Post-Floorplanning Power/Ground Ring Synthesis for Multiple-Supply-Voltage Designs *

Wan-Ping Lee
Graduate Institute of Electronics Engineering
National Taiwan University
Taipei 10617, Taiwan
planet@eda.ee.ntu.edu.tw

Diana Marculescu
Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh, PA 15213
dianam@ece.cmu.edu

Yao-Wen Chang
Department of Electrical Engineering
National Taiwan University
Taipei 10617, Taiwan
ywchang@cc.ee.ntu.edu.tw

ABSTRACT

The multiple-supply voltage (MSV) design style has been extensively applied to mitigate dynamic-power consumption. The MSV design paradigm, however, brings many crucial challenges, especially in the power-ring synthesis. Unlike the previous works that form the power rings as the enclosing bounding boxes of voltage islands, we enable power rings alignment to the outer boundaries of voltage islands. With this new formulation, the power-ring estimation becomes more accurate during floorplanning, and the power-ring synthesis is more practical after floorplanning. In this paper, we first propose a linear-time voltage-island power-ring search algorithm to identify the power rings of voltage islands and then present a linear-time optimal power-ring corner-patching algorithm to minimize the number of corners in the power rings by using post-floorplanning whitespaces. Experimental results first demonstrate that reducing corners in power rings significantly mitigates IR drop, and then show that the proposed algorithm can reduce the number of corners by 33% on average for the GSRC floorplan benchmarks. In particular, the *total* running time for the 16 GSRC benchmarks is less than one second on an AMD-64 machine with a 2.2 GHz CPU and 8 GB memory.

Categories and Subject Descriptors

B.7.2 [**Integrated Circuits**]: Design Aids

General Terms

Algorithms, Design, Performance

Keywords

VLSI, Physical Design, Floorplanning, Multiple-Supply Voltage Design

1. INTRODUCTION

Performance-constrained low-power design has been studied extensively in the literature. Among existing techniques, the *multiple-supply voltage* (MSV) design style [13] provides an effective way for dynamic-power reduction. Generally, dynamic-power consumption can be computed by

$$P_{dynamic} = k \cdot C_{load} \cdot V_{dd}^2 \cdot f,$$

where k is the circuit switching rate, C_{load} is the loading capacitance, V_{dd} is the supply voltage, and f is the clock frequency. The dynamic-power consumption is proportional to the square of V_{dd}, implying that lowering supply voltages can effectively reduce dynamic-power consumption.

While dynamic-power consumption is mitigated by the MSV technique, the MSV design style induces several crucial challenges for power-ring synthesis. In this work, to make the MSV designs more flexible and more practical, we allow one or several blocks to form a voltage island; in other words, each supply voltage may have several physical independent voltage islands whose power rings are naturally independent. As a result, this method provides much higher flexibility, but dramatically increases the complexity of the power-ring synthesis.

To reduce the complexity of the power-ring synthesis, previous works defined the power rings as consisting of the bounding boxes of the voltage islands. However, treating the problem this way is perhaps over simplistic; an example is shown in Section 2. Therefore, a more practical methodology for MSV power-ring synthesis is desired. In this paper, we propose a power-ring synthesis methodology that first identifies the power rings that align to the outer boundaries of voltage islands and then adjusts the power rings for fewer corners while fixing the positions of the blocks (at the floorplanning packing or post-floorplanning stages). Furthermore, the proposed methodology is proven to take linear time when employing a hash-table based algorithm. Due to its efficiency, our methodology can be applied to post-floorplan MSV design as well as MSV-aware floorplan synthesis.

MSV systems have been applied at various design stages such as the floorplanning stage [3, 4, 5, 7, 10], the post-floorplanning stage [9], the placement stage [8], or the post-placement stage [1, 14, 15]. Most of these works do not consider power-network synthesis or consider it by rough models, e.g., the bounding-box power-ring formulation. However, such an approach might be over simplistic for power-ring estimation during floorplanning and could be impractical for power-ring synthesis at the post-floorplanning stage. In contrast, our outer-boundary formulation is amenable to being employed in the floorplanning and post-floorplanning stages because of its accuracy and practicality. Section 2 gives an example to compare the two power-ring formulations, the traditional bounding-box formulation and our outer-boundary one.

