Detailed-Placement-Enabled Dynamic Power Optimization of Multitier Gate-Level Monolithic 3-D ICs

Sheng-En David Lin, *Student Member, IEEE*, and Dae Hyun Kim, *Member, IEEE*

***Abstract*—Monolithic 3-D integration is expected to provide significantly higher degree of device density than through-silicon-via-based 3-D integration due mainly to its nano-scale intertier connections. By stacking more than two device layers (multitier) within a 3-D chip, further wirelength reduction could be achieved, which can lead to additional performance and power benefits. In this paper, we propose a detailed placement algorithm called nonuniform-scaling-based placement to optimize the dynamic power consumption of multitier gate-level monolithic 3-D ICs. We also introduce delay- and length-based timing constraints to prevent potential degradation of the performance metric during placement. Under the same timing constraints, our algorithm reduces dynamic power consumption more effectively than the uniform-scaling-based placement algorithm by 2% to 14%.**

***Index Terms*—3-D IC, monolithic, multitier, power optimization.**

I. Introduction

MONOLITHIC 3-D integration is emerging as an improved way for 3-D stacking to increase device density further [1]. A monolithic interlayer via (MIV) used for an intertier electrical connection is much smaller than a through-silicon via (TSV) as shown in Fig. 1, so monolithic 3-D integration is almost free from area and capacitance overhead, whereas TSV-based 3-D integration suffers from area and capacitance overhead [2], [3]. Thus, monolithic 3-D integration is expected to enable the highest degree of wirelength reduction, performance improvement, power reduction, and intertier bandwidth improvement.

The wirelength reduction of monolithic 3-D integration could be converted into both power and performance benefits. For example, Panth *et al.* [4] used a design methodology that uniformly scales the cell locations of a high-quality 2-D placement result by a constant ratio $1/\sqrt{2}$ to generate two-tier monolithic 3-D placement. The location scaling generates overlaps among the cells, so the author uses a partitioning algorithm to distribute the cells into different tiers to remove the overlaps. This methodology, namely a uniform-scaling-based placement algorithm (USBP), scales down the length of each net by almost the same ratio as the constant scaling ratio, thereby reducing wirelength and dynamic power consumption. However, the reduced wirelength cannot be directly converted into a higher clock frequency if gate delay and pin capacitance dominate the critical path delay or the design has a potential power-density problem that will lead to a thermal problem. In this case, we can reduce the dynamic power consumption further by converting the total delay gain obtained from wirelength reduction into power saving. In this paper, we propose a 3-D detailed placement algorithm, namely a nonuniform-scaling-based placement algorithm (NUSBP), that provides more dynamic power reduction than the USBP algorithm under the same operating frequency.

It is expected that multitier 3-D integration will provide more benefits than two-tier 3-D integration [5], and some monolithic 3-D integration technologies can fabricate multiple device layers in a single package [6]. Thus, we also generate multitier monolithic 3-D placement results using the USBP and NUSBP algorithms and compare the quality of the algorithms for multitier monolithic 3-D ICs.

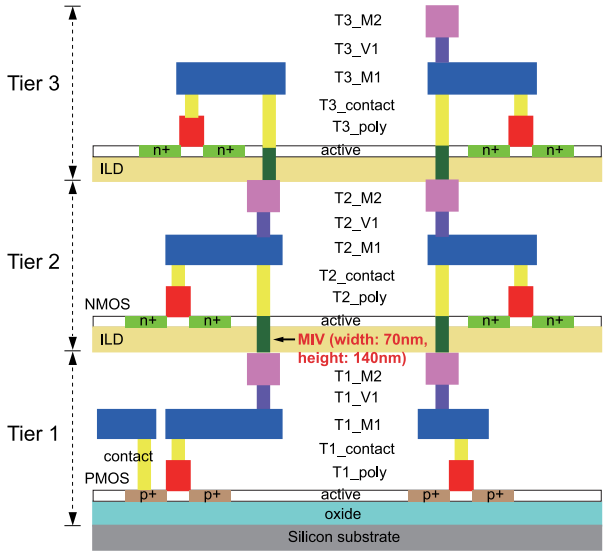

Fig. 1. Multitier monolithic 3-D integration.

Manuscript received October 7, 2016; revised February 8, 2017 and May 12, 2017; accepted June 28, 2017. Date of publication July 19, 2017; date of current version March 29, 2018. This work was supported in part by the New Faculty Seed Grant 125679-002 funded by Washington State University and in part by the DARPA Young Faculty Award D16AP00119 funded by the Defense Advanced Research Projects Agency. This paper was recommended by Associate Editor Y. P. Liu. *(Corresponding author: Dae Hyun Kim.)*

The authors are with the School of Electrical Engineering and Computer Science, Washington State University, Pullman, WA 99164 USA (e-mail: slin3@eecs.wsu.edu; daehyun@eecs.wsu.edu).

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

Digital Object Identifier 10.1109/TCAD.2017.2729401

Our contributions in this paper are as follows.

1) We develop a 3-D detailed placement algorithm that optimizes dynamic power consumption of monolithic 3-D ICs more effectively than the USBP.
2) We present theoretical background on the minimization of dynamic power consumption in monolithic 3-D ICs.
3) We develop length-based and delay-based timing constraint algorithms to prevent aggressive cell movements that could degrade the quality of the monolithic 3-D ICs.
4) We apply the uniform-scaling-based and nonuniform-scaling-based 3-D placement algorithms to multitier monolithic 3-D IC design and present their results with detailed analyses.

The rest of this paper is organized as follows. We review the previous work on the design of monolithic 3-D ICs in Section II. In Section III, we present theoretical background on dynamic power consumption and timing constraints. In Section IV, we propose a dynamic power optimization algorithm in detail. Then, we present and analyze simulation results in Section V. Finally, we conclude in Section VI.

II. Preliminaries and Related Work

In this section, we briefly review three monolithic 3-D IC design methodologies presented in the literature and discuss the uniform-scaling-based 3-D placement algorithm.

A. Design Methodologies for Monolithic 3-D ICs

Monolithic 3-D ICs can be designed in several different design levels. The most fine-grained design style is the transistor-level monolithic integration (TMI) proposed in [7]. In TMI, nMOS, and pMOS transistors of each standard cell are placed in different tiers, e.g., the nMOS and pMOS transistors are placed in the top and bottom tiers, respectively. In this case, MIVs are used for both intracell and intercell 3-D connections. TMI reduces the footprint area of each standard cell almost by half, but it overuses MIVs for intracell 3-D routing, which increases routing complexity and leads to unroutable designs. Block-level monolithic integration proposed in [8] is another monolithic 3-D IC design methodology in which each 2-D functional block is designed with 2-D standard cells and all the blocks are placed in 3-D using a 3-D floorplanner. Thus, nMOS and pMOS transistors are placed in both bottom and top tiers and MIVs are inserted into whitespace between the blocks. Gate-level monolithic integration (GMI) proposed in [7] places 2-D standard cells in 3-D. GMI can reuse existing 2-D standard cells and timing/power libraries. In addition, a design methodology using 2-D placement tools was proposed in [4] for the design of gate-level monolithic 3-D ICs and achieved almost 20% wirelength reduction and 16% power reduction. Thus, GMI is a prospective design methodology for monolithic 3-D IC design with respect to the design effort and the quality (wirelength, timing, and power) of 3-D ICs.

