a. Voltage noise induced DRAM soft error reduction technique for 3D-CPU. In *Proc. 2016 Int. Symp. Low Power Electron. Design (ISLPED'16)*. ACM, 82–87.
- Tiantao Lu and Ankur Srivastava. 2014. Gated low-power clock tree synthesis for 3D-ICs. In *Proc. 2014 Int. Symp. Low Power Electron. Design (ISLPED'14)*. 319–322.
- Tiantao Lu and Ankur Srivastava. 2015a. Electromigration-aware clock tree synthesis for TSV-based 3D-ICs. In *Proc. 25th Ed. Great Lakes Symp. VLSI (GLSVLSI'15)*. 27–32.
- Tiantao Lu and A. Srivastava. 2015b. Modeling and layout optimization for tapered TSVs. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* 23, 12 (Dec. 2015), 3129–3132.
- Tiantao Lu, Zhiyuan Yang, and Ankur Srivastava. 2016b. Electromigration-aware placement for 3D-ICs. In *Int. Symp. Quality Electron. Design (ISQED'16)*. 35–40.
- Tiantao Lu, Zhiyuan Yang, and Ankur Srivastava. 2016c. Post-placement optimization for thermal-induced mechanical stress reduction. In *2016 IEEE Comput. Soc. Ann. Symp. VLSI (ISVLSI)* (2016), 158–163.
- Chiao-Ling Lung, Yu-Shih Su, Hsih-Hsiu Huang, Yiyu Shi, and Shih-Chieh Chang. 2013. Through-silicon via fault-tolerant clock Networks for 3-D ICs. *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.* 32, 7 (July 2013), 1100–1109.
- Jens Massberg and Jens Vygen. 2008. Approximation algorithms for a facility location problem with service capacities. *ACM Trans. Algorithms* 4, 4 (Aug. 2008), 50:1–50:15.
- Jaewon Oh and M. Pedram. 2001. Gated clock routing for low-power microprocessor design. *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.* 20, 6 (Jun. 2001), 715–722.
- Heechun Park and Taewhan Kim. 2015. Synthesis of TSV fault-tolerant 3-D clock trees. *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.* 34, 2 (Feb. 2015), 266–279.
- Sematech. 2010. Concerns of 3D Integration Technology Using TSV. Retrieved from http://www.semtech.org/meetings/archives/symposia/9028/Session2_3D/Lee_KangWook.pdf.
- Claude E. Shannon. 2001. A mathematical theory of communication. *SIGMOBILE Mob. Comput. Commun. Rev.* 5, 1 (Jan. 2001), 3–55. DOI: <http://dx.doi.org/10.1145/584091.584093>
- Rupesh S. Shelar. 2012. A fast and near-optimal clustering algorithm for low-power clock tree synthesis. *Trans. Comp.-Aided Des. Integr. Cir. Sys.* 31, 11 (Nov. 2012), 1781–1786.
- Rupesh S. Shelar and Marek Patyra. 2010. Impact of local interconnects on timing and power in a high performance microprocessor. In *Proc. ACM Int. Symp. Phys. Design*.
- Weixiang Shen, Yici Cai, Xianlong Hong, and Jiang Hu. 2010. An effective gated clock tree design based on activity and register aware placement. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* 18, 12 (Dec. 2010), 1639–1648.
- Natarajan Viswanathan and Chris Chong-Nuen Chu. 2005. FastPlace: Efficient analytical placement using cell shifting, iterative local refinement, and a hybrid net model. *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.* 24, 5 (May 2005), 722–733.
- Yang Xie, Chongxi Bao, Tiantao Lu, and Ankur Srivastava. 2016. Security and vulnerability implications of 3D ICs. *IEEE Trans. Multi-Scale Comput. Syst.* 2, 2 (Apr. 2016), 108–122.
- Jaeseok Yang, J. S. Pak, Xin Zhao, Sung Kyu Lim, and D. Z. Pan. 2011. Robust clock tree synthesis with timing yield optimization for 3D-ICs. In *Proc. 16th Asia South Pac. Des. Autom. Conf. (ASP-DAC'11)*. 621–626.
- Xin Zhao, J. Minz, and Sung Kyu Lim. 2011. Low-Power and Reliable Clock Network Design for Through-Silicon Via (TSV) Based 3D ICs. *IEEE Trans. Compon. Packag. Manuf. Technol.* 1, 2 (Feb. 2011), 247–259.

Received May 2016; revised September 2016; accepted November 2016