In addition to the aforementioned novel methodology of power rings for voltage islands, we summarize our major contributions as follows:

- To the best of our knowledge, this is the first work proposing power-ring alignment to the outer boundaries of voltage islands. Compared with previous works that approximate power rings by the bounding boxes of voltage islands, our methodology is more realistic.

*This work was supported in part by TSMC and NSC of Taiwan under Grant No's. NSC 96-2752-E-002-008-PAE, NSC 96-2628-E-002-248-MY3, NSC 96-2628-E-002-249-MY3, and NSC 96-2221-E-002-245.

ISPD'09, March 29–April 1, 2009, San Diego, California, USA.

- A linear-time voltage-island power-ring (contour) search algorithm is presented. When using a hash-table approach, the proposed algorithm takes only linear time on average to search for a power ring.
- A linear-time optimal power-ring corner-patching algorithm is presented. While power rings are aligned to the outer boundaries of voltage islands, there may be multiple corners for each power ring. This implies that a significant number of *vias* would be introduced in the power rings, which is definitely not desirable. Therefore, it is important to minimize the number of corners in the power rings by patching the whitespaces in the given floorplan into the circuit blocks and make power rings more regular. We empirically validate this undesired effect of the corners on IR drops.
- Experimental results demonstrate that reducing corners in power rings significantly mitigates IR drop, and show that the proposed algorithm can reduce the number of corners by 33% on average for the GSRC floorplan benchmarks. The *total* running time for the 16 GSRC circuits is less than one second on an AMD-64 machine with a 2.2 GHz CPU and 8 GB memory.
- Since the proposed algorithms are computationally efficient, they are amenable to being embedded into MSV floorplanners to achieve better MSV floorplans. In addition, they can also be applied for synthesizing the power rings during the post-floorplanning stage.

The remainder of this paper is organized as follows. Section 2 discusses the advantages of the outer-boundary formulation for MSV designs. Section 3 formulates the power-ring synthesis and adjusting problem for MSV designs. Section 4 describes our proposed methodology for the addressed problem. Section 5 shows the experimental results, and Section 6 concludes this paper.

2. COMPARISONS OF POWER-RING FORMULATIONS

We first introduce the bounding-box formulation and our outer-boundary one. Take Figure 1 for example, in which the given floorplan contains five circuit blocks operating at two specific supply voltages, three blocks in the left voltage island and two in the right one, and the interconnection wires among the blocks are shown. Figures 1(a) and (b) employ the bounding-box formulation for power-ring estimation while Figures 1(c) and (d) adopt our outer-boundary formulation. The bounding-box formulation estimates and forms power rings as enclosing bounding boxes of voltage islands, as shown in Figures 1(a) and (b); our outer-boundary formulation, in contrast, searches power rings by aligning the outer boundaries of voltage islands and then makes the power rings more regular by patching whitespace into voltage islands, e.g., w_1 and w_2 as shown in Figure 1(c).

Compared with our outer-boundary formulation, the bounding-box one might underestimate the complexity of the power-ring synthesis during floorplanning and/or post-floorplanning, and thus might mislead MSV-aware floorplanning and/or post-floorplanning optimization. The floorplan in Figure 1(b) is more preferable than that in Figure 1(a) if the bounding-box formulation is employed, because they both have the same cost of the power-ring synthesis, but the floorplan in Figure 1(b) has shorter wirelength. As a matter of fact, however, the cost/complexity of the power-ring synthesis for the floorplan in Figure 1(b) is underestimated due to the overlap between the two voltage islands. The overlap between two voltage islands implies that the power lines for different supply voltages need to be interleaved in the overlapped region, widening the power-line pitch and thus increasing IR drop. In contrast, our outer-boundary formulation can capture the cost of the power-ring synthesis more accurately, as shown in Figures 1(c) and (d). Although the floorplan in Figure 1(d) has shorter wirelength, its power cost (e.g., IR drop) and power-ring complexity could be higher due to the larger number of corners in the power rings. Note that to reduce the number of corners and thus make the power rings more regular, our outer-boundary formulation