B. Uniform-Scaling-Based 3-D Global Placement

The 3-D global placement algorithm presented in [4] works as follows. First, they determine a downscaling ratio s based on the ratio between the width ($w_{2\text{-D}}$) of the 2-D layout and the width ($w_{3\text{-D}}$) of a target 3-D layout of the design ($s = w_{3\text{-D}}/w_{2\text{-D}}$).[1] Then, they shrink the size of each cell in a given standard cell library by the scaling ratio s and place the cells in 2-D using a commercial tool. By changing the library set from the downscaled one to the original one after the placement, the authors obtain a layout in which the cells overlap with each other. The overlaps are removed by partitioning, which also automatically converts the 2-D layout into a 3-D layout. The whole process of the downscaling of the cell size, placing cells in 2-D, and restoring the original cell size is very similar to placing the cells first with the original standard cell library and then scaling the locations of the cells uniformly by the same downscaling ratio s. Thus, we call this approach USBP.

USBP reduces the length of each net almost by the downscaling ratio s, so the dynamic power consumption and the delay of each net are also reduced. However, the reduced net delay cannot be directly converted into higher clock frequency if the delay of the critical path is primarily due to gate delay and pin capacitance. Thus, USBP can easily reduce the dynamic power consumption, but cannot guarantee that it can increase the clock frequency. However, we can convert the increased timing margin into further dynamic power consumption. In this paper, therefore, we propose an algorithm to convert the reduced net delay in noncritical paths into power reduction by a detailed placement algorithm, which we call NUSBP.

TABLE I
Variables Used in This Paper

Variable	Meaning
N_{T}	# tiers
α_i	Switching activity of net i
f_{clk}	Clock frequency
V_{DD}	Supply voltage
R_i	Output resistance of cell i
$R_{\mathrm{w},i}$	Wire resistance of net i
$C_{\mathrm{w},i}$	Wire capacitance of net i
$C_{\mathrm{p},i}$	Sum of the capacitance of all input pins connected to net i

III. Dynamic Power Reduction in Gate-Level Monolithic 3-D ICs

In this section, we analyze dynamic power consumption in monolithic 3-D ICs and investigate how we can reduce dynamic power consumption further. We also discuss timing constraints we take into account during dynamic power optimization. Table I shows the variables used in this paper and their meanings.

A. Power Reduction by Uniform Scaling

Dynamic power consumption is estimated by the following, well-known formula:

$$P_{\mathrm{int}} = \sum_{i \in N} \alpha_i \cdot f_{\mathrm{clk}} \cdot \left(C_{w,i} + C_{p,i}\right) \cdot {V_{\mathrm{DD}}}^2 \tag{1}$$

where N is the set of all the nets in the design and we are breaking down the capacitance into two capacitive components, wire capacitance and input pin capacitance of each net.

[1] Assuming both the 2-D and 3-D layouts have the same total silicon area, $w_{3\text{-D}}$ is $w_{2\text{-D}}/\sqrt{N_T}$.

TABLE II
IDEAL BENEFITS OBTAINED BY MONOLITHIC 3-D INTEGRATION AND USBP

	2-D	3-D
Wirelength	l	$\frac{l}{\sqrt{N_T}}$
Wire R and C	R_w, C_w	$\frac{R_w}{\sqrt{N_T}}$, $\frac{C_w}{\sqrt{N_T}}$
RC delay	$\propto R_w \cdot C_w$	$\propto R_w(\frac{C_{w,i}}{N_T} + \frac{C_{p,i}}{\sqrt{N_T}})$
Net switching power	$\propto (C_{w,i} + C_{p,i})$	$\propto (\frac{C_{w,i}}{\sqrt{N_T}} + C_{p,i})$

Assuming the 2-D layout and the target 3-D layout have the same total silicon area, the scaling factor that the USBP algorithm uses becomes $1/\sqrt{N_T}$. Thus, the USBP algorithm ideally reduces the length of each wire by $1/\sqrt{N_T}$, which is converted into delay and power reduction. Table II shows ideal benefits we can obtain by USBP.

Since the switching activity of each net, clock frequency, and the supply voltage are constants, we can reduce the dynamic power consumption by reducing the wire capacitance and/or the input pin capacitance as shown in (1). Reducing wire capacitance requires wirelength reduction, routing layer reassignment, wire spreading, and so on. Reducing input pin capacitance requires gate sizing (downsizing in most cases).

B. Conversion of Delay Benefit Into Power Reduction

As shown in Table II, the USBP algorithm reduces both net delay and dynamic power consumption by wirelength reduction. As explained in Section II-B, however, increasing the clock frequency in monolithic 3-D ICs is not possible or desirable. In this case, we can adjust the cell locations to convert the delay benefit into further power reduction as shown below.

Fig. 2 shows an example in which three cells are connected through two nets. Assuming that the switching activities of Net 1 and Net 2 in the figure are α_1 and α_2, respectively, the power consumption before uniform scaling is

$$P_{\text{before}} = f_{\text{clk}} \cdot {V_{\text{DD}}}^2 \cdot \left(\alpha_1\left(C_{w,1} + C_{p,1}\right) + \alpha_2\left(C_{w,2} + C_{p,2}\right)\right) \tag{2}$$

and the power consumption after uniform scaling is

$$P_{\text{after}} = f_{\text{clk}} \cdot {V_{\text{DD}}}^2 \cdot \left(\alpha_1\left(\frac{C_{w,1}}{\sqrt{N_T}} + C_{p,1}\right) + \alpha_2\left(\frac{C_{w,2}}{\sqrt{N_T}} + C_{p,2}\right)\right). \tag{3}$$

Thus, the power benefit ($\Delta P = P_{\text{before}} - P_{\text{after}}$) obtainable from USBP is

$$\Delta P = f_{\text{clk}} \cdot {V_{\text{DD}}}^2 \cdot \left(\alpha_1 \cdot C_{w,1} + \alpha_2 \cdot C_{w,2}\right) \cdot \left(1 - \frac{1}{\sqrt{N_T}}\right). \tag{4}$$

However, we can reduce the power consumption further by moving the cells. For instance, if α_1 is greater than α_2, moving Cell 2 closer to Cell 1 along Net 1 will reduce the power consumption. Suppose Cell 2 is moved toward Cell 1 by x(um) after the uniform scaling ($x > 0$). Then, the power consumption after the movement is

$$P'_{\text{after}} = f_{\text{clk}} \cdot {V_{\text{DD}}}^2 \cdot \left(\alpha_1\left(\frac{C_{w,1}}{\sqrt{N_T}} - c_u \cdot x + C_{p,1}\right) + \alpha_2\left(\frac{C_{w,2}}{\sqrt{N_T}} + c_u \cdot x + C_{p,2}\right)\right). \tag{5}$$

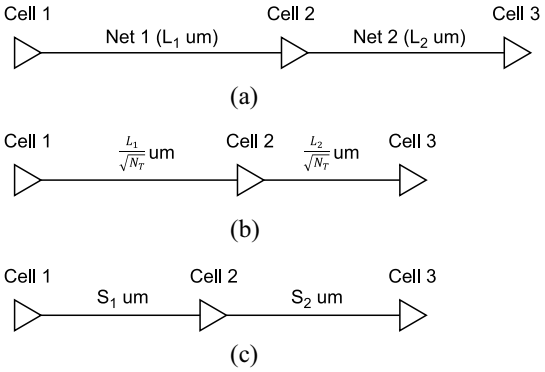

Fig. 2. USBP and NUSBP. (a) Before uniform scaling. (b) After uniform scaling (scaling factor: $\sqrt{N_T}$). (c) After nonuniform scaling.

Then, the new power benefit ($\Delta P' = P_{\text{before}} - P'_{\text{after}}$) becomes

$$\Delta P' = f_{\text{clk}} \cdot {V_{\text{DD}}}^2 \cdot \left(\alpha_1 \cdot C_{w,1} + \alpha_2 \cdot C_{w,2}\right) \cdot \left(1 - \frac{1}{\sqrt{N_T}}\right) + f_{\text{clk}} \cdot {V_{\text{DD}}}^2 \cdot c_u \cdot x \cdot (\alpha_1 - \alpha_2) \tag{6}$$

where c_u is the capacitance per micro-meter for the nets. The second term in (6) is positive because we assume that α_1 is greater than α_2. Thus, the power benefit goes up further by moving Cell 2 closer to Cell 1 in this case.

