Voltage-Island Partitioning and Floorplanning Under Timing Constraints

Wan-Ping Lee, Hung-Yi Liu, and Yao-Wen Chang, *Member, IEEE*

***Abstract*—Power consumption is a crucial concern in nanometer chip design. Researchers have shown that multiple supply voltage (MSV) is an effective method for power consumption reduction. The underlying idea behind MSV is the tradeoff between power saving and performance. In this paper, we present an effective voltage-assignment technique based on dynamic programming. For circuits without reconvergent fan-outs, an optimal solution for the voltage assignment is guaranteed; for circuits with reconvergent fan-outs, a near-optimal solution is obtained. We then generate a level shifter for each net that connects two blocks in different voltage domains and perform power-network-aware floorplanning for the MSV design. Experimental results show that our floorplanner is very effective in optimizing power consumption under timing constraints.**

***Index Terms*—Floorplanning, layout, low power, multiple supply voltage (MSV), physical design.**

I. Introduction

As THE CMOS technology enters the nanometer era, power dissipation is a key challenge in nanometer chip design. Power consumption generally breaks down into two sources, dynamic and static power. While static power in modern technology mainly comes from leakage current, dynamic power $P_{\rm switch}$ is incurred from a device's switching activities. It can be computed by

$$P_{\rm switch} = k \cdot C_{\rm load} \cdot V_{\rm dd}^2 \cdot f \tag{1}$$

where k is the switching rate, $C_{\rm load}$ is the load capacitance, $V_{\rm dd}$ is the supply voltage, and f is the clock frequency. Compared with static power, dynamic power often dominates the total power consumption in high-frequency circuit design.

In a VLSI design, power consumption and performance optimizations often conflict with each other. Minimizing power consumption and simultaneously satisfying the performance constraint is a challenging problem. Researchers have proposed many low-supply-voltage approaches, among which multiple supply voltage (MSV) [23] is a popular technique for power consumption reduction. The underlying idea behind MSV is the tradeoff between the power saving and performance. Under the performance constraints, it is desired to assign cells along noncritical paths with lower power supply voltages for power saving. Thus, the timing slack available on noncritical paths can be effectively converted to power saving.

There are two major categories of existing algorithms for the VDD assignment, clustered voltage scaling (CVS) [23] and extended CVS (ECVS) [24]. Both algorithms assign appropriate supply voltages to gates by traversing a combinational circuit from the primary outputs (POs) to the primary inputs (PIs) in a levelized order. CVS dose not allow low VDD (VDDL) gates to drive high VDD (VDDH) gates. Relaxing this restriction, ECVS uses *level shifters* for VDDL gates to drive VDDH ones. As a result, ECVS can provide an appreciably larger power reduction compared with CVS. For example, Kulkarni *et al.* [18] recently presented a heuristic based on ECVS for power saving. In addition to CVS and ECVS, Chang and Pedram [7], [8] applied dynamic programming for voltage assignment. In physical design, Wu *et al.* [26] minimized the number of voltage islands after placement. (Each voltage island is composed of cells/blocks with the same supply voltage.) They focused on the minimization of the number of voltage islands but did not consider the constraint imposed by the architecture of the power/ground (P/G) network. To generate a good physical topology for MSV, Ma and Young [20] partitioned voltage islands and assigned voltage levels during floorplanning. In Ma and Young's work, the voltage-level choices are independent of timing effects; any voltage-level choice can satisfy the timing constraint. In other words, there is no tradeoff between power saving and performance.

Although MSV techniques have been studied extensively, there are some deficiencies in the previous works.

1) None of those previous works considers the physical positions of *level shifters*, which is essential for voltage conversion between two circuit components operated at different supply voltages [21]. An inferior level-shifter placement may worsen the timing, and thereby, the timing constraint might be violated.

Manuscript received May 30, 2008; revised September 15, 2008 and December 6, 2008. Current version published April 22, 2009. This work was supported in part by Etron, by SpringSoft, by TSMC, and by the NSC of Taiwan under Grants NSC 96-2628-E-002-248-MY3, NSC 96-2628-E-002-249-MY3, NSC 96-2221-E-002-245, and NSC 96-2752-E-002-008-PAE. An earlier version of this paper was presented at the 2006 and 2007 IEEE/ACM International Conference on Computer-Aided Design in November 2006 and 2007 [12], [13]. This paper was recommended by Associate Editor I. Markov.

W.-P. Lee is with the Graduate Institute of Electronics Engineering, National Taiwan University, Taipei 106, Taiwan, and also with the Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA 15213 USA (e-mail: planet@eda.ee.ntu.edu.tw).

H.-Y. Liu is with the Graduate Institute of Electronics Engineering, National Taiwan University, Taipei 106, Taiwan, and also with Taiwan Semiconductor Manufacturing Company Ltd., Hsinchu 300, Taiwan (e-mail: dianel@eda.ee.ntu.edu.tw).

Y.-W. Chang is with the Department of Electrical Engineering and the Graduate Institute of Electronics Engineering, National Taiwan University, Taipei 106, Taiwan (e-mail: ywchang@cc.ee.ntu.edu.tw).

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.2009.2013997

2) None of those previous works considers the power-network routing resources which makes the MSV design more complicated.

To tackle the more practical MSV problem that considers level-shifter positions and power-network routing resources, we present an effective voltage-assignment technique and perform power-network-aware floorplanning for the MSV design. The proposed floorplanner facilitates the power-network synthesis, which is usually complicated in MSV designs, with a reasonable area overhead. Our main contributions are summarized as follows.

1) We develop a voltage-assignment technique based on *dynamic programming* [5] to handle the voltage-assignment problem. The proposed method is inspired by the delay-constrained technology mapping [6], [14] with enhanced techniques to handle the effects of level shifters. Compared with the previous heuristic voltage-assignment methods [18], [23], [24], our voltage-assignment technique can obtain the optimal solution when the circuit is reconvergent-fan-out free.
2) We propose a new model to estimate the power-network routing resources at the floorplanning stage. The new model estimates the power-network routing resources based on the half-perimeter wirelength (HPWL) of the enclosing rectangle of a voltage island. Compared with the traditional model that estimates the power-network routing resources by the *area* of the enclosing rectangle of a voltage island, empirical results show that our model is more accurate.
3) We present an MSV floorplanner which places the circuit blocks and level-shifter blocks simultaneously while considering the power-network routing resources. To facilitate the power-network synthesis, we consider power-network routing resources during floorplanning. To the best knowledge of the authors, this is the first work that considers the power-network routing resources in the floorplanning stage.
4) Experimental results show the effectiveness of our proposed algorithm in power optimization under timing constraints. Satisfying the timing constraint, for example, it reduces the power-network routing resources by 17.58% on average with a reasonable overhead of 2.76% in area.

The remainder of this paper is organized as follows. Section II introduces level shifters and reviews the B^*-tree floorplanning representation. Section III gives the formulation of voltage-island partitioning and power-network-aware floorplanning. Section IV presents the algorithm flow to solve the addressed problem. Section V proves the optimality of the proposed voltage-assignment algorithm. Section VI reports the experimental results. Finally, Section VII concludes this paper.

II. Preliminaries

This section gives the preliminaries on level shifters and floorplanning techniques. Level shifters, to be introduced in Section II-A, are circuits that handle the magnitude and timing differences between different voltage domains; they are essential components in the MSV design. Furthermore, since we address the MSV design in the floorplan stage, we shall also describe the underlying floorplanning techniques in Section II-B.

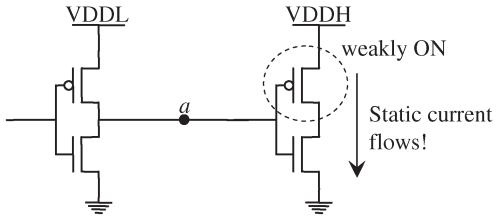


Fig. 1. Direct connection of lower and higher supply-voltage circuits. Static current flows through the PMOS of the higher supply-voltage circuit. A typical approach to blocking the static current is to insert a level-shifter circuit at the position of node a.

A. Level Shifters

A level shifter is an essential circuit to avoid static-current flow caused by nets which are from a lower supply-voltage domain to a higher supply-voltage one. Usami *et al.* in [24] pointed out a serious problem of the static-current flow due to the direct connection of circuits with different supply voltages, as shown in Fig. 1. In which, the static current flows through the circuit on the right-hand side because of the voltage difference between the source and the gate, causing much power consumption and possibly function failures. A typical approach to blocking the static current is to insert a level-shifter circuit at the position of node a. A number of level shifters have been developed. There are two kinds of level shifters introduced in [21]. One requires both high and lower supply voltages, and the other requires only higher supply voltages, which eliminates the constraints that level shifters must be placed on the boundary of voltage islands. In this paper, we consider the latter case because it results in a higher flexibility for floorplanning.

B. Floorplanning Techniques

We adopt the simulated annealing (SA) algorithm [16] with the B^*-tree representation [2] for floorplan optimization. The SA algorithm is a randomized combinatorial optimization technique which simulates the equilibrium states in a physical system. During the cooling process, SA randomly perturbs floorplans and uses a cost function to evaluate the quality of each floorplan. By iterative improvement, SA often can converge to a desired floorplan according to the cost function. Furthermore, SA adopts nonzero probabilities for uphill moves (i.e., hill climbing) to escape from a local optimum.