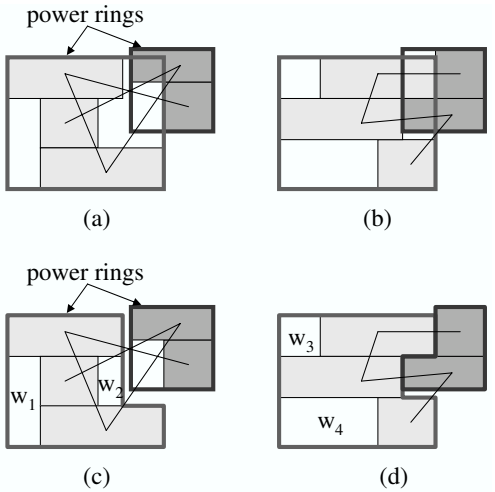

Figure 1: Comparison between the bounding-box and outer-boundary formulations. The given floorplan contains five circuit blocks operating at two specific supply voltages, three blocks in the left voltage island and two in the right one, and the wires interconnecting the blocks are shown. (a) and (b) employ the bounding-box formulation for power-ring evaluation while (c) and (d) adopt the outer-boundary one. In (c) and (d), w_1, w_2, w_3, and w_4 are patched to the voltage islands to make the power rings more regular.

patches whitespaces into voltage islands, as shown in Figure 1(c), which might lead to a better MSV floorplan solution.

In addition, the power meshes for voltage islands in industrial designs are usually not regarded as the enclosing bounding boxes of the voltage islands if there are overlapping regions in the bounding boxes. To avoid widening the power-line pitch and save more power-network resource, the power ring for a voltage island is typically aligned to the outline of the voltage island. Hence, our outer-boundary formulation is more realistic and more suitable for MSV designs than the bounding-box one.

3. PROBLEM FORMULATION

To achieve better power-ring synthesis for MSV designs, our objective is to identify the power rings and then reduce the number of corners for each power ring, while keeping all circuits blocks fixed (during the floorplanning packing or the post-floorplanning stages). In this problem, an MSV floorplan is given, in which circuit blocks occupy specified physical spaces and operate at specified supply voltages, as shown in Figure 2(a).

We formally define a voltage island, a power ring, and a corner as follows, and also illustrate them by the examples shown in Figure 2(b).

DEFINITION 1. **(Voltage Island)** *A voltage island characterized by a supply voltage consists of a set of blocks that operate at the specified supply voltage and are adjacent to at least one other block in the set.*

DEFINITION 2. **(Power Ring of a Voltage Island)** *A power ring for a voltage island is the contour (outer boundary) of the voltage island.*

Since the input is a floorplan consisting of rectangular blocks, a power ring of a voltage island is composed of a set of *vertical* and *horizontal* segments that enclose the associated blocks in the voltage island.

DEFINITION 3. **(Corner of a Power ring)** *A corner in a power ring is a point connecting consecutive vertical and horizontal segments of the power ring.*