This *post-scaling adjustment of cell locations* can be performed in three different ways. First, the location of each cell is scaled with its own scaling ratio as follows:

$$(x_i, y_i) \rightarrow \left(\left(\frac{1}{\sqrt{N_T}} + s_{x,i}\right) \cdot x_i, \left(\frac{1}{\sqrt{N_T}} + s_{y,i}\right) \cdot y_i\right) \tag{7}$$

where (x_i, y_i) is the location of Cell i, $s_{x,i}$, and $s_{y,i}$ are small variations for the x- and y-coordinate scaling factors for Cell i, respectively. Second, the locations of all the cells are uniformly scaled down with a constant scaling ratio ($1/\sqrt{N_T}$) and the locations are slightly adjusted as follows:

$$(x_i, y_i) \rightarrow \left(\frac{x_i}{\sqrt{N_T}}, \frac{y_i}{\sqrt{N_T}}\right) \rightarrow \left(\frac{x_i}{\sqrt{N_T}} + \delta_{x,i}, \frac{y_i}{\sqrt{N_T}} + \delta_{y,i}\right) \tag{8}$$

where $\delta_{x,i}$ and $\delta_{y,i}$ are small, post-scaling displacement for Cell i. Third, the location of each cell is adjusted and then the locations of all the cells are uniformly scaled down by a constant scaling ratio ($1/\sqrt{N_T}$) as follows:

$$(x_i, y_i) \rightarrow \left(x_i + \delta'_{x,i}, y_i + \delta'_{y,i}\right) \rightarrow \left(\frac{x_i + \delta'_{x,i}}{\sqrt{N_T}}, \frac{y_i + \delta'_{y,i}}{\sqrt{N_T}}\right) \tag{9}$$

where $\delta'_{x,i}$ and $\delta'_{y,i}$ are small, prescaling displacement for Cell i. All of these approaches produce the same result, but we use the third approach in this paper and call it NUSBP.

Although NUSBP reduces the power consumption further, we should take two important constraints, timing and density constraints, into account in the computation of $\delta'_{x,i}$ and $\delta'_{y,i}$. The next section shows how we take the timing constraint into account and Section IV-D explains how we handle the density constraint.

C. Ideal Nonuniform Scaling Under Timing Constraints

In Fig. 2, suppose $d_{1,3}$ and $d_{1,3}''$ be the Elmore delays from the output of Cell 1 to the input of Cell 3 for the 2-D case and after nonuniform scaling, respectively. Then, $d_{1,3}$ is expressed as follows:

$$d_{1,3} = R_1\left(C_{w,1} + C_{p,2}\right) + R_{w,1}C_{p,2} + \frac{R_{w,1}C_{w,1}}{2} + R_2\left(C_{w,2} + C_{p,3}\right) + R_{w,2}C_{p,3} + \frac{R_{w,2}C_{w,2}}{2} \quad (10)$$

where R_1 and R_2 are the drive resistances of Cell 1 and Cell 2, respectively. If the distance between Cell 1 and Cell 2 in Fig. 2 becomes S_1(um) and that between Cell 2 and Cell 3 becomes S_2(um) after nonuniform scaling, the difference between $d_{1,3}$ and $d_{1,3}''$ becomes

$$\begin{aligned}\Delta d_{1,3}' = {} & \left(R_1C_{w,1} + R_{w,1}C_{p,2}\right)\left(1 - \frac{S_1}{L_1}\right) \\ & + \frac{R_{w,1}C_{w,1}}{2}\left(1 - \frac{{S_1}^2}{{L_1}^2}\right) \\ & + \left(R_2C_{w,2} + R_{w,2}C_{p,3}\right)\left(1 - \frac{S_2}{L_2}\right) \\ & + \frac{R_{w,2}C_{w,2}}{2}\left(1 - \frac{{S_2}^2}{{L_2}^2}\right). \end{aligned} \quad (11)$$

Setting $\Delta d_{1,3}'$ to zero and solving it with a constraint $S_1+S_2 = [(L_1 + L_2)/\sqrt{N_T}]$ gives us the ranges of S_1 and S_2 that do not degrade the delay from Cell 1 to Cell 3.

D. Delay-Based Timing Constraint

Applying the timing constraint presented in the previous section to a target cell requires the routing topologies of all the nets connected to the cell to obtain the net lengths. In this paper, however, we use the half-perimeter wirelength (HPWL) to estimate the length of each net for the following reasons. First, it might be too costly to construct a routing topology for each net at this step. Second, even if constructing a routing topology is not time-consuming, we do not need to constrain the range of the target locations of a target cell along a fixed routing topology at this step. Thus, we modify the timing constraint from the Elmore-delay-based constraint to the HPWL-based constraint for Cell 2 in Fig. 2 as follows:

$${\mathrm{HPWL}_1}'^2 + {\mathrm{HPWL}_2}'^2 \leq {\mathrm{HPWL}_1}^2 + {\mathrm{HPWL}_2}^2 \quad (12)$$

where HPWL_i is the HPWL of net i, HPWL_i' is the HPWL of net i after moving the target cell and scaling, and Net 1 and Net 2 are one of the input nets and one of the output nets connected to the target cell, respectively. We call this a *delay-based timing constraint*.

The delay-based timing constraint preserves the total delay of two adjacent nets connected through a target cell. However, applying this constraint is time-consuming because it should also consider all the combinations of the input and output nets associated with each target cell. For instance, suppose a target cell has m input pins and n output pins. Each pin is connected to a net and the driver of each net drives multiple cells. If we move the target cell, the length of each net changes (might not change depending on the location of the target cell), then we should check the timing of all the cells connected to the net. If the average number of cells connected to a net is r and the average input and output pin counts of a cell are m and n, respectively, moving a target cell requires application of the delay-based timing constraint to $r \cdot m + (r-1) \cdot n$ cells. The complexity of applying (12) to a cell is $\mathcal{O}(mn)$, so the complexity of applying the delay-based timing constraint to moving a cell is $\mathcal{O}(rmn(m+n))$. Since this is too time-consuming, we developed a length-based timing constraint, which is presented in the next section.

E. Length-Based Timing Constraint

Suppose the original HPWL of net i is HPWL_i and that of the net after nonuniform scaling is HPWL_i'. Then, the following constraint strictly preserves the delay of the net after scaling:

$$\mathrm{HPWL}_i' \leq \mathrm{HPWL}_i + \delta_i \quad (13)$$

where δ_i is a relaxation factor for net i and empirically determined for a given process technology.

The new HPWL of net i after nonuniform scaling is

$$\mathrm{HPWL}_i' = \frac{\mathrm{HPWL}_i + \Delta\mathrm{HPWL}_i}{\sqrt{N_T}} \quad (14)$$

so the substitution of (14) into inequality (13) gives

$$\Delta\mathrm{HPWL}_i \leq \left(\sqrt{N_T} - 1\right)\mathrm{HPWL}_i + \sqrt{N_T}\delta_i \quad (15)$$

which is a new length-based timing constraint with relaxation for each net. Since the delay is proportional to the square of the length of a net, a decreasing function, such as the following can be used for δ_i:

$$\begin{aligned}\delta_i &= k(\mathrm{um}) \quad \text{if } \mathrm{HPWL}_i \leq t(\mathrm{um}) \\ &= \frac{k}{\mathrm{HPWL}_i} + \frac{t-1}{t} \cdot k(\mathrm{um}) \text{ if } \mathrm{HPWL}_i > t(\mathrm{um}) \end{aligned} \quad (16)$$

where t is a sufficiently small wirelength, such as 5um and b and k are constants tuned for a given process technology by exhaustive delay simulations. We call this a *length-based timing constraint with relaxation*. If δ_i is zero, it is called a *length-based timing constraint*.