To efficiently perturb floorplans in SA, we adopt the B^*-tree to represent a floorplan because of its well-proven nice properties for modern floorplan designs. A B^*-tree is an ordered binary tree representing a compacted floorplan, in which every block can no longer be moved to the left and bottom. As shown in Fig. 2, each node of the B^*-tree corresponds to a block of a compacted floorplan. The root of a B^*-tree corresponds to the block on the bottom–left corner. The left child of the node n represents the lowest adjacent block on the right-hand side of b, while the right child of n represents the first block above b with the same horizontal coordinate.

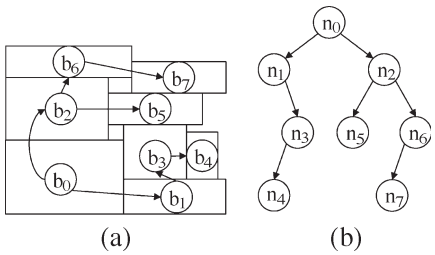


Fig. 2. (a) Compacted floorplan. (b) B*-tree representing the compacted floorplan.

Given a B*-tree, we can calculate the position of each block by a preorder tree traversal. Suppose each block b_f, represented by a node n_i, has the bottom–left coordinate (x_i, y_i), the width w_i, and the height h_i. Then, for the left child n_j of n_i, $x_j = x_i + w_i$; for the right child n_k of n_i, $x_k = x_i$. In addition, we can maintain a contour structure to calculate the y-coordinates for all blocks. Thus, starting from the root node, whose bottom–left coordinate is (0, 0), then visiting the root's left subtree, and then its right subtree, this preorder tree traversal procedure, a.k.a. B*-tree packing, calculates all coordinates of blocks in a floorplan. Using a doubly linked list to implement the contour structure, the total packing time is linear to the number of blocks, which achieves the lower bound complexity for packing.

III. Problem Formulation

We formulate a netlist as a directed acyclic graph (DAG). A vertex represents a PI, a PO, or a block, while an edge denotes a net.

Given k choices of supply voltages $VDDj$, $1 \le j \le k$, an n-vertex DAG $G = (V, E)$ and delay d_i for each vertex $v_i \in V$, $d_i \in \{d_i^1, d_i^2, \ldots, d_i^k\}$, where d_i^j denotes the delay of a vertex v_i operated at the jth voltage domain $VDDj$, according to static timing analysis (STA), the arrival time a_i and the required time r_i of v_i are derived as follows:

$$a_i = \begin{cases} \max_{v_j \in FI_i} a_j, & FI_i \neq \phi \\ 0, & FI_i = \phi \end{cases} \tag{2}$$

$$r_i = \begin{cases} \min_{v_j \in FO_i} a_j - d_i, & FO_i \neq \phi \\ T_{\text{cycle}}, & FO_i = \phi \end{cases} \tag{3}$$

where FI_i and FO_i are sets of the fan-in and fan-out vertices of v_i, respectively, and T_{cycle} is the clock cycle time of the netlist. Using the STA model, we define the static-timing constraint as follows.

Definition 1—(Static-Timing Constraint): Given a clock cycle time and a DAG $G = (V, E)$, corresponding to a netlist, the static-timing constraint of the netlist is $a_i \le r_i, \forall v_i \in V$, where a_i and r_i are given in (2) and (3).

For nanometer VLSI design, the interconnect delay typically dominates the circuit performance. However, STA cannot model the interconnect delay without physical information. In the floorplanning stage, since block positions are determined (and so is wirelength), we can further estimate the timing more accurately. Based on the STA result, we transform the timing slack of each block b into wirelength for more efficient estimation [10]. Such a wire-delay estimation method is often used during floorplanning [22], [25]. (Note that, even with a nonlinear delay model, the wire delay of a net can still be modeled as a function of its wirelength.) The length upper bound o_i of the net, whose source is b_i, is derived from the following linear normalization:

$$o_i = \zeta \cdot s_i = \zeta \cdot (r_i - a_i) \tag{4}$$

where s_i is the slack of block b_i and ζ is a constant to scale timing to wirelength. Since wire and gate loading would also affect the wire delay and further affect the wirelength calculated for the floorplanning, ζ would be set with considering the wire- and gate-loading effects.

Definition 2—(Floorplan-Timing Constraint): A floorplan satisfies the floorplan-timing constraint if and only if, for each net whose source is block b_i, the net length is less than or equal to o_i, as derived in (4).

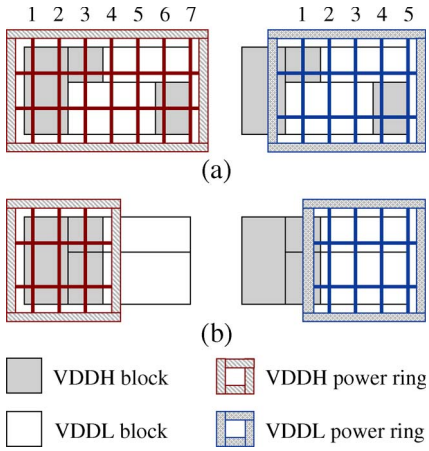


Fig. 3. Example dual-voltage floorplan with a uniform power mesh. The power-network routing-resource requirement of (b) is smaller (requires fewer P/G lines), and thus, (b) is a better floorplan in terms of the power-network routing-resource requirement.

Another important cost metric in the MSV design is the power-network routing-resource requirement. As shown in Fig. 3, the floorplan in Fig. 3(a) needs more P/G lines than that in Fig. 3(b). In practical designs, a P/G mesh is synthesized in a uniform pitch. Therefore, even lower power blocks inside a higher power ring would be masked by higher power lines and vice versa. This is the reason why the vertical power lines 4 and 5 in the left side of Fig. 3(a) are still needed. It should be noted that this power-network model is one of the popular ways to synthesize power networks for MSV design. This model was also adopted by Chen *et al.* [9] and Usami *et al.* [24], and Kulkarni and Sylvester [17] further proposed a technique to deal with the power distribution based on this model. Accordingly, we propose the cost metric, *the power-network routing-resource requirement*, as follows.

Definition 3—(Power-Network Routing-Resource Requirement): Given a floorplan of a set of blocks $B = B_1 \cup B_1 \cup \cdots \cup B_k$, $B_i \cap B_j = \phi$, $i \neq j$, where B_i is the set of blocks operated at voltage $VDDi$, the power-network routing-resource requirement of the floorplan equals $\sum_{i=1}^{k} u_i$, where u_i is the HPWL of the bounding box enclosing B_i.

We observe that the proposed HPWL-based power-network routing-resource estimation is better than the area-based one. A

uniform power mesh for each voltage island B_i consists of a power ring and power lines inside the power ring. Therefore, the power-network routing resource is the total metal area of the power ring and the power lines, given in the following:

$$\Phi_{\mathrm{PNR}} = \sigma_r + \sigma_l \tag{5}$$

where σ_r and σ_l are the total metal areas of the power ring and the power lines, respectively. Suppose that the widths of metal lines used for the power ring and for the power lines are the same. We have

$$\sigma_r = \omega_r \cdot u_i \tag{6}$$

$$\sigma_l = \omega_l \cdot w_i/p_i + \omega_l \cdot h_i/p_i = \omega_l \cdot u_i/p_i \tag{7}$$

where ω_r and ω_l are the widths of metal lines for the power ring and for the power lines, u_i, w_i, and h_i are the HPWL, the width, and the height of the bounding box enclosing island B_i, respectively, and p_i is the pitch of the power lines. Aside from the aforementioned observation, we also justify our HPWL-based model by empirical results in Section VI-A.

According to Definition 3, the power-network routing-resource requirement of the floorplan in Fig. 3(a) is greater than that in Fig. 3(b) since both bounding boxes of VDDH and VDDL blocks in the floorplan of Fig. 3(a) are larger than those of Fig. 3(b). Consequently, the floorplan in Fig. 3(b) is more desirable.

However, a floorplan satisfying the static- and floorplan-timing constraints, consuming low power, and requiring modest power-network routing resources may have an undesirable shape, e.g., all blocks are in a row. Therefore, we need a fixed-outline constraint to control the shape of the floorplan. Furthermore, fixed-outline floorplanning is more popular for modern VLSI designs [3], [15].

Definition 4—(Fixed-Outline Constraint): Given a fixed outline (W^*, H^*) of a desired rectangular bounding box, where W^* (H^*) is the width (height) of the box, every block of a floorplan must be placed inside the bounding box.

Based on the aforementioned definitions, the problem addressed in this paper is formulated as follows.

Definition 5—(The MVF Problem): Given MSV choices, a set of blocks, a netlist, static-timing and fixed-outline constraints, and a constant ζ to scale timing to wirelength, assign each block with a supply voltage and its coordinate in a floorplan so that the power consumption and the power-network routing-resource requirement are minimized and the static-timing, floorplan-timing, and fixed-outline constraints are satisfied.

Note that we also intend to minimize the number of voltage islands. As pointed out in [26], there are significant overheads in voltage shifting devices and implementation costs for a fragmented voltage island. As a result, we allow only one voltage for an island to consider the number of voltage islands.

IV. Algorithm

Fig. 4 shows our flow for solving the multivoltage floorplanning (MVF) problem. The flow consists of three phases: I) voltage assignment; II) level-shifter (block) insertion; and III) power-network-aware floorplanning. For Phase I, we present a dynamic-programming-based method to solve the voltage-assignment problem. As supply voltages are assigned to the circuit blocks in Phase I, in Phase II, we check whether a net needs a level shifter and insert one as a *soft* block if needed. Finally, in Phase III, we transfer the precomputed slack as the wirelength constraint and perform floorplanning on all blocks including the original circuit blocks and the additional level shifters (soft blocks) to minimize the power-network routing resource. The floorplanning is based on SA [16] using the B*-tree floorplan representation [2]–[4].