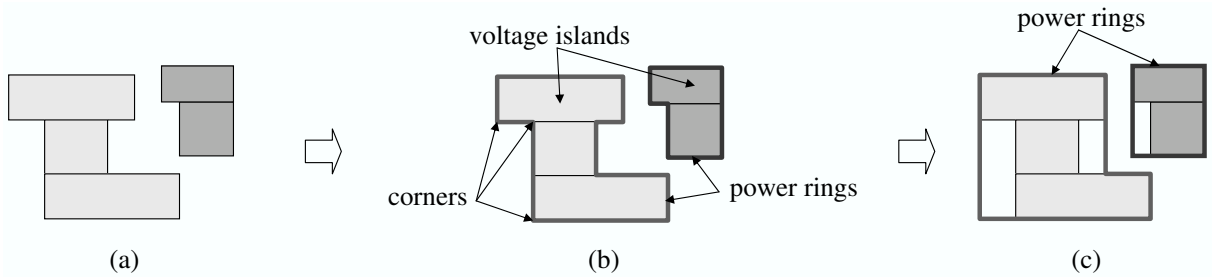

Figure 2: An example problem. (a) Inputs: a floorplan with five blocks operating at specified supply voltages. (b) Definitions: voltage islands, power rings, and corners, as defined in Definitions 1–3. (c) Objective: Identify voltage islands and minimize the number of corners for each power ring. The original numbers of corners of these two power rings are 10 and 6, respectively. After applying the proposed algorithm, the numbers of corners reduce to 6 and 4, respectively.

We define the power-ring synthesis and adjustment problem as follows (see also Figure 2 for an illustration).

PROBLEM 1. **(Problem: Power-Ring Synthesis (PRS))**
Inputs: *a floorplan consisting of rectangular circuit blocks operating at the specified supply voltages*
Objective: *identify the power ring for each voltage island and distribute whitespace to minimize the number of corners of the power rings, i.e.,*

$$\min \quad \Sigma_{i=1}^{n} c_i, \tag{1}$$

where n is the number of power rings and c_i is the number of corners of power ring i.

4. OUR OPTIMAL ALGORITHMS

To deal with the addressed problem, a two-phase framework is proposed. The framework can produce the optimal solution; in other words, the number of corners in each power ring is minimized while all power rings do not overlap with each other (by patching floorplan whitespaces). In the first phase, a voltage-island contour search algorithm is applied to identify voltage islands and their corresponding power rings, as shown in Figure 2(b). In the second phase, a corner-patching algorithm is used to minimize the number of corners by patching whitespaces, as shown in Figure 2(c).

4.1 Voltage-Island Contour Search Algorithm

Given a floorplan consisting of rectangular circuit blocks, we first apply a *depth-first search* (DFS) to identify blocks belonging to each voltage island. Moreover, for each voltage island, we construct two hash tables H_x and H_y corresponding to the x and y coordinates of points defining the blocks, as shown in Figure 3. The first element in H_x links points p_{11} and p_8 since they have the smallest x coordinate; the first element in H_y links points p_1 and p_2 since they have the smallest y coordinate.

Then, given a voltage island, finding its power ring is a contour search problem. Two straightforward, but incorrect methods are illustrated in Figure 4. In Figure 4(a), we search the contour by examining the edge positions. If the edges overlap no other edges, they are assumed to be the contour edges. This method, however, cannot distinguish the edges of inner holes, which neither overlap any edges nor are contour edges of the voltage island; consequently, the method might not produce correct results. As shown in Figure 4(a), the lightly shaded segments are not parts of the contour, but the method might fail to identify them. As shown in Figure 4(b), the second method tries to search the contour by line sweeping. The algorithm tests if an edge is a contour segment when the scanning line sweeps the edge. This method, however, might not correctly determine the contour segments while sweeping an edge. In Figure 4(b), the heavily shaded segments of the sweeping line are parts of contour edges while the lightly shaded one is not. Since these two straightforward methods are incorrect,

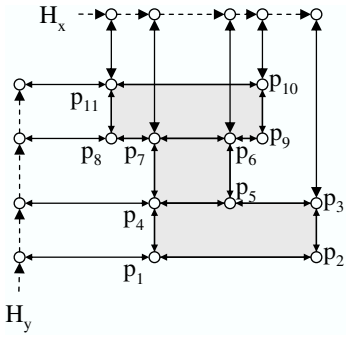

Figure 3: For each voltage island, construct two hash tables H_x and H_y to sort the x and y coordinates of points defining the blocks.

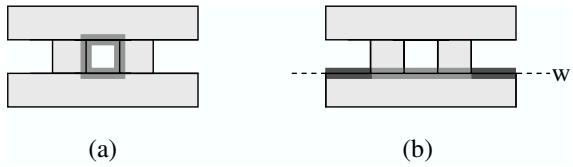

Figure 4: (a) Contour search by examining edge overlap. (b) Contour search by line sweeping. These two methods might not search the contour correctly. The lightly shaded segments might be mistakenly considered as parts of the contour. Here, w indicates a sweeping line.

it is therefore desirable to find a correct and efficient method to tackle the contour search problem. We apply the concept of the contour traversal sequence, inspired by the previous work [11], to develop our algorithm.