IV. Dynamic Power Optimization Algorithms

In this section, we present our algorithms for the minimization of dynamic power consumption in gate-level monolithic 3-D ICs.

A. Overall Algorithm

Fig. 3 shows the overall design flow and the step in the green box shows the proposed algorithm. For a given 2-D placement result, the NUSBP algorithm adjusts the cell locations in the layout and then uniformly scales the locations by $1/\sqrt{N_T}$ to generate an N_T-tier monolithic 3-D IC layout. The objective is to minimize the dynamic power consumption estimated by the following formula:

$$P = f_{\mathrm{clk}} \cdot {V_{\mathrm{DD}}}^2 \sum_{i \in N} \cdot(\alpha_i \cdot \mathrm{HPWL}_i) \quad (17)$$

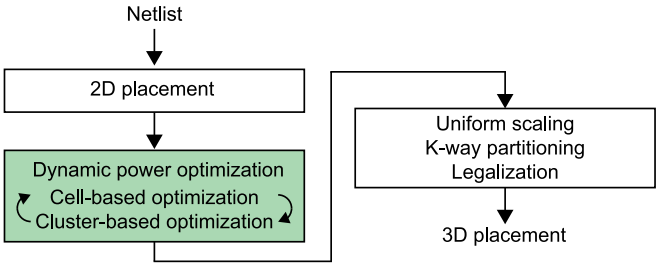

Fig. 3. Our 3-D IC design flow. The USBP skips the dynamic power optimization step.

while satisfying either the delay- or length-based timing constraints shown in Section III. The cell location adjustment is sequentially applied to either each cell or a set of cells (clusters) until there is no more noticeable improvement in the power consumption. The following sections describe how to find optimal locations for each cell or cluster and how to integrate the timing and density constraints into the optimization algorithm.

B. Finding Optimal Locations

For each cell in a given 2-D placement result, we find an optimal location that can minimize the sum of the dynamic power of all the nets connected to the cell. The idea is to move the cell in a direction we can reduce the sum of the dynamic power.

The following theorem helps find optimal locations that minimize the dynamic power consumption for a cell.

Theorem 1: For Cell A connected to k nets $(n_1, \ldots, n_k)$, construct two bounding boxes, one $(B_{q,1})$ without Cell A and the other $(B_{q,2})$ with Cell A, for n_q $(1 \leq q \leq k)$. Let B_A be the set of all the bounding boxes, $B_A = \{B_{1,1}, B_{1,2}, B_{2,1}, \ldots, B_{k,2}\}$ and EP_A be the set of all extremal points (four end points) of all the bounding boxes in B_A. Let T_A be the set of all intersection points of all pairs of the bounding boxes in B_A. Then: 1) the current location of Cell A is optimal or 2) there exists at least one optimal point in $T_A \cup EP_A$ that minimizes the sum of the dynamic power of all the nets connected to Cell A.

Proof: The objective function we minimize is $\lambda = \sum_{i \in N} \alpha_i \cdot \mathrm{HPWL}_i$. λ is piecewise linear, so the optimal points minimizing λ exist: 1) inside or on the boundary of some rectangles; 2) on some intervals (segments); or 3) on some extremal points (the endpoints of some intervals or rectangles) or intersection points. Since rectangles and intervals include their endpoints, at least one of the extremal points or the intersection points are optimal. ■

To apply the above theorem to finding optimal locations for a given target cell, we use the following observations.

1) $\lambda = \sum_{i \in N} \alpha_i \cdot (\mathrm{HPWL}_{x,i} + \mathrm{HPWL}_{y,i})$ where $\mathrm{HPWL}_{x,i}$ and $\mathrm{HPWL}_{y,i}$ are the x- and y-components of the HPWL of net i.
2) $\mathrm{HPWL}_{x,i}$ and $\mathrm{HPWL}_{y,i}$ are independent of each other, so we can optimize each of them separately (so we focus only on the x-coordinates from this point).
3) Suppose net i connects k cells, $\{C_1, C_2, \ldots, C_k\}$, and the x-coordinate of C_j is x_j. If C_t is the target cell to move, $x_{\min,i}$ and $x_{\max,i}$ are defined as the minimum and the maximum of $x_j (j = 1, \ldots, k, j \neq t)$, respectively. Then, $\mathrm{HPWL}_{x,i}$ linearly decreases as the target cell is moved in the positive direction from $-\infty$, stays constant between $x_{\min,i}$ and $x_{\max,i}$, and then linearly increases as the cell is moved toward ∞.
4) The x-coordinate of each intersection point in T_A is an extremal point of a rectangle in B_A.

In summary, the x-coordinate of an optimal point minimizing λ for a given cell is always an extremal point of one of the bounding boxes in B_A constructed for the cell. In addition, if we move the cell from $-\infty$ to ∞, λ linearly decreases, then stays constant, then linearly increases. Thus, instead of enumerating all the extremal and intersection points, we first construct B_A, extract the x-coordinates of the left and right extremal points of each rectangle in B_A, and sort the coordinates in the increasing order. Then, we find an optimal coordinate by the binary search. Algorithm 1 describes how to find an optimal x-coordinate for a given target cell using the binary search algorithm. The $Cost(x)$ function for a given x-coordinate x computes the dynamic power consumption when the cell is moved to x. The algorithm uses the observation that the cost function is convex.

Algorithm 1: Find an Optimal x-Coordinate for a Given Cell

Input: A given target cell A and a netlist N.
Output: An optimal x-coordiate of A.

```
K = {n|n ∈ N, A ∈ n};
B = {BoundingBox(n) = (x1, y1, x2, y2)|n ∈ K};
P = {x1(b), x2(b)|b ∈ B} ∪ x(A);
  // x(A) is the x-coordinate of A.
Array X = Sort (P);
l = 1, r = |X|;
while l < r do
    m = (l + r)/2;
    Cost_L = Cost(X[l]);
    Cost_M = Cost(X[m]);
    if Cost_L ≤ Cost_M then
        r = m;
    else
        Cost_T = Cost(X[m − 1]);
        if Cost_T ≤ Cost_M then
            r = m − 1;
        else
            l = m;
end while
Return X[l];
```

Fig. 4 shows an example. In the figure, Cell A is connected to Net 1 ($\{A, C_1, C_2, C_3\}$) and Net 2 ($\{A, C_4, C_5\}$). If the switching activity α_1 of Net 1 is greater than α_2 of Net 2, the optimal location for Cell A is $(x_1, [y_5, y_4])$, where $[y_5, y_4]$ is the range from y_5 to y_4, i.e., any value greater than or equal to y_5 and less than or equal to y_4. If the x-coordinate a_x of Cell A is greater than x_1, but less than x_2, the HPWL of

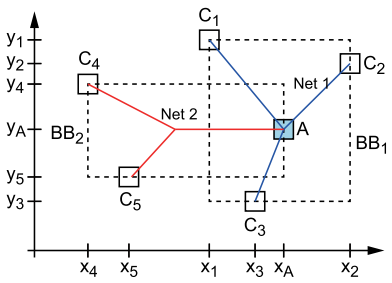

Fig. 4. Two nets and their bounding boxes. Net 1 = $\{A, C_1, C_2, C_3\}$. Net 2 = $\{A, C_4, C_5\}$.

Net 1 is minimal, but the HPWL of Net 2 increases. If a_x is less than x_1, the HPWL of Net 1 increases, but that of Net 2 decreases. Since α_1 is greater than α_2, the total power consumption increases. We can easily verify that $(x_1, [y_5, y_4])$ is the optimal location for Cell A in this way. If α_2 is greater than α_1, however, the optimal location for Cell A is $(x_5, [y_5, y_4])$.