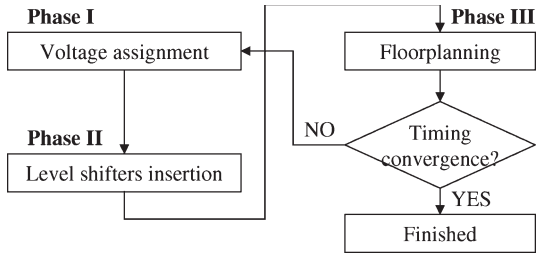


Fig. 4. Algorithm flow for the MVF problem.

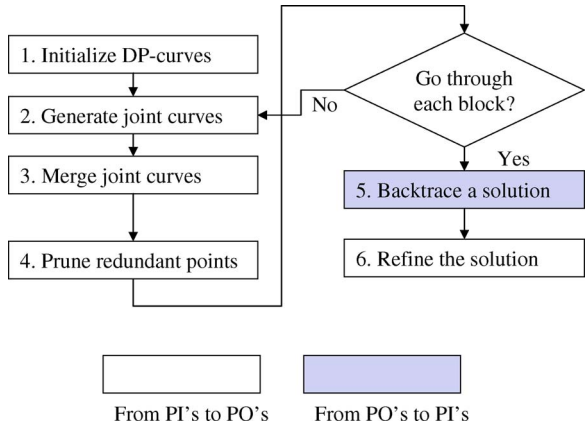


Fig. 5. Six steps of the voltage-assignment algorithm in Phase I.

After the floorplanning, we check if the timing converges. If not, we feed back the current physical information to Phase I and make the timing constraint (T_{cycle}) more stringent to reserve more timing slack for floorplanning. Note that the iteration will eventually terminate; in the worst case, all blocks are assigned the highest supply voltage, and thus, the resulting timing must satisfy the timing constraint (unless the given timing constraint is over constrained, for which no feasible solution is possible).

A. Dynamic Programming for Voltage Assignment

In this section, we present a dynamic-programming approach for supply-voltage assignment, which consists of six steps: 1) delay-power (DP) curve initialization; 2) joint-curve generation; 3) joint-curve merging; 4) redundant point pruning; 5) solution backtracing; and 6) solution refinement (see Fig. 5 for the algorithm flow).

The underlying idea is based on delay-constrained technology mapping [6], [14] with additional considerations for level shifters. Fig. 6 shows the difference between delay-constrained

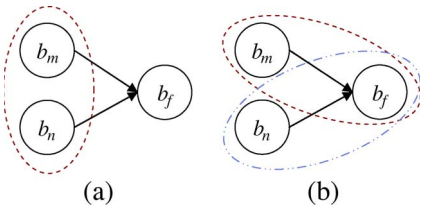


Fig. 6. (a) Traditional delay-constrained technology mapping. It considers all fan-ins of a block first and then propagates the results. (b) Our voltage-assignment algorithm. It considers each b_f's fan-in and b_f and then merges and propagates the results.

Fig. 7. Example DP curve. The three points of the DP curve represent the delay-power characteristics of different supply voltages.

technology mapping and our voltage-assignment algorithm. The traditional delay-constrained technology mapping considers all fan-ins of b_f first and then propagates the results to b_f. In contrast, our voltage-assignment algorithm considers each b_f's fan-in and b_f and then merges and propagates the results.

In our algorithm, given a netlist, we integrate and propagate the DP curves (see Property 1) from PIs to POs and backtrace the final solution from POs to PIs by using dynamic programming. Section IV-A1 defines DP curves and initializes the DP curve for each block. Section IV-A2 generates points of a new joint curve (see Definition 6). Section IV-A3 reviews the lower bound merge operation proposed in [6] and extends it to our algorithm. Section IV-A4 prunes redundant points. Section IV-A5 proposes a backtracing method to find an optimal solution for circuits without reconvergent fan-outs. Finally, Section IV-A6 refines the solution for a circuit that is not reconvergent-fan-out free.

1) DP Curve Initialization: We represent the delay-power characteristics of a block as a *DP curve*. For each block b, a DP curve of b is a power-consumption function of the circuit delay.

2) Property 1: Given a set of candidate supply voltages for a block, the DP curve of the block is a discrete monotonic decreasing power-consumption function of delay.

The property is followed by the natural characteristic of the tradeoff between power saving and performance. To have a smaller delay, a block has to consume more power and vice versa. See Fig. 7 for an example DP curve. Therefore, in the first step, we initialize a DP curve for each block according to its delay and power properties.

3) Joint-Curve Generation: For each block b_j in a topological order, we individually merge the DP curves of each b_j's fan-in block and b_j to derive a *joint curve*, which is formally defined in Definition 6. Therefore, the number of b_j's joint curves depends on the number of its fan-in blocks. Take Fig. 8 for example. Block b_f has two fan-in blocks b_m and b_n, and thereby, two joint curves J_m and J_n would be generated for b_f after the joint-curve generation.

The points in each joint curve are generated by all the combinations of the points from the DP curves of each b_f's fan-in block and b_f. See (8) and (9) for example. The combination is generated by points m_h and f_k from the respective DP curves of b_m and b_f. Note that the delay and power overheads of level shifters are considered in the following:

$$\delta_{hk} = \delta_h + \delta_k + x_{hk} \cdot \delta_s \tag{8}$$

$$\rho_{hk} = \rho_h + \rho_k + x_{hk} \cdot \rho_s \cdot \omega_{hk} \tag{9}$$

where $\delta_h(\rho_h)$ is the delay (power) of point m_h, $\delta_k(\rho_k)$ is the delay (power) of point f_k, δ_s (ρ_s) is the delay (power) of a level shifter, x_{hk} is a 0–1 variable indicating whether a level shifter is needed from point m_h to point f_k, and ω_{hk} is the connection width (in bits) between blocks b_m and b_f. Here, $x_{hk} = 1$ if the supply voltage of point m_h is lower than that of point f_k; it is zero if otherwise. From the equations, we know that, if b_m and b_f have heavy connections, a large number, which is equal to the connection width, of level shifters would be required when the supply voltage of b_m is lower than that of b_f.

Definition 6—(Joint Curve): Suppose that block b_m is a fan-in block of block b_f. The joint curve J_m for b_f is derived by merging the DP curves of b_m and b_f using (8) and (9) and thus consists of points (δ_{hk}, ρ_{hk}).

Fig. 8 shows an example of the joint-curve generation. Block b_f has two fan-in blocks b_m and b_n. Suppose that b_f, b_m, and b_n have three candidate supply voltages. After the DP curve initialization, therefore, there are three points in the DP curves of b_f, b_m, and b_n, respectively. Then, according to Definition 6, joint curves, which are denoted by J_m and J_n, for b_f would be generated, and moreover, the points in each joint curve would be classified according to b_f's supply voltages. Hence, it can be seen that J_m and J_n are classified into three clusters, denoted by C_1, C_2, and C_3, because b_f has three candidate supply voltages. The reason to classify points in a joint curve is for further applying the lower bound merge operation (see Section IV-A3 for details).

Now, we show how to derive the points in the joint curves, using point i_{21} in joint curve J_m in Fig. 8 for example. Since i_{21} comes from the combination of points m_2 and f_1, it is classified into cluster C_1. Moreover, according to (8) and (9) and assuming that the delay and power of a level shifter are both two units and the connection width is one, then we have $i_{21} = (\delta_{21}, \rho_{21}) = (7, 10)$

$$\delta_{21} = 4 + 1 + 1 \cdot 2 = 7 \tag{10}$$

$$\rho_{21} = 3 + 5 + 1 \cdot 2 \cdot 1 = 10. \tag{11}$$

Merging b_n's DP curve with b_f's in the same way results in the points j_{hk}'s.

4) Joint-Curve Merging: As mentioned before, the number of joint curves of a block depends on the number of its fan-in blocks, and thus, the number of joint curves of a block may be more than one. Therefore, we propose the joint-curve merging to merge those joint curves.

Our joint-curve merging extends the *lower bound merge operation* proposed by Chaudhary and Pedram [6], with an

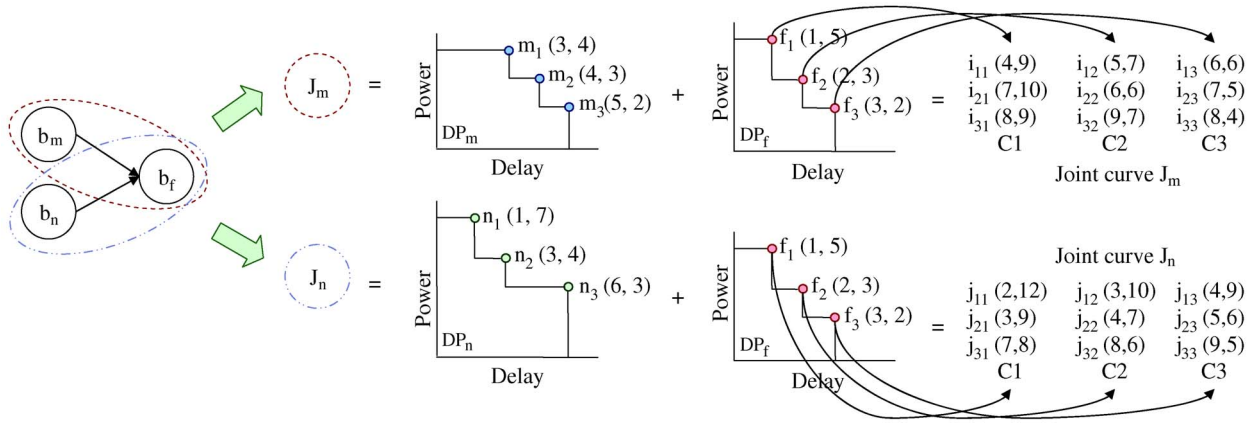


Fig. 8. b_m and b_n are two fan-in blocks of b_f, and two joint curves J_m and J_n are generated for b_f. The points in J_m and J_n are calculated by (8) and (9), and moreover, these points are divided into three clusters $C1$, $C2$, and $C3$ because of b_f's three available supply voltages. Due to the space limitation, J_m and J_n are represented by text.