4.1.1 Properties of Contour Traversal Sequences

As mentioned before, a contour is composed of a set of vertical and horizontal outer segments. If we trace the respective vertical and horizontal segments counterclockwise from the starting segments with the smallest x and y coordinates, the traversal sequences S_x^* and S_y^* would be obtained. Note that in this paper when we say counterclockwise traversal, we mean the traversal

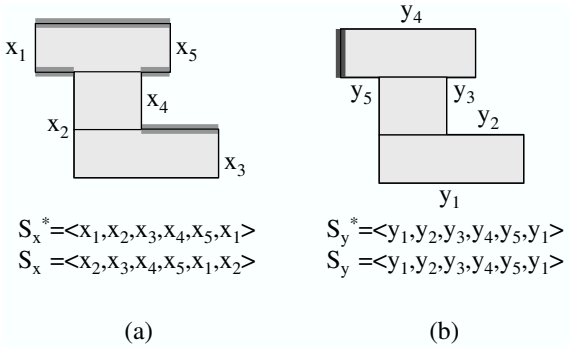

Figure 5: (a) $S_x^* =< x_1, x_2, x_3, x_4, x_5, x_1 >$, and an example $S_x =< x_2, x_3, x_4, x_5, x_1, x_2 >$. The lightly shaded segments are *horizontal inversions*, as defined in Definition 4. (b) $S_y^* =< y_1, y_2, y_3, y_4, y_5, y_1 >$, and an example $S_y =< y_1, y_2, y_3, y_4, y_5, y_1 >$ The heavily shaded segment is a *horizontal inversion*, as defined in Definition 4.

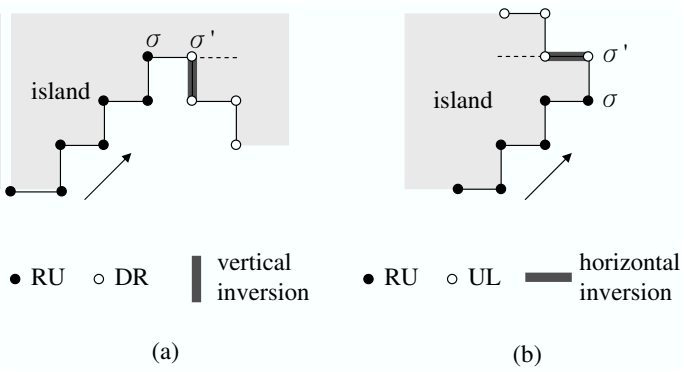

Figure 6: An example switching of direction pair RU. The shaded segments are *vertical* and *horizontal inversions*. Assume that the current direction pair is RU. If the traversal meets *vertical* and *horizontal inversions*, the direction pair would be changed to DR and UL, respectively.

begins counterclockwise; In Figure 5, for example,

$$S_x^* =< x_1, x_2, x_3, x_4, x_5, x_1 >,$$

and

$$S_y^* =< y_1, y_2, y_3, y_4, y_5, y_1 >,$$

in which x_1 and y_1 are the respective segments with the smallest x and y coordinates. Without loss of generality, the traversal sequences start and end at the same segment.

We have the property for S_x^* and S_y^* as follows:

LEMMA 1. **(Property of S_x^* and S_y^*)** *S_x^* and S_y^* start with increasing subsequences and must end with decreasing ones. Moveover, S_x^* and S_y^* are composed of alternate increasing and decreasing traversal subsequences.*

In Figure 5(a), for example, S_x^* begins in an increasing subsequence $< x_1, x_2, x_3 >$, followed by a decreasing one $< x_3, x_4 >$, followed by an increasing one $< x_4, x_5 >$, and finally ends in a decreasing one $< x_5, x_1 >$.