Notice that the proposed algorithm finds an optimal location for each cell. To find optimal locations for all the cells in the netlist, we sequentially choose a cell and move it. Thus, the order of the cells we choose might affect the quality of the solution. In this paper, we sort all the nets in the decreasing order of the switching activity and start from the cells connected to the highest-activity net.

C. Timing Constraints

Optimal locations of some cells found by Theorem 1 might violate the timing constraints explained in Section III. Thus, before we move a cell to its optimal location, we check whether the move will violate the timing constraints or not. If it violates the constraints, we move the cell to the farthest location satisfying the timing constraints from its current location, along and inside the segment connecting the current and the optimal location.

If the timing constraint is the length-based timing constraint, we compute the maximum value of ΔHPWL satisfying inequality (13) for each net connected to the target cell and choose the smallest value among them. The computation of ΔHPWL is performed as follows. We first construct the bounding box of net i connected to the target cell and divide the plane into nine regions as shown in Fig. 5. The target cell X is in the center region ($R5$) and the dotted line shows the bounding of net i. Depending on the optimal location (red circles), we split the line segment connecting the current and optimal locations into multiple segments, one per region (we call the line segment an *optimal segment*). Then, we compute ΔHPWL in each region separately. For example, if the target location is in $R5$, ΔHPWL is 0 even if we move the target cell between its current and the optimal locations. If the target location is in $R2$, however, ΔHPWL is 0 if the target cell is moved inside $R5$, but Δy if it is moved in $R2$, where Δy is the distance between the new location of the target cell in $R2$ and the y-coordinate of the upper horizontal line of the bounding box of net i.

Starting from the region of the optimal location, we compute ΔHPWL satisfying inequality (13). If we find it, it is

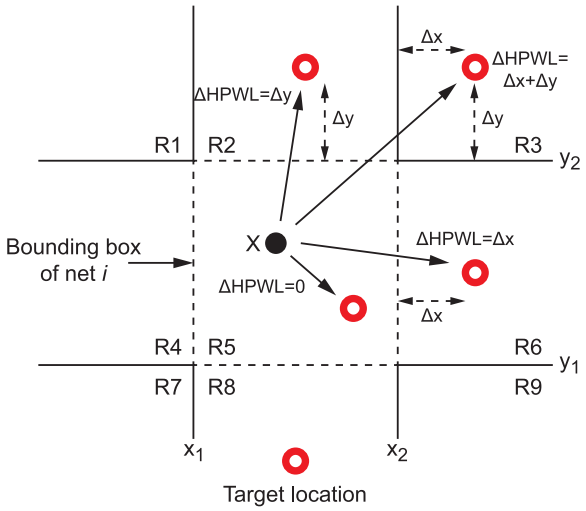

Fig. 5. Consideration of length-based timing constraints.

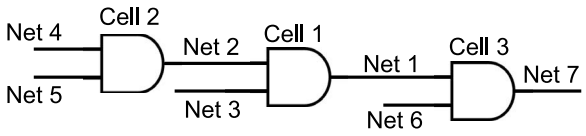

Fig. 6. Delay-based timing constraints.

the optimal location satisfying the length-based timing constraint for net i. If we do not find it, however, we move to the next farthest region along the optimal segment and compute ΔHPWL again. We repeat this process until we find ΔHPWL satisfying the inequality. Notice that the maximum number of regions we should consider for each net is three, which occurs when the optimal location is in $R1, R3, R7$, or $R9$ in Fig. 5.

If the timing constraint is the delay-based timing constraint, we should consider not only each pair of the two adjacent nets connected to the target cell, but also the cells adjacent to the target cell. Fig. 6 shows an example. Suppose Cell 1 in the figure is the target cell. If Cell 1 is moved, the lengths of Net 1, Net 2, and Net 3 are changed. In this case, we apply the delay-based timing constraint to the following net pairs, (Net 1, Net 2) and (Net 1, Net 3). However, the changes of the lengths of Net 1, Net 2, and Net 3 also affect the delay values from Net 4 to Net 2, from Net 5 to Net 2, and from Net 1 to Net 7. Thus, we should also check the delay-based timing constraint to the following net pairs, (Net 4, Net 2), (Net 5, Net 2), and (Net 1, Net 7).

To apply the delay-based timing constraint to a pair of two nets, Net 1 and Net 2, we first split the plane into max. Twenty five regions[2] and compute $\Delta HPWL_1$ and $\Delta HPWL_2$ in each region over which the optimal segment belongs to. Starting from the region of the optimal location, we compute $\Delta HPWL_1$ and $\Delta HPWL_2$ minimizing λ and satisfying inequality (12). If we do not find a value satisfying the inequality in the region, we proceed to the next farthest region on the optimal segment in a similar way to the length-based timing constraint case.

[2]Four x- and four y-coordinates from the eight extremal points of the two bounding boxes split the plane into 25 regions.

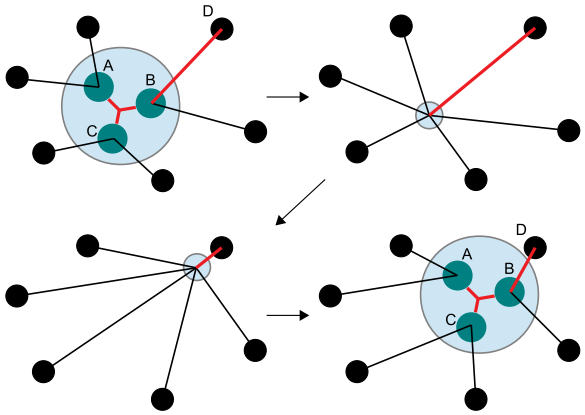

Fig. 7. Illustration of the clustering technique. The red nets are high-activity nets.

D. Density Constraints

As mentioned in Section III, moving a cell to its optimal location might increase the density of the layout area around the optimal location. Thus, we need to control the layout density efficiently during optimization. In this paper, we predetermine a bin size, obtain the maximum bin density in a given layout, and limit the density of each bin to be at most the maximum bin density. If moving a cell to its optimal location violates the density constraint of the bin, we find the next farthest bin that does not violate the density segment along the optimal segment.

We satisfy the timing and density constraints for each move by considering both at the same time. Thus, we guarantee that we never violate the timing and density constraints during/after the optimization.

E. Clustering

A problem we found in moving a cell individually to its optimal location is that we cannot move any cell connected to a high-activity net in some cases. For example, moving Cell *A*, Cell *B*, and Cell *C* toward Cell *D* in Fig. 7 will reduce the dynamic power consumption, but moving the three cells one by one is prohibited because moving each one of them increases the dynamic power consumption or leads to no power benefit. Thus, we cluster the cells connected to high-activity nets and move the cells simultaneously to reduce dynamic power consumption further in addition to moving each cell individually. However, we only cluster the cells connected to a net whose HPWL is less than a predetermined threshold value (K). In fact, there exist some uncertainties in our optimization methodology because our wirelength and power computation are based on the HPWL. Thus, if K is large, the uncertainty goes up and the final power value does not accurately match our power computation. If K is too small, however, there exists just a few clusters that can be moved by the clustering technique. Therefore, we empirically determined K based on simulations. Similar to the cell-based optimization, the solution quality of the clustering-based optimization depends on the order of the clusters we move. In this paper, we sort the nets in the decreasing order of their switching activities and start from the highest-activity net. For each selected net, we cluster the cells connected to the net into a single super-cell and apply the cell-based optimization algorithm to the super cell. After we move the super cell, we flatten the super cell.