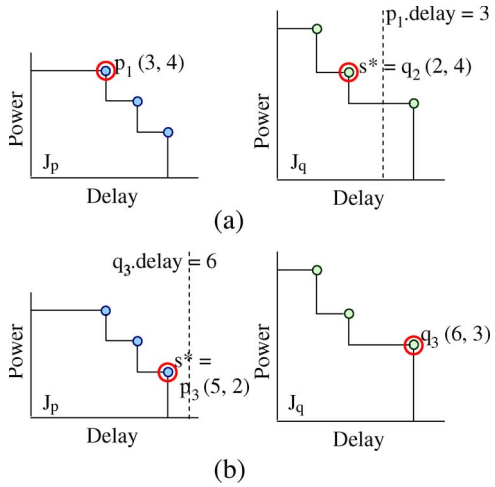


Fig. 9. Example of s^* point defined in Definition 7. (a) The s^* point of p_1 is q_2. (b) The s^* point of q_3 is p_3.

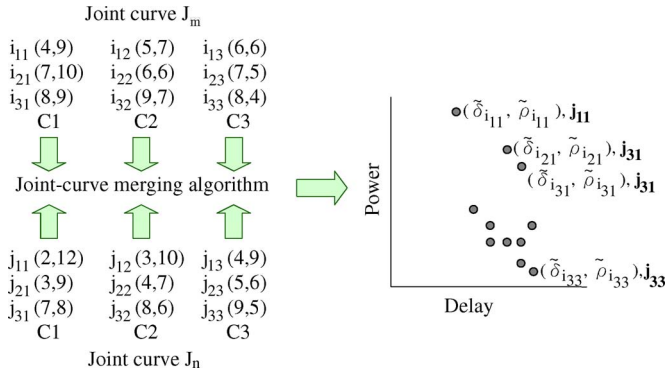


Fig. 10. Example of our joint-curve merging. J_m and J_n are merged by (12) and (13) point by point and cluster by cluster. The resulting points are shown in the right-hand side, in which the boldface characters outside the parentheses give the s^* points in J_n. For example, point $(\tilde{\delta}_{i_{11}}, \tilde{\rho}_{i_{11}})$ is the resulting point by merging i_{11} and j_{11}, which is the s^* point of i_{11}.

additional consideration of level shifters. In our joint-curve merging, a point in a joint curve would find a corresponding point in other joint curves for merging as a merged point, and these merged points are the objective of the joint-curve merging. Precisely, our joint-curve merging is a process to find an s^* point, defined in Definition 7, for merging. Again, note that a point p would find an s^* point in each joint curve except the one containing point p.

Definition 7—(s^* Point): Given two joint curves J_p and J_q, point p in J_p can find *at most one* point q in J_q, such that q's delay is the closest to yet less than p's delay, and meanwhile, q's power is the minimum. The point q is defined as the s^* point of point p.

As shown in Fig. 9(a), q_2 is the s^* point of p_1 because the delay of q_2 is the closest to yet less than that of p_1, and meanwhile, the power of q_2 is the minimum. In other words, under p_1's delay bound, we find a point q_2 in J_q such that the sum of p_1's and q_2's powers is the minimum. The joint-curve merging can be summarized as follows.

Definition 8—(Joint-Curve Merging): Given a set of joint curves $\{J_1, J_2, \ldots, J_n\}$, merge these joint curves using (12) and (13), point by point and cluster by cluster, to obtain points $(\tilde{\delta}_p, \tilde{\rho}_p)$'s. Here

$$\tilde{\delta}_p = \delta_p \tag{12}$$

$$\tilde{\rho}_p = \rho_p + \Sigma_{i \neq x} \rho_{s_i^*} \tag{13}$$

where δ_p (ρ_p) is the delay (power) of point p and $\rho_{s_i^*}$ is the power of the s^* point of p in every joint curve J_i, except the one, denoted by J_x, containing p.

Fig. 10 shows an example of our joint-curve merging. J_m and J_n, shown in the left-hand side, are two joint curves individually consisting of three clusters; the points, shown in the right-hand side, are obtained by applying our joint-curve merging. Take point $(\tilde{\delta}_{i_{11}}, \tilde{\rho}_{i_{11}})$ for example. i_{11} finds the point j_{11} as its s^* point, and according to (12) and (13), we have $(\tilde{\delta}_{i_{11}}, \tilde{\rho}_{i_{11}}) = (4, 21)$

$$\tilde{\delta}_{i_{11}} = \delta_{i_{11}} = 4 \tag{14}$$

$$\tilde{\rho}_{i_{11}} = \rho_{i_{11}} + \rho_{j_{11}} = 9 + 12 = 21. \tag{15}$$

It should be noted that both i_{11} and j_{11} are in cluster $C1$.

Here, it is clearer that the reason to split the joint curves into three clusters is for the correctness of our joint-curve merging.

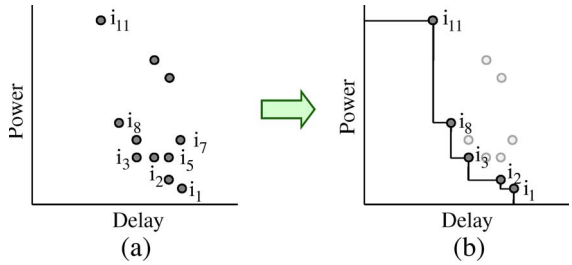


Fig. 11. Two processes of redundant-point pruning: sorting and pruning. (a) Sort all points by their y-coordinates. i_j represents that this point is the jth lowest one in the curve. (b) The final pruned result; there are five nonredundant points.

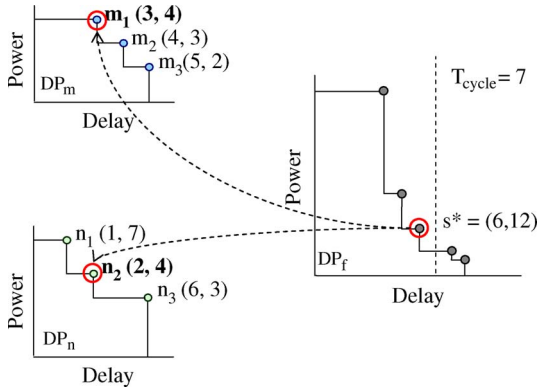


Fig. 12. Backtracing procedure for finding an optimal solution. According to T_{cycle}, we identify the best resulting s^* point in the DP curves of POs. Then, we backtrace an optimal solution of each block.

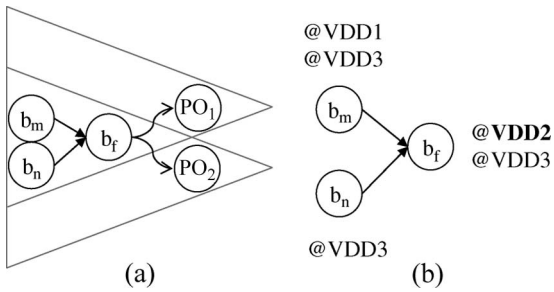


Fig. 13. According to Definition 10, b_f is the common block in PO_1's and PO_2's fan-in cones. (a) PO_1 and PO_2 share some blocks, as in the overlapping portion. (b) After backtracing a solution, these overlapped blocks may be set in several different voltages. Assign the highest one to the common block b_f and then run the voltage-assignment algorithm again to find a better solution.

If we apply the joint-curve merging to $C1$ of J_m and $C2$ of J_n, the resulting point comes from different supply-voltage points in the DP curve of b_f. However, we should make the supply voltage of the point in b_f's DP curve the same for every merging; otherwise, an error situation occurs.

Theorem 1: Given a set of n joint curves $\{J_1, J_2, \ldots, J_n\}$ and a set P_i of points for each joint curve J_i, $i = 1, 2, \ldots, n$, the time complexity of joint-curve merging is $O(n^2 \cdot X)$, where $X = \max_{1 \le i \le n} |P_i|$.