Here, we also introduce the traversal sequences S_x and S_y to relax the constraint in the definitions of S_x^* and S_y^* that the traversals start at the segments with the smallest x and y coordinates; that is, treating S_x and S_y as circular lists, they have the same orders as those of S_x^* and S_y^*, respectively, but might start with different segments. Obviously, S_x and S_y are also composed of alternate increasing and decreasing traversal subsequences, but may not start and end with increasing and decreasing subsequences since they may start in the middle of subsequences. In Figure 5(a), for example, S_x starts at x_2 which is in the middle of the first increasing subsequence in S_x^*, and thus S_x starts and ends in the increasing subsequences. Note that S_x^* and S_y^* are unique, but S_x and S_y are not.

Next, we define an *inversion* as follows, which is the key to our contour-traversal strategy.

DEFINITION 4. **(Vertical and Horizontal Inversion)** *A horizontal (vertical) inversion occurs when S_x (S_y) changes from increasing to decreasing or from decreasing to increasing.*

In Figure 5, for example, *horizontal* and *vertical inversions* occur in $\{(x_3, x_4), (x_4, x_5), (x_5, x_1), (x_1, x_2)\}$ and $\{(y_4, y_5)\}$ respectively, giving that $S_x =< x_2, x_3, x_4, x_5, x_1, x_2 >$ and $S_y =< y_1, y_2, y_3, y_4, y_5, y_1 >$.

LEMMA 2. **(Property of inversion)** *To determine an inversion, at least three consecutive contour segments are required.*

Subsequently, our contour-traversal strategy is presented in Table 1. In this table, the direction pair d_σ, which is presented by a tuple consisting of a *major* (the first letter) and a *minor* (the second letter) directions where point σ denotes the possible directions of the succeeding point. Here, U, D, L, and R denote respective up, down, left, and right directions, and the direction pairs without and with parentheses are for the counterclockwise and clockwise traversals, respectively. According to d_σ, succeeding point σ' will be found, and then according to the inversion after σ', $d_{\sigma'}$ will be set. Then, we repeatedly search succeeding points in the contour and set their direction pairs until coming back to the starting point. Following this strategy, we can traverse the contour correctly.

THEOREM 1. **(Correctness of direction-pair switch)** *The contour-traversal strategy listed in Table 1 is correct.*

PROOF 1. *The switching of the direction pair RU is proven here while others can be proven in the same way. Note that we prove by the properties (increasing and decreasing) of traversal sequences S_x and S_y. Given that point σ' is succeeding to point σ whose direction pair d_σ is RU, we prove that $d_{\sigma'}$ is set to RU, DR, and UL according to the inversion after σ'. $d_\sigma = RU$ implies that S_x and S_y are increasing. Consider a* vertical inversion. *S_y changes to decreasing, but S_x is still increasing, and thus $d_{\sigma'}$ may be set to DR or RD. However, it is impossible to set $d_{\sigma'}$ to RD; otherwise, the following traversal may enter the voltage island and lead to an incorrect traversal, as shown in the dashed line in Figure 6(a). Consider a* horizontal inversion. *In this case, S_x changes to decreasing, but S_y is still increasing, and thus $d_{\sigma'}$ may be set to UL or LU. However, it is impossible to set $d_{\sigma'}$ to LU; otherwise, the following traversal might enter the voltage island and lead to an incorrect traversal, as shown in the dashed line in Figure 6(b). The switching of other direction pairs can also be proven in the same way.*

COROLLARY 1. *If a traversal of a contour is started wth the RU direction and at the point with the smallest y coordinate (if there are multiple points point with the smallest y coordinate, choose the one with the smallest x coordinate), the direction pairs enclosed between parentheses in Table 1 would NOT appear. In other words, in Table 1, the ways of switching for the direction pairs without and with parentheses are independent.*

4.1.2 Contour Search Algorithm

Based on the properties of traversal sequences and our proposed contour-traversal strategy, we develop our voltage-island contour search algorithm, as illustrated in Figure 7. In the following, we detail the algorithm step by step.