F. Complexity Analysis

The cell-based optimization finds an optimal location for each cell and moves the cell to the location. Suppose a target cell is connected to maximum n nets, each of which connects maximum c cells. Then, the complexity of finding two bounding boxes (one with the cell included and the other without the cell) for each net is $\mathcal{O}(c)$ and that of finding all bounding boxes of the target cell is $\mathcal{O}(cn)$. The runtimes for the linear sweeping from $-\infty$ to ∞ for the x-coordinates of the bounding boxes is $\mathcal{O}(n)$. Thus, so the complexity of finding an optimal location for each target cell is $\mathcal{O}(cn)$, which is practically $\mathcal{O}(1)$ because c and n are bounded for most of the cells and nets. For an optimal location found by the above process, finding an optimal location that satisfies the timing and density constraints takes a constant amount of time, so the complexity of moving all the cells to their optimal locations is $\mathcal{O}(C)$, where C is the total number of cells. The cluster-based optimization also moves a cluster for each net, so practically the complexity of finding an optimal location for a cluster is also $\mathcal{O}(1)$. Thus, the complexity of moving all the clusters to their optimal locations is $\mathcal{O}(N)$, where N is the total number of nets. We iterate the cell- and cluster-based optimizations only a few times, so the overall complexity of the NUSBP algorithm is $\mathcal{O}(N + C)$.

V. Simulation Results

In this section, we present our simulation results and detailed analysis.

A. 3-D IC Design Flow and Simulation Setup

For NUSBP, we iterate the cell- and cluster-based optimization multiple times until the power reduction saturates in the dynamic power optimization step. We use the Nangate [9] 45nm library for the standard cell library, Synopsys Design Compiler for synthesis, and Cadence Encounter for 2-D placement and legalization. We also use Cadence Encounter to obtain the switching activity of each net by propagating a constant activity at the primary pins. We use hMetis [10] for the k-way partitioning to design k-tier monolithic 3-D ICs. To obtain area-balanced placement results, we split a given layout into a grid and sequentially apply hMetis to each bin of size $5 * r$ by $5 * r$, where r is the height of a standard cell row. The bin size for the density check is 20 um by 20 um. All the results we obtained by NUSBP in this section did not violate the delay and density constraints.

B. Comparison of Dynamic Power Consumption in Two-Tier Monolithic 3-D ICs

Table III shows wirelength ($\sum$HPWL) and dynamic power consumption ($\sum \alpha \cdot$HPWL) of 2-D and two-tier monolithic 3-D ICs designed by USBP [denoted by 2-tier uniform (2TU)]

TABLE III
COMPARISON OF 2-D, k-TIER UNIFORM-SCALING-BASED, AND k-TIER NONUNIFORM-SCALING-BASED PLACEMENT RESULTS USING DIFFERENT TIMING CONSTRAINTS (-D: DELAY-BASED, -L: LENGTH-BASED, -LR: LENGTH-BASED WITH RELAXATION). THE VALUES IN PARENTHESES SHOW THE RATIO BETWEEN THE 3-D DESIGNS AND THE 2-D DESIGNS. FP IS THE FOOTPRINT AREA AND RT IS THE RUNTIME

Benchmark	Design	FP(um^2)	HPWL(um)	Power (α·HPWL)	RT (s)	Benchmark	Design	FP(um^2)	HPWL(um)	Power (α·HPWL)	RT (s)
LDPC	2-D	600x600	4.58 (1.000)	3.12 (1.000)	-	DES	2-D	529x527	0.977 (1.000)	0.155 (1.000)	-
	2TU	429x429	3.27 (0.714)	2.22 (0.710)	-		2TU	379x377	0.694 (0.711)	0.110 (0.709)	-
	3TU	353x353	2.68 (0.585)	1.81 (0.580)	-		3TU	312x310	0.569 (0.582)	0.090 (0.581)	-
	4TU	307x307	2.33 (0.508)	1.57 (0.503)	-		4TU	271x270	0.494 (0.506)	0.0781 (0.505)	-
	2TNU-D	429x429	3.17 (0.692)	2.11 (**0.677**)	199.9		2TNU-D	379x377	0.657 (0.673)	0.101 (**0.655**)	314.7
	3TNU-D	353x353	2.60 (0.567)	1.72 (**0.552**)	176.5		3TNU-D	312x310	0.536 (0.549)	0.082 (**0.530**)	253.9
	4TNU-D	353x353	2.26 (0.492)	1.49 (**0.478**)	166.3		4TNU-D	271x270	0.464 (0.475)	0.071 (**0.458**)	222.4
	2TNU-L	429x429	3.16 (0.690)	2.10 (**0.672**)	63.0		2TNU-L	379x377	0.642 (0.657)	0.097 (**0.629**)	123.9
	3TNU-L	353x353	2.59 (0.565)	1.71 (**0.548**)	62.5		3TNU-L	312x310	0.526 (0.538)	0.079 (**0.514**)	117.1
	4TNU-L	353x353	2.25 (0.490)	1.48 (**0.474**)	66.0		4TNU-L	271x270	0.457 (0.468)	0.069 (**0.446**)	116.7
	2TNU-LR	429x429	3.06 (0.667)	1.98 (**0.635**)	120.2		2TNU-LR	379x377	0.636 (0.651)	0.095 (**0.612**)	126.8
	3TNU-LR	353x353	2.50 (0.545)	1.61 (**0.517**)	130.0		3TNU-LR	312x310	0.523 (0.535)	0.078 (**0.504**)	126.7
	4TNU-LR	307x307	2.17 (0.474)	1.40 (**0.448**)	122.8		4TNU-LR	271x270	0.456 (0.467)	0.068 (**0.440**)	128.8
FFT	2-D	1058x1050	6.07 (1.000)	1.18 (1.000)	-	M256	2-D	767x766	9.60 (1.000)	1.90 (1.000)	-
	2TU	753x747	4.30 (0.709)	0.834 (0.708)	-		2TU	547x547	6.80 (0.708)	1.35 (0.709)	-
	3TU	617x612	3.52 (0.581)	0.682 (0.579)	-		3TU	449x448	5.56 (0.579)	1.10 (0.579)	-
	4TU	536x532	3.06 (0.505)	0.591 (0.502)	-		4TU	390x390	4.82 (0.502)	0.954 (0.502)	-
	2TNU-D	753x747	4.15 (0.685)	0.816 (**0.692**)	705.5		2TNU-D	547x547	6.70 (0.698)	1.31 (**0.689**)	392.1
	3TNU-D	617x612	3.41 (0.562)	0.666 (**0.565**)	642.7		3TNU-D	449x448	5.47 (0.569)	1.06 (**0.559**)	517.6
	4TNU-D	536x532	2.97 (0.490)	0.577 (**0.490**)	609.4		4TNU-D	390x390	4.74 (0.493)	0.919 (**0.483**)	465.8
	2TNU-L	753x747	4.16 (0.686)	0.805 (**0.683**)	262.6		2TNU-L	547x547	6.65 (0.693)	1.29 (**0.678**)	301.3
	3TNU-L	617x612	3.43 (0.565)	0.656 (**0.556**)	267		3TNU-L	449x448	5.44 (0.566)	1.05 (**0.553**)	299.7
	4TNU-L	536x532	2.99 (0.493)	0.567 (**0.481**)	260.1		4TNU-L	390x390	4.71 (0.491)	0.910 (**0.479**)	320.8
	2TNU-LR	753x747	4.26 (0.701)	0.779 (**0.661**)	462.9		2TNU-LR	547x547	6.64 (0.691)	1.28 (**0.674**)	313.9
	3TNU-LR	617x612	3.50 (0.577)	0.635 (**0.539**)	466.4		3TNU-LR	449x448	5.43 (0.566)	1.05 (**0.551**)	311.1
	4TNU-LR	536x532	3.05 (0.503)	0.500 (**0.467**)	467.9		4TNU-LR	390x390	4.71 (0.491)	0.908 (**0.478**)	318.7

and NUSBP [denoted by 2-tier nonuniform (2TNU)] with different timing constraints. As the table shows, the USBP algorithm reduces the dynamic power consumption by roughly 29% compared to the 2-D placement result and the NUSBP algorithm reduces the dynamic power consumption by 31% to 35%, 32% to 37%, and 33% to 39% for the delay-based (-D), length-based (-L), and length-based with relaxation (-LR) timing constraints, respectively, compared to the 2-D placement result. In addition, the NUSBP algorithm constantly outperforms the USBP algorithm for all the benchmarks by 5% to 14%, 4% to 11%, and 3% to 8% for the -D, -L, and -LR cases, respectively.