Proof: The time complexity depends on the point-merging calculation, and it is dominated by (13). Point p in a joint curve should find an s^* point for merging in every other joint curve. Since there are n joint curves in total, it needs $n-1$ iterations for p to find its s^* points. Moreover, there are at most $n \cdot X$ points in those joint curves; therefore, the overall time complexity is $O(n^2 \cdot X)$. ■

5) Redundant-Point Pruning: After generating the points for a merged curve, a monotonic decreasing DP curve should be and can be constructed from the merged curve by a line-sweeping algorithm. The line-sweeping algorithm consists of two steps: sorting and pruning. First, sort all points by the y-coordinate from the smallest to the largest, and if the points have the same y-coordinate, sort the points by their x-coordinates from the smallest to the largest, using Fig. 11(a) for example. i_1 is in front of i_2 because i_1 has the smaller y-coordinate than i_2; i_3 is in front of i_5 because i_3 and i_5 have the same y-coordinate, but i_3 has the smaller x-coordinate than i_5. In this figure, point i_j means that the point is the jth lowest one in a DP curve.

Definition 9—(Point Dominance): In a DP curve, a point i dominates another point j if and only if $i.x \le j.x$ and $i.y < j.y$, where $i.x$ and $i.y$ denote the x- and y-coordinates of i, respectively.

After sorting, we prune the points which are dominated by other points. Since the points have been sorted by their y-coordinates, a point i is in front of another point j if $i.y \le j.y$, e.g., i_1 is in front of i_2. Thus, if $i.x \le j.x$, j is dominated by i. More precisely, checking the x- and y-coordinates of a point and its previous one is enough to find all dominated points; it takes a linear time. Fig. 11 shows the process of the monotonic decreasing chain generation.

To reduce the number of level shifters, if two points have the same delay and power, the one with fewer level shifters would be chosen, and the one with more level shifters would be pruned. Note that the number of induced level shifters depends on the connection width between two blocks.

TABLE I
INFORMATION OF THE DAGS

Ckts	Depth of paths		Pins of blocks	
	Max	Min	Average	Standard deviation
n10	6	2	17.9	4.8
n30	12	2	17.0	6.5
n50	14	2	16.8	7.7
n100	18	2	15.3	5.7
n200	25	2	15.2	6.0
n300	21	2	12.6	5.0

6) Solution Tracing: After initializing DP curves, generating joint curves, merging joint curves, and propagating DP curves from PIs to POs, we need to trace a netlist and get an optimal solution of voltage assignment from POs to PIs. We determine the solution point s^* (see Fig. 12) according to T_{cycle}, and the delay and power of this circuit are decided accordingly. Since our algorithm is based on dynamic programming, we can backtrace solutions (voltage assignment to each block) until we reach PIs.

Theorem 2: Given a netlist without reconvergent fan-outs, an optimal solution for the voltage-assignment problem can be traced in linear time.

Proof: Our algorithm is based on dynamic programming by combining the solutions to subproblems. In other words, each solution to the problem or a subproblem knows how the

TABLE II
AVERAGE AND STANDARD DEVIATION OF DELAY AND POWER

Ckts	Average				Standard deviation			
	Delay (VDDH)	Power (VDDH)	Delay (VDDL)	Power (VDDL)	Delay (VDDH)	Power (VDDH)	Delay (VDDL)	Power (VDDL)
n10	22337.7	21684.1	59566.8	9637.1	10497.1	9533.9	27992.3	4237.2
n30	7063.5	6855.0	18835.7	3046.3	1756.9	1751.5	4685.2	778.5
n50	4027.1	3902.9	10738.6	1734.3	2169.6	2081.5	5785.5	925.0
n100	1799.2	1800.2	4797.5	799.7	896.2	900.2	2389.9	400.1
n200	882.5	888.2	2353.7	394.4	485.5	498.0	1294.6	221.3
n300	913.8	911.8	2436.6	404.9	489.8	487.3	1306.2	216.6
LS	200	200	–	–	0	0	–	–

solution is combined. Partitioning the combination for each solution takes constant time, and we partition the combination once for each block. Hence, the tracing process takes linear time. ■

We will prove the optimality in Section V.

7) Solution Refinement: We first give the following definition (see also Fig. 13).

Definition 10—(Common Block): In a netlist, if there are different timing paths reconverged at the inputs of a block, the block is said to be a *common block* of the paths.

To handle the circuit with reconvergent fan-outs, we resort to a two-pass technique to deal with the voltage-assignment problem. The first pass works in the same way as described in Sections IV-A1–A5. After the first pass, a common block may be assigned several different voltages, since different paths may set the common block in different voltages, shown in Fig. 13. For those voltages, we assign a highest one to the block and then apply dynamic programming from the common block to POs again. The second pass can find a better solution by using more timing budget which is gained from common blocks. Avoiding wasting timing budgets, the second pass is thus needed.

When we set voltages for the common blocks, we also consider the number of level shifters. To reduce the number of level shifters, we let the voltages of common blocks be equal to or higher than the voltages of their fan-out blocks in the cost of larger power consumption.

B. Level-Shifter (Soft Block) Insertion

This is Phase II of our proposed algorithm flow. Level shifters are inserted into a net that connects two blocks in different power domains. Note that the level shifters' delay and power effects have been considered in Phase I when we assign voltages to blocks. In this phase, we insert level shifters as soft blocks for floorplanning. We trace the circuits from PIs to POs to search for the nets that need level shifters by breadth-first search.

We treat level shifters as soft blocks. A soft block in a connection contains all needed level shifters. The number of level shifters in a connection is equal to the number of bits in the connection. Thus, we insert a level-shifter block according to the connection width (in bits). Another issue is that a larger fan-out load needs a larger level shifter to drive it.

C. Power-Network-Aware Floorplanning

The objective in this phase is to find a floorplan which simultaneously minimizes the power-network routing-resource requirement (Definition 3) and satisfies the timing (Definition 2) and the fixed-outline constraints. Hence, we propose a cost function (16) to minimize the power-network routing resource without violating the constraints. Given a B*-tree T representing a floorplan of a set of blocks $B = \{b_1, b_2, \ldots, b_n\}$

$$\Phi(T) = \alpha\Phi_{\mathrm{PNR}} + (1-\alpha)\Phi_{\mathrm{area}} + \Phi_{\mathrm{timing}} + \Phi_{\mathrm{outline}}, \quad 0 \le \alpha \le 1 \tag{16}$$

where Φ_{PNR} is the power-network routing resource of B, Φ_{area} is the area of the floorplan, and α is a weighting factor. Note that the four terms are all normalized to the same scale order in advance.

In addition, suppose that each net i, $1 \le i \le p$, has q fan-out blocks, a fan-in block, and a wirelength upper bound o_i [see (4)]. Let l_{ij} be the HPWL of the bounding box of net i's fan-out and fan-in blocks. Then, the timing violation penalty Φ_{timing} is defined as

$$\Phi_{\mathrm{timing}} = \sum_{i=1}^{p} \max\left(\sum_{j=1}^{q} l_{ij} - o_i, 0\right). \tag{17}$$

Similarly, we give a floorplan the fixed-outline violation penalty Φ_{outline} if the floorplan exceeds the desired fixed outline by

$$\Phi_{\mathrm{outline}} = (R - R^*)^2 \tag{18}$$

where $R^*(R)$ is the aspect ratio of the desired fixed-outline (the current floorplan).

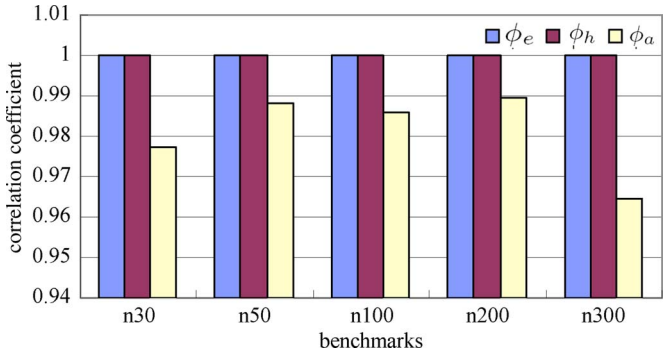


Fig. 14. Comparison between HWPL- and area-based power-network routing-resource estimation. ϕ_e, ϕ_h, and ϕ_a are calculated according to (19)–(21), respectively. The proposed HPWL power-network routing-resource model ϕ_h is closer to the expected power-network routing-resource value and achieves a more accurate estimation.

TABLE III
PHASE I: VOLTAGE-ASSIGNMENT RESULTS USING THE DYNAMIC PROGRAMMING METHOD

	Original Design			Dynamic Programming							
Ckts	Total Power (in VDDH)	Critical blocks	Non-Critical blocks	Total Power (with LS)	Power Saving (%)	#VDDL blocks	#VDDH blocks	#LS blocks	Ratio (VDDL/Non-Critical)	Runtime (sec)	#Rerun iterations
n10	216841	10	0	216841	0	0	10	0	0	0.001	0
n30	205650	12	18	190717	7.26	6	24	57	0.333	0.103	1
n50	195140	29	21	163088	16.43	20	30	51	0.952	62.316	2
n100	180022	34	66	162802	9.57	33	67	78	0.500	713.695	2
n200	177633	42	158	175964	0.94	86	114	326	0.722	1775.227	4
n300	273499	60	240	205928	24.71	135	165	384	0.563	898.829	4

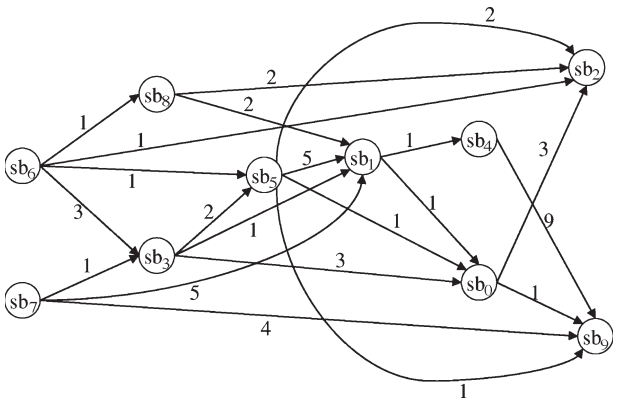


Fig. 15. Corresponding netlist of n10. The numbers on nets are the bits of the nets. Note that the nets connecting blocks to pads are not shown.

TABLE IV
POWER-SAVING COMPARISON BETWEEN OUR VOLTAGE ASSIGNMENT AND (22) PROPOSED BY GUPTA *et al.* [11]

	Total Power with level shifters			Runtime (sec.)	
Ckts	All in VDDH	Ours	Gupta *et al.*'s	Ours	Gupta *et al.*'s
n10	216841	216841	216841	0.001	0.001
n30	205650	190717	197852	0.103	0.001
n50	195140	163088	192396	62.316	1.733
n100	180022	162802	173384	713.695	3.679
n200	177633	175964	177633	1775.227	5.052
n300	273499	205928	255879	898.829	5.698
saving		10.69%	2.79%		

V. OPTIMALITY OF OUR VOLTAGE ASSIGNMENT

Our voltage-assignment algorithm can obtain an optimal solution when a circuit does not have any reconvergent fan-out. In this section, we prove this claim. The voltage-assignment algorithm is based on dynamic programming. Its optimality relies on the properties of optimal substructure and overlapping subproblems [5]. A problem exhibits an optimal substructure if an optimal solution to the problem contains within it optimal solutions to subproblems; when a recursive algorithm revisits the same problem over and over again, the optimization problem has overlapping subproblems [5]. The following two lemmas show these properties for voltage assignment.

Lemma 1: The voltage-assignment problem exhibits an optimal substructure.

Proof: We use the popular *cut-and-paste* technique [5] to characterize the optimal substructure. Suppose that a block b_f has two fan-in blocks b_m and b_n such that the two joint curves J_m and J_n are generated. A point m_i in J_m finds a point n_j in J_n as an s^* point. Equation (13) must give an optimal (lowest) power consumption under the m_i's delay; i.e., $P = \rho_{m_i} + \rho_{n_j}$ is minimum. By contradiction, if there is another point n_x in J_n and the power summation of ρ_{m_i} and ρ_{n_x} is smaller than P, then we could cut n_j and paste n_x to produce a lower power-consumption result, thus contradicting P's optimality. The optimal substructure thus follows. ■

Lemma 2: The voltage-assignment problem has overlapping subproblems.

Proof: Lemma 1 implies that there is at least one subproblem when finding an optimal solution of a block except PIs. Take the same illustration shown in Lemma 1 for example. Each resulting point in the final DP curve of b_f is generated from J_m's and J_n's points, which are recursively generated from b_m's and b_n's fan-ins with the minimum power; as a result, (13) again and again requests for the points with the minimum power. It is obvious that the voltage-assignment algorithm has overlapping subproblems. ■

With Lemmas 1 and 2, we have the following theorem.

Theorem 3: The dynamic-programming-based voltage-assignment algorithm correctly computes the optimal solutions for circuits without reconvergent fan-outs.

VI. EXPERIMENTAL RESULTS

We conducted two experiments. The first experiment, which is presented in Section VI-A, verifies the effectiveness of modeling the power-network routing resource based on the HPWL, while the second one, which is presented in Section VI-B, tests the effectiveness and efficiency of our proposed algorithm.

Our algorithm was implemented in the C++ programming language and executed on a Linux machine with a 3.20-GHz CPU and 2-GB memory. We tested on the GSRC floorplan benchmarks. Since the information in the GSRC benchmark is not sufficient for voltage-island optimization, we need to add some additional information for the experiment. For each test case, it was carried out in the following steps.

Step 1) We assign the direction (input/output) for each PAD and each net; then, each GSRC benchmark can be modeled by a DAG. The information of DAGs is shown in Table I. In this table, we list the maximum and minimum depths of paths and the average and standard deviations of the numbers of block pins. Since it is time consuming to search all paths and then calculate their average depth, here, we list the maximum and the minimum depths of paths.

Step 2) After constructing the corresponding DAG, according to the blocks' area, we assign the timing and power consumption for each block. The standard

TABLE V
PHASE III: FLOORPLANNING RESULTS OF A TRADITIONAL A-FP ($\alpha = 0$) AND OUR PN-FP ($\alpha = 0.6$). THE FIXED-OUTLINE CONSTRAINT IS SET TO [800, 800]

Netlist Information					Power-Network Routing Resource		Area		Wirelength		Runtime (sec)	
Ckts	#Net	#VDDL blocks	#VDDH blocks	#LS blocks	A-FP	PN-FP	A-FP	PN-FP	A-FP	PN-FP	A-FP	PN-FP
n10	118	0	10	0	965	965	233024	233024	1729	1729	8	6
n30	406	6	24	57	1650	1369	225379	229289	8184	8202	132	115
n50	512	20	30	83	2056	1372	241443	259206	12453	13780	551	498
n100	668	33	67	78	2124	1643	262539	275331	16316	16731	1401	1489
n200	1613	114	86	326	2193	2048	291563	301095	20149	20288	2962	2797
n300	1905	135	165	384	2785	2306	460753	465439	26190	27241	4587	4195
Average					1962.2	1617.2	285783.5	293897.3	14170.2	14661.8	1606.8	1516.7
Difference (%)					-17.58		+2.76		+3.35		-5.94	

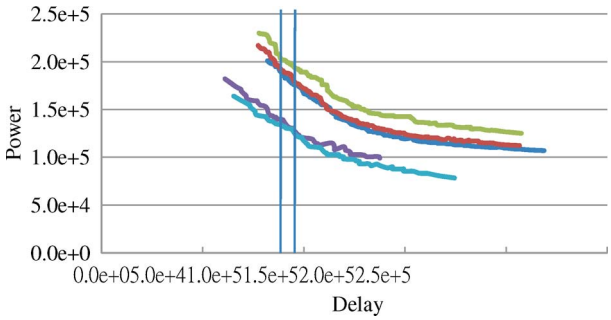


Fig. 16. DP curves of the five most critical blocks in n30. The two vertical lines denote two different timing constraints.

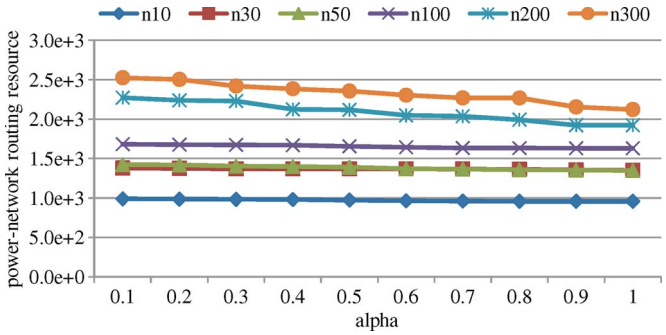


Fig. 17. Negative correlation between α in (16) and power-network routing resources of the resulting floorplans.

deviation and the average of delay and power are shown in Table II.

Step 3) The fixed-outline constraint is set to [800, 800], and the ζ to scale timing to wirelength is set to 0.25. In other words, a 200-unit delay would be incurred for each 50-unit wirelength between two blocks.

A. Results on the Power-Network Routing-Resource Model

In the floorplanning stage, we estimate the power-network routing resource by using the HPWL of the enclosing rectangle of voltage islands. However, the intuitive way to estimate the power-network routing resource is based on the area of voltage islands. Therefore, here, we perform experiments to justify the advantages of our proposed method, HPWL-based power-network routing-resource estimation (please see also our observation in Section III).

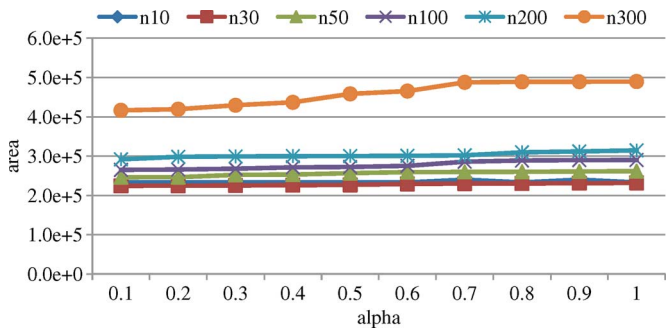


Fig. 18. Positive correlation between α in (16) and areas of the resulting floorplans.

The power-network routing-resource requirement depends on the total area of metal lines required for power networks. Hence, for the 15 best floorplans [in terms of (16)] obtained in our floorplanner, we create a uniform power mesh for each voltage island and then calculate the average total area of metal lines of the power meshes; the obtained value is our expected power-network routing-resource usage, denoted by σ_e. In addition, for the 15 best floorplans, we also calculate the average power-network routing resources σ_h and σ_a, based on the HPWL and the area of voltage islands, respectively. The resulting values shown in Fig. 14 are obtained by

$$\phi_e = \sigma_e/\sigma_e \tag{19}$$

$$\phi_h = \sigma_h/\sigma_e \tag{20}$$

$$\phi_a = \sigma_a/\sigma_e. \tag{21}$$

As it can be seen, ϕ_h, which is the normalized HPWL-based power-network routing-resource estimation, is closer to the expected value than ϕ_a. We therefore claim that the HPWL-based model proposed in this paper is more preferable to evaluate power-network routing resources.