For more detailed analysis, we show the difference between the power consumption of 2TU and 2TNU-L for each net in Fig. 8(a). We group all nets into each switching activity bin of width 0.001, compute the sum of the dynamic power of the nets in each bin for 2TU and 2TNU-L, and plot the differences. In the figure, we observe that the power reduction comes primarily from the power reduction in high-activity nets, i.e., many high-activity ($\alpha > 0.8$) nets in 2TNU-L are shorter than in 2TU. Thus, we reduce the dynamic power consumption by shortening the high-activity nets. However, some low-activity nets ($\alpha \sim 0.5$) in 2TNU-L have higher power consumption than those in 2TU. This is unavoidable because the further power reduction in NUSBP is due to making high-activity nets shorter and low-activity nets longer. Fig. 8(b) shows the difference between the HPWL of 2TU and 2TNU-L for each net. In the figure, we observe that the trend of the wirelength difference is similar to the trend of the power reduction shown in Fig. 8(a). In other words, the high-activity nets are shortened at a cost of the elongated low-activity nets.

Regarding the timing constraints, the length-based timing constraint reduces the power consumption more effectively than the delay-based timing constraint as shown in Table III. The reason is mainly because the delay-based timing constraint is tighter than the length-based timing constraint. The following analysis shows the reason. Without loss of generality, suppose α_1 is greater than α_2 in Fig. 2. The original lengths of Net 1 and Net 2 are L_1 and L_2, respectively. Since α_1 is greater than α_2, we move Cell 2 toward Cell 1 by δ. In the worst case, the length of Net 1 after nonuniform scaling becomes $(L_1 - \delta)/\sqrt{N_T}$ and that of Net 2 becomes $(L_2 + \delta)/\sqrt{N_T}$. Substituting these new lengths to inequality (12) gives the following:

$$\left(\frac{L_1 - \delta}{\sqrt{N_T}}\right)^2 + \left(\frac{L_2 + \delta}{\sqrt{N_T}}\right)^2 \leq {L_1}^2 + {L_2}^2 \tag{18}$$

$$\delta_{\text{MAX},d} = \frac{L_1 - L_2 + \sqrt{(L_1 - L_2)^2 + 2(N_T - 1)\left({L_1}^2 + {L_2}^2\right)}}{2} \tag{19}$$

which is the maximum value of δ satisfying the delay-based timing constraint. On the other hand, substituting the length of Net 2 to inequality (12) gives the following:

$$\delta_{\text{MAX},l} = \left(\sqrt{N_T} - 1\right)L_2 \tag{20}$$

which is the maximum value of δ satisfying the length-based timing constraint.

If L_1 is much longer than L_2, $\delta_{\text{MAX},d}$ is greater than $\delta_{\text{MAX},l}$, so the delay-based timing constraint is looser than the length-based timing constraint. If L_1 is much shorter than L_2, however, $\delta_{\text{MAX},l}$ is greater than $\delta_{\text{MAX},d}$, so the length-based timing constraint is looser. Although the length-based timing constraint led to lower power consumption than the delay-based timing constraint in the benchmarks we used, the delay-based timing constraint could lead to lower power consumption if the former case (L_1 is much longer than L_2) dominates the design.

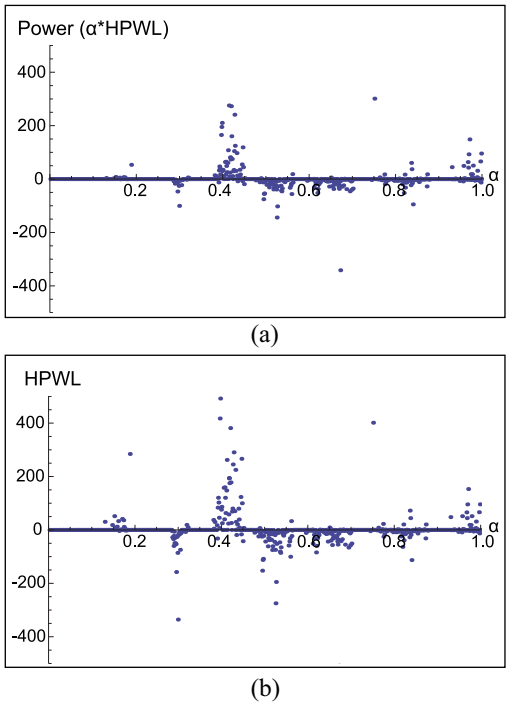

Fig. 8. Comparison of (a) dynamic power consumption and (b) HPWL between 2TNU-L and 2TU. The *x*-axis is the net activity. Benchmark: LDPC.

C. Comparison of Dynamic Power Consumption in Multitier Monolithic 3-D ICs

Table III also shows that the multitier monolithic 3-D ICs reduce the dynamic power consumption more effectively than the two-tier monolithic 3-D ICs. The USBP algorithm outperforms the 2-D layout by 29%, 42%, and 50% in the two-, three-, and four-tier designs, respectively. In addition, the NUSBP algorithm outperforms the USBP algorithm by 2% to 14%, 2% to 13%, and 2% to 13% for the two-, three-, and four-tier designs, respectively. Although the dynamic power consumption monotonically decreases as the tier count goes up, the decrement also reduces. Thus, the dynamic power reduction will eventually saturate even if more tiers are stacked. This is due to the saturation in the wirelength reduction as shown in the same table and other previous work [5]. Since the amount of wirelength reduction is proportional to the scaling ratio $(1/\sqrt{N_T})$, wirelength reduction saturates, which is also translated into the saturation of the dynamic power reduction. However, the NUSBP algorithm still outperforms the USBP algorithm constantly in the multitier monolithic 3-D designs.

Regarding the runtime, applying the delay-based timing constraint takes the highest runtime to optimize the designs. The reason is because the delay-based timing constraint is applied not only to the pairs of the nets connected to a target cell, but also to some pairs of the nets connected to the cells adjacent to the target cell. Thus, applying the delay-based timing constraint requires more computation time than the other timing constraints. On the other hand, the length-based timing constraint with relaxation requires more computations than the length-based timing constraint because the former is tighter than the latter, so the latter searches more optimal points than the former.

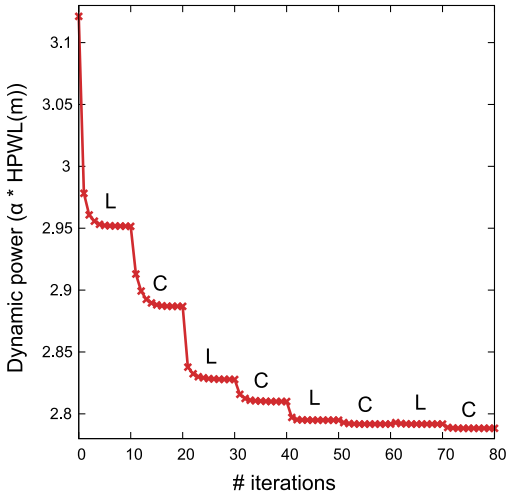

Fig. 9. Variation of the dynamic power consumption of the two-tier LDPC design with cell- and cluster-based optimization interleaving (L: cell-based, and C: cluster-based).