B. MSV Results

We refer to a block to be *critical* if the block will induce a timing violation when its supply voltage is changed to a lower one. Table III shows the voltage-assignment results. There are two factors affecting the experimental results. One is noncritical blocks, and the other is common blocks. It should be noted that noncritical blocks are the blocks in noncritical paths determined by the STA. The third and fourth columns

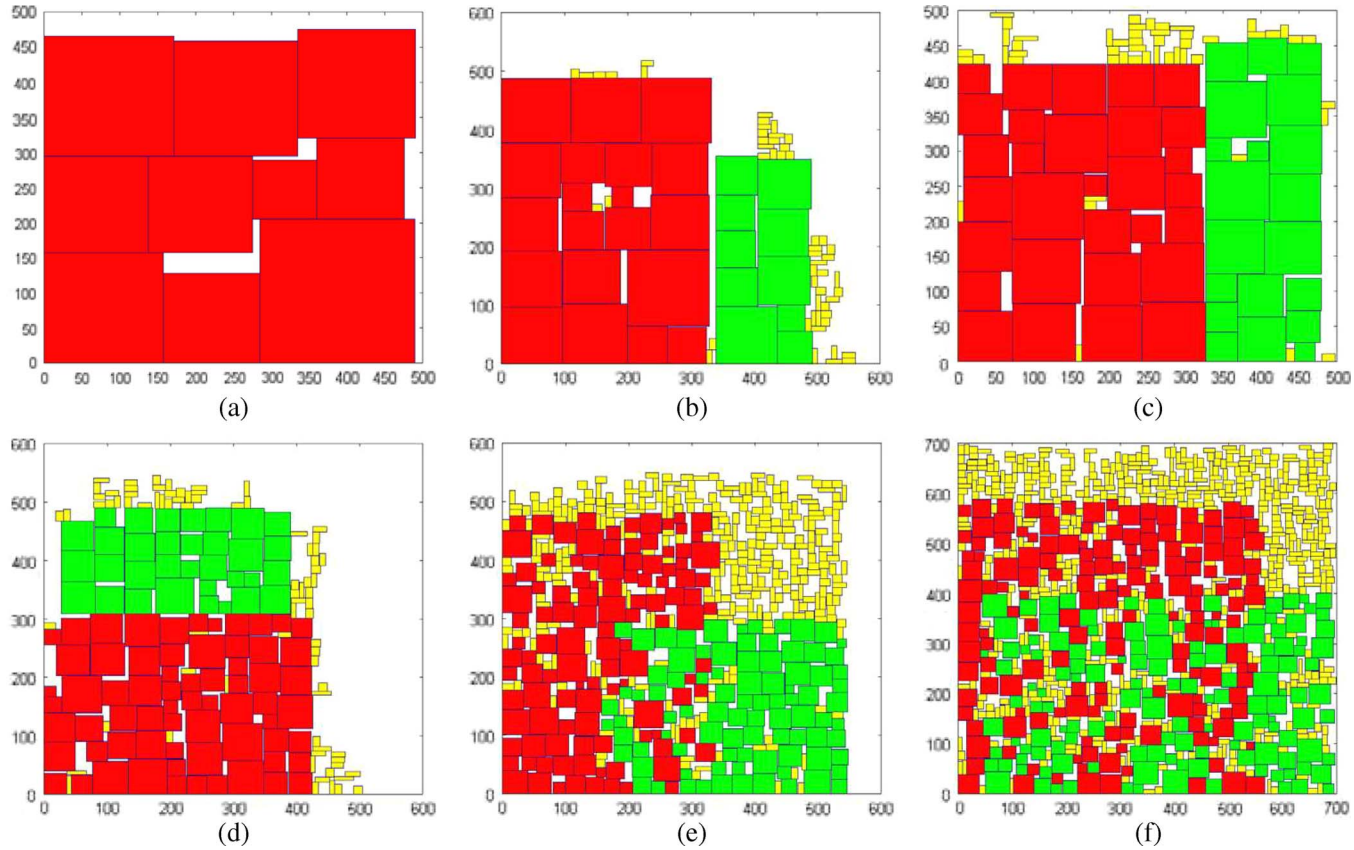


Fig. 19. Power-network-aware floorplans of n10, n30, n50, n100, n200, and n300 are shown in (a) to (f) respectively. VDDH blocks, VDDL blocks, and level shifters are colored in red, green, and yellow, respectively. Level shifters here are assumed to require VDDH only so the power-network routing resource of level shifters is computed together with VDDH blocks.

show the respective number of critical and noncritical blocks in each test case. We find that the ratio of critical blocks to noncritical blocks in n30 is $2:3$ and that in n300 is $1:4$. In a small test case, if the ratio is high, we cannot achieve much power saving. On the other hand, all the test cases have many common blocks. For example, Fig. 15 shows the DAG of n10, in which there are many common blocks in n10. Those common blocks will decrease the power saving (see Section IV-A5). Note that the nets connecting blocks and pads are not shown in Fig. 15.

In the sixth column in Table III, we show the total power saving of each test case; the results show that our algorithm is effective to reduce power consumption by up to 24.71%. We also compare between our method and that of Gupta *et al.* [11]. Although the work of Gupta *et al.* focuses on dual-threshold-voltage assignment (i.e., the work deals with more than the assignment problem for the comparative study here), it is still significant to make the comparison with its core technique since the work of Gupta *et al.* is a state-of-the-art method for voltage assignment. The equation presented in [11] and employed for the comparative study is given as follows:

$$P_p = \frac{l_p - l'_p}{s'_p - s_p} \tag{22}$$

where $l_p(s_p)$ and $l'_p(s'_p)$ denote the original and the final power consumptions (timing slacks) of block b_p before and after scaling down the supply voltage. The block with the maximum sensitivity, denoted by P_p, gets the highest priority to scale down its supply voltage for the dynamic power optimization. Table IV lists the comparison. It can be seen that our method saves 7.9% more power than (22). Furthermore, practical designs' netlists will be simpler than our test cases (more noncritical blocks and fewer common blocks); therefore, we expect that our algorithm will achieve more power saving for practical designs.

The eleventh column of Table III lists the running time of each test case. In which, the running time of n200 is larger than that of n300. At first glance, this result might not be as expected since n200 is smaller than n300. However, the reason is that the DAG of n200 is more complicated than that of n300. In Table I, it can be seen that the blocks in n200 averagely have more pins than those in n300, and the maximum depth of the circuit paths in n200 is longer than that in n300. As shown in Table V, furthermore, the blocks in n200 averagely have more nets $(1842/200 = 9.21)$ than those in n300 $(2231/300 = 7.44)$. As a result, the DAG of n200 is more complicated than that of n300, and thus, it takes more time to handle n200 than n300.

Fig. 16 shows the DP curves of five most timing-critical blocks in n30; each of them is the last block on one of the five most timing-critical paths. Moreover, the two vertical lines represent two different timing constraints; the more stringent the timing constraint, the more timing slack is reserved for delay optimization during floorplanning.

The cost metric employed by the proposed floorplanner is shown in (16). In which, α is a weighting factor for the tradeoff between the power-network routing resource and the area. We also conducted the experiments to explore the impact of different values of the weighting factor α by setting α from 0.1 to 1.0 with the step size of 0.1, as shown in Figs. 17 and 18. A smaller α leads to a smaller weight on the power-network routing resource, but a larger one on the resulting area. Although α affects the resulting power-network routing resource and area, the effects are not significant since the area minimization is just a by-product of minimizing the power-network routing resource. Consequently, we choose $\alpha = 0.6$ for the experiments on the performance of our floorplanner. Although not presented here, the conclusion is the same based on different α's.

Table V shows the effectiveness of our power-network-aware floorplanner (PN-FP, setting α in (16) to 0.6). Compared with a traditional area-aware floorplanner (A-FP, setting α to 0), PN-FP indeed reduces the power-network routing resource by 17.58% with a reasonable overhead of 2.76% more area, on the average. As for timing requirements, both floorplanners produce timing-satisfied floorplans with a negligible difference of total wirelength. Aside from the effectiveness, PN-FP even runs faster than A-FP by 5.94% less runtime. This could result from the fact that, during SA, the cost function simultaneously considering area and the power-network routing resource may have a faster converging rate than that considering area alone. Empirically, PN-FP significantly reduces power-network usage with a slight overhead of area.

Fig. 19 shows all the resulting floorplans. Blocks of the same supply voltage are almost clustered together to reduce the power-network routing resource, while level shifters are spread around to meet the timing constraint. Interestingly, the area of level shifters is much smaller than that of voltage islands, e.g., Fig. 19(b)–(d); the distribution of islands are nearly bipartitioned to reduce the power-network routing resource. Otherwise, the VDDL voltage island would be grouped, surrounded by the VDDH island and level shifters, e.g., Fig. 19(e) and (f), since the level shifters also require VDDH. These experimental results reveal that our PN-FP is very effective.

VII. Conclusion

In this paper, we have proposed a dynamic-programming-based voltage scaling algorithm and a PN-FP for the MSV design. The experimental results have shown that our algorithm is very effective in reducing power (up to 24.71%) and power-network routing resources (17.58%) with a reasonable area overhead of 2.76%.

References

[1] R. K. Brayton, G. D. Hachtel, and A. L. Sangiovanni-Vincenmtelli, "Multilevel logic synthesis," *Proc. IEEE*, vol. 78, no. 2, pp. 264–300, Feb. 1990.

[2] Y.-C. Chang, Y.-W. Chang, G.-M. Wu, and S.-W. Wu, "B*-trees: A new representation for non-slicing floorplans," in *Proc. ACM/IEEE Des. Autom. Conf.*, Los Angeles, CA, Jun. 2000, pp. 458–463.

[3] T.-C. Chen and Y.-W. Chang, "Modern floorplanning based on fast simulated annealing," in *Proc. ACM Int. Symp. Phys. Des.*, San Francisco, CA, Apr. 2005, pp. 104–112.

[4] T.-C. Chen and Y.-W. Chang, "Modern floorplanning based on B*-trees and fast simulated annealing," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 25, no. 4, pp. 637–650, Apr. 2006.

[5] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed. New York: McGraw-Hill, 2001.

[6] K. Chaudhary and M. Pedram, "Computing the area versus delay trade-off curves in technology mapping," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 14, no. 12, pp. 1480–1489, Dec. 1995.

[7] J. Chang and M. Pedram, "Energy minimization using multiple supply voltages," in *Proc. Int. Symp. Low Power Electron. Des.*, Monterey, CA, 1996, pp. 157–162.

[8] J. Chang and M. Pedram, "Energy minimization using multiple supply voltages," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 5, no. 4, pp. 436–443, Dec. 1997.

[9] C. Chen, A. Srivastava, and M. Sarrafzadeh, "On gate level power optimization using dual-supply voltages," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 9, no. 5, pp. 616–629, Oct. 2001.

[10] M.-C. Wu and Y.-W. Chang, "Placement with alignment and performance constraint using the B*-tree representation," in *Proc. IEEE Int. Conf. Comput. Des.*, San Jose, CA, Oct. 2004, pp. 568–571.

[11] P. Gupta, A. B. Kahng, and P. Sharma, "A practical transistor-level dual threshold voltage assignment methodology," in *Proc. IEEE Int. Symp. Quality Electron. Des.*, San Jose, CA, Mar. 2005, pp. 421–426.

[12] W.-P. Lee, H.-Y. Liu, and Y.-W. Chang, "Voltage island aware floorplanning for power and timing optimization," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, San Jose, CA, Nov. 2006, pp. 389–394.

[13] W.-P. Lee, H.-Y. Liu, and Y.-W. Chang, "An ILP algorithm for post-floorplanning voltage-island generation," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, San Jose, CA, Nov. 2007, pp. 605–655.

[14] K. Keutzer, "DAGON: Technology binding and local optimization by DAG matching," in *Proc. IEEE/ACM Des. Autom. Conf.*, Jun. 1987, pp. 341–347.

[15] A. B. Kahng, "Classical floorplanning harmful?" in *Proc. ACM Int. Symp. Phys. Des.*, San Diego, CA, Apr. 2000, pp. 207–213.

[16] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 22, no. 4598, pp. 671–680, May 1983.

[17] S. Kulkarni and D. Sylvester, "Power distribution techniques for dual VDD circuits," in *Proc. IEEE Asia South Pacific Des. Autom. Conf.*, Yokohama, Japan, Jan. 2006, pp. 838–843.

[18] S. H. Kulkarni, A. N. Srivastava, and D. Sylvester, "A new algorithm for improved VDD assignment in low power dual VDD systems," in *Proc. ACM Int. Symp. Lower Power Electron. Des.*, Newport, CA, Aug. 2004, pp. 200–205.

[19] W.-K. Mak and J.-W. Chen, "Voltage island generation under performance requirement for SoC designs," in *Proc. IEEE Asia South Pacific Des. Autom. Conf.*, Yokohama, Japan, Jan. 2007, pp. 798–803.

[20] Q. Ma and E. F. Y. Young, "Voltage island-driven floorplanning," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, San Jose, CA, Nov. 2007, pp. 644–649.

[21] R. Puri, L. Stok, J. Cohn, D. Kung, D. Pan, D. Sylvester, A. Srivastava, and S. Kulkarni, "Pushing ASIC performance in a power envelope," in *Proc. ACM/IEEE Des. Autom. Conf.*, Anaheim, CA, Jun. 2003, pp. 788–793.

[22] X. Tang and D. F. Wong, "Floorplanning with alignment and performance constraints," in *Proc. ACM/IEEE Des. Autom. Conf.*, New Orleans, LA, Jun. 2002, pp. 848–853.

[23] K. Usami and M. Horowitz, "Clustered voltage scaling technique for low-power design," in *Proc. ACM Int. Symp. Lower Power Electron. Des.*, Dana Point, CA, Aug. 1995, pp. 3–8.

[24] K. Usami, M. Igarashi, F. Minami, M. Ishikawa, M. Ichida, and K. Nogami, "Automated low-power technique exploiting multiple supply voltages applied to a media processor," *IEEE Trans. Solid-State Circuits*, vol. 33, no. 3, pp. 463–472, Mar. 1998.

[25] M.-C. Wu and Y.-W. Chang, "Placement with alignment and performance constraints using the B*-tree representation," in *Proc. ACM/IEEE Des. Autom. Conf.*, San Diego, CA, Jun. 2004, pp. 568–571.

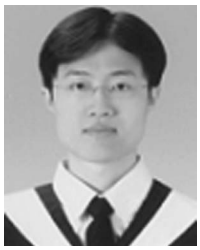
[26] H. Wu, I. M. Liu, M. D. F. Wong, and Y. Wang, "Post-placement voltage island generation under performance requirement," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, San Jose, CA, Nov. 2005, pp. 309–316.

[27] H. Wu, M. Wong, and I.-M. Liu, "Timing-constrained and voltage-island-aware voltage assignment," in *Proc. ACM/IEEE Des. Autom. Conf.*, San Francisco, CA, Jul. 2006, pp. 429–432.

[28] Y.-J. Yeh and S.-Y. Kuo, "An optimization-based low-power voltage scaling technique using multiple supply voltage," in *Proc. IEEE Int. Symp. Circuits Syst.*, Sydney, Australia, May 2001, pp. 535–538.

Wan-Ping Lee received the B.C. degree in management information system from Chung Yuan Christian University, Chungli, Taiwan, in 2001 and the M.S. degree in computer science from National Sun Yat-Sen University, Kaohsiung, Taiwan, in 2004. She is currently working toward the Ph.D. degree in the Graduate Institute of Electronics Engineering, National Taiwan University (NTU), Taipei, Taiwan.

She is also a Visiting Researcher at the Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA. After graduating from NTU, she is going to start her postdoctoral training in bioinformatics in March 2009. Her current research interests focus on multiple-supply-voltage floorplanning and 3-D ICs.



Hung-Yi Liu received the M.S. degree from the Graduate Institute of Electronics Engineering, National Taiwan University (NTU), Taipei, Taiwan, in 2007.

He is currently an Engineer with Taiwan Semiconductor Manufacturing Company Ltd., Hsinchu, Taiwan. He is also with the Graduate Institute of Electronics Engineering, NTU.

Yao-Wen Chang (S'94–A'96–M'96) received the B.S. degree from National Taiwan University (NTU), Taipei, Taiwan, in 1988, and the M.S. and Ph.D. degrees from the University of Texas at Austin in 1993 and 1996, respectively, all in computer science.

He is a Professor in the Department of Electrical Engineering and the Graduate Institute of Electronics Engineering, NTU. He is currently also a Visiting Professor at Waseda University, Kitakyushu, Japan. He was with National Chiao Tung University (NCTU), Hsinchu, Taiwan from 1996 to 2001 and IBM T. J. Watson Research Center in the summer of 1994. His current research interests lie in VLSI physical design, design for manufacturability/reliability, and design automation for biochips. He has been working closely with industry in these areas. He has co-edited one textbook on EDA and coauthored one book on routing and over 150 ACM/IEEE conference/journal papers in these areas.

Dr. Chang is a winner of the 2009 ACM ISPD Clock Tree Synthesis Contest, the 2008 ACM ISPD Global Routing Contest, and the 2006 ACM ISPD Placement Contest. He is a recipient of Best Paper Awards at ICCD-95 and the 2007 and 2008 VLSI Design/CAD symposia, 12 Best Paper Award Nominations from DAC (four times), ICCAD (twice), ISPD (three times), ACM TODAES, ASP-DAC, and ICCD in the past eight years. He has received many research awards, such as the 2007 Outstanding Research Award, the inaugural 2005 First-Class Principal Investigator Award, and the 2004 Dr. Wu Ta You Memorial Award, all from National Science Council of Taiwan, and the 2004 MXIC Young Chair Professorship from the MXIC Corp, and excellent teaching awards from NTU (four times) and NCTU. He is currently an associate editor of IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS (TCAD) and an editor of the *Journal of Information Science and Engineering* (JISE) and the *Journal of Electrical and Computer Engineering* (JECE). He has served on the ICCAD Executive Committee, the ASP-DAC Steering Committee, the ACM/SIGDA Physical Design Technical Committee, the ACM ISPD and IEEE FPT Organizing Committees, and the technical program committees of ASP-DAC, DAC, DATE, FPL, FPT, GLSVLSI, ICCAD, ICCD, IECON, ISPD, SOCC, TENCON, and VLSI-DAT. He is currently an independent board director of Genesys Logic, Inc, a technical consultant of RealTek Semiconductor Corp., a member of board of governors of Taiwan IC Design Society, and a member of the IEEE Circuits and Systems Society, ACM, and ACM/SIGDA.