D. Effect of Interleaving Cell- and Cluster-Based Optimizations

If we apply only the cell- or cluster-based optimization repeatedly, the dynamic power consumption decreases at the beginning, but saturates eventually. As explained in Section IV-E, therefore, we alternate between the cell- and cluster-based optimizations to escape from local minima and reduce the dynamic power consumption further. In the NUSBP design flow, we run *m* iterations of the cell-based optimization, then run *n* iterations of the cluster-based optimization and repeat the cell- and cluster-based optimization until the difference between the dynamic power consumption values before and after the optimization is less than a predetermined number. We set both *m* and *n* to 10 in our simulation.

Fig. 9 shows the variation of the dynamic power consumption of the two-tier LDPC design, where *L* and *C* denote the cell- and cluster-based optimizations, respectively. As the figure shows, the dynamic power reduction saturates after four to six iterations in each optimization mode. Once the first cell-based optimization saturates, the cluster-based optimization reduces the dynamic power consumption further by moving multiple cells connected to high-activity nets at the same time. The dynamic power reduction in the cluster-based optimization, however, also saturates. The cluster-based optimization perturbs the placement, so switching back to the cell-based optimization helps reduce the power consumption again. As the number of iterations increases, the power consumption eventually saturates as shown in Fig. 9.

E. Impact of the Density Control

As explained in Section IV-D, we control the density of each bin in the layout during both the cell- and cluster-based dynamic power optimization. Since it is not straightforward to estimate the routability of a design at the placement stage,

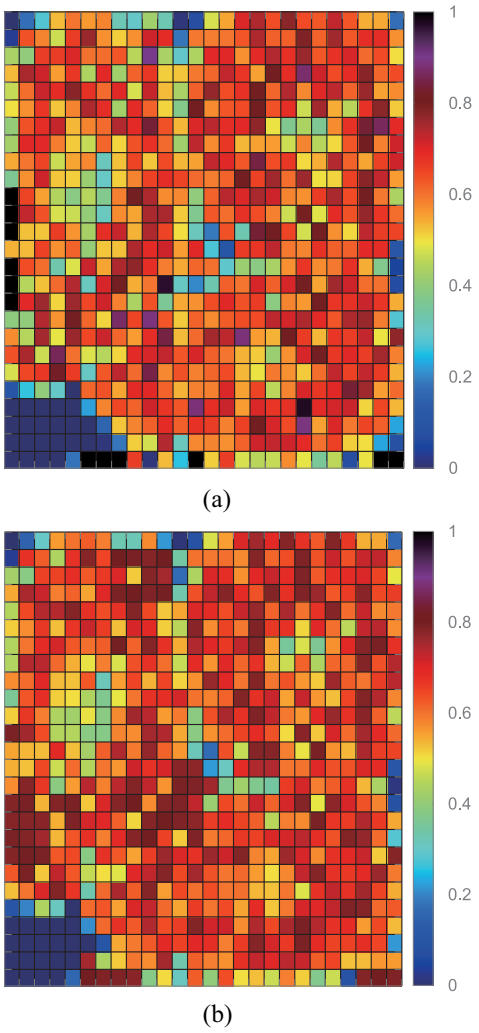

Fig. 10. Density distribution of the DES benchmark. (a) Without density constraints and (b) With density constraints.

many placement papers use some variants of density estimation and control schemes to effectively alleviate potential routing congestions. Fig. 10 shows the bin density distribution maps for the DES benchmark without and with the density constraint and control. The density constraint for each bin in the design is set to 0.8. As shown in the figure, optimizing the design without the density constraint violates the density constraint and the utilization of some of the bins is greater than 1.0. With the density control, however, no bin violates the density constraint.

VI. Conclusion

In this paper, we proposed an NUSBP algorithm for dynamic power optimization in multitier gate-level monolithic 3-D ICs. The algorithm finds an optimal location minimizing the sum of the dynamic power consumption of the nets connected to the cell for each cell without violating the timing and density constraints. The simulation results show that the algorithm outperforms the USBP algorithm by an average of 2% to 14% for two- to four-tier designs.

References

[1] P. Batude *et al.*, "Advances in 3D CMOS sequential integration," in *Proc. IEEE Int. Electron Devices Meeting*, Baltimore, MD, USA, Sep. 2009, pp. 1–4.
[2] D. Henry *et al.*, "Via first technology development based on high aspect ratio trenches filled with doped polysilicon," in *Proc. IEEE Electron. Compon. Technol. Conf.*, Reno, NV, USA, May 2007, pp. 830–835.
[3] J. U. Knickerbocker *et al.*, "Three-dimensional silicon integration," *IBM J. Res. Develop.*, vol. 52, no. 6, pp. 553–569, Nov. 2008.
[4] S. Panth, K. Samadi, Y. Du, and S. K. Lim, "Design and CAD methodologies for low power gate-level monolithic 3D ICs," in *Proc. Int. Symp. Low Power Electron. Design*, Aug. 2014, pp. 171–176.
[5] D. H. Kim, S. Mukhopadhyay, and S. K. Lim, "TSV-aware interconnect distribution models for prediction of delay and power consumption of 3-D stacked ICs," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 33, no. 9, pp. 1384–1395, Sep. 2014.
[6] S. Bobba *et al.*, "CELONCEL: Effective design technique for 3-D monolithic integration targeting high performance integrated circuits," in *Proc. Asia South Pac. Design Autom. Conf.*, Yokohama, Japan, Jan. 2011, pp. 336–343.
[7] Y.-J. Lee and S. K. Lim, "Ultrahigh density logic designs using monolithic 3-D integration," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 32, no. 12, pp. 1892–1905, Dec. 2013.
[8] S. Panth, K. Samadi, Y. Du, and S. K. Lim, "High-density integration of functional modules using monolithic 3D-IC technology," in *Proc. Asia South Pac. Design Autom. Conf.*, Yokohama, Japan, Jan. 2013, pp. 681–686.
[9] (2011). *Nangate 45nm Open Cell Library*, Nangate, Santa Clara, CA, USA. [Online]. Available: http://www.nangate.com
[10] G. Karypis and V. Kumar. *hMETIS, A Hypergraph Partitioning Package Version 1.5.3*. Accessed on Nov. 15, 2014. [Online]. Available: http://glaros.dtc.umn.edu/gkhome/metis/hmetis/download

Sheng-En David Lin (S'16) received the B.S. degree in electrical engineering from Washington State University, Pullman, WA, USA, in 2014, where he is currently pursuing the Ph.D. degree with the Department of Electrical Engineering and Computer Science.

His current research interests include modeling for very large scale integration (VLSI) circuits and systems and algorithms for VLSI CAD automation with current focus on designing of monolithic 3-D ICs.

Dae Hyun Kim (S'08–M'12) received the B.S. degree in electrical engineering from Seoul National University, Seoul, South Korea, in 2002, and the M.S. and Ph.D. degrees in electrical and computer engineering from the Georgia Institute of Technology, Atlanta, GA, USA, in 2007 and 2012, respectively.

He is an Assistant Professor with the School of Electrical Engineering and Computer Science, Washington State University, Pullman, WA, USA. He researched on physical layout optimization with Cadence Design Systems, Inc., San Jose, CA, USA, from 2012 to 2014. His current research interests include electronic design automation and computer-aided design for very large scale integration (VLSI), high-performance and/or low-power VLSI and computer systems, and 3-D integrated circuits and systems.

Dr. Kim was a recipient of the Cadence Excellence in Innovation Award in 2014, the Defense Advanced Research Projects Agency Young Faculty Award in 2016, and the EECS Early Career Award from the School of Electrical Engineering and Computer Science at Washington State University in 2017.