Table 1: The contour-traversal strategy. Each direction pair is composed of a *major* (the first letter) and a *minor* (the second letter) directions. Point σ' is succeeding to point σ in a contour, and its direction pair $d_{\sigma'}$ can be determined by d_σ and the inversions after point σ'. The direction pairs without and with parentheses are for the counterclockwise and clockwise traversals, respectively.

d_σ	$d_{\sigma'}$		
	no inversion	vertical inversion	horizontal inversion
RU/(RD)	RU/(RD)	DR/(UR)	UL/(DL)
LD/(LU)	LD/(LU)	UL/(DL)	DR/(UR)
UL/(UR)	UL/(UR)	LD/(RD)	RU/(LU)
DR/(DL)	DR/(DL)	RU/(LU)	LD/(RD)

U: up; D: down; L: left; R: right.

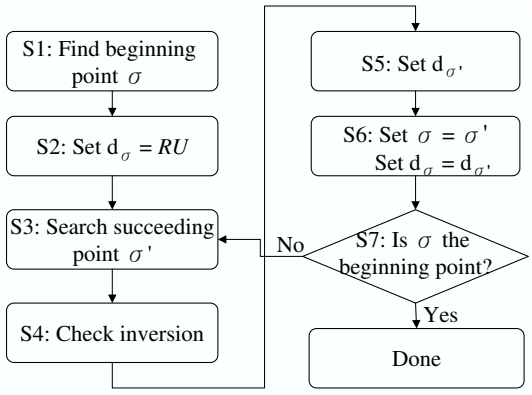

Figure 7: Flow of the voltage-island power-ring search algorithm.

• Steps 1 and 2: Initializing the traversal

The traversal starts at the point σ with the smallest y coordinate (if there are multiple points with the smallest y coordinate, choose the one with the smallest x coordinate) and initially set $d_\sigma = RU$. It can be seen that the starting point σ is the first element in H_y, as shown in Figure 3.

Setting $d_\sigma = RU$ is correct. Since σ is the first element in H_y, it is impossible to make d_σ related to directions L and D. In addition, we traverse the contour in counterclockwise manner. Hence, R is set as the major direction while U is the minor one; if we set U as the major direction, then the traversal is done clockwise.

• Step 3: Searching the succeeding point

In the traversal, the succeeding point σ' must be in the major or minor direction specified in d_σ.

LEMMA 3. *Given the point σ and its direction pair d_σ, the succeeding point in the contour (denoted by σ') must exist in the major or minor direction specified in d_σ.*

Lemma 3 and the hash-table implementation show that finding the succeeding point in the contour takes constant time on average since searching in hash tables takes constant time on average [2].

• Step 4: Checking the inversion after σ'

This is the core step of the algorithm. After obtaining point σ', we check the inversion after σ' and then set the direction pair $d_{\sigma'}$ in the next step.

To check the inversion after σ', we should be given the succeeding point of σ'. We, however, cannot determine the succeeding point of σ' before determining the inversion after σ'. Although we do not know where the succeeding point of σ' is, we can still guess and then verify it (three times at most).

For a clearer presentation, the point succeeding σ' is denoted by σ'', and the directions from σ to σ' and from σ' to σ'' are denoted by $D_{\sigma\to\sigma'}$ and $D_{\sigma'\to\sigma''}$, respectively. Since the traversal

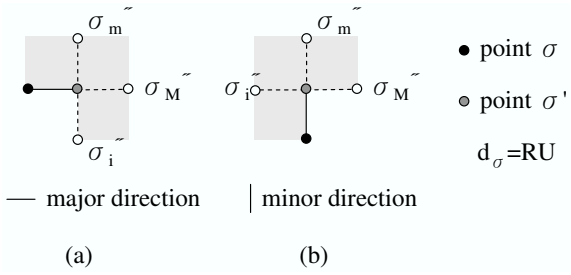

Figure 8: An example of checking the inversion after σ. (a) $D_{\sigma\to\sigma'}$ is in the major direction. It is obvious that σ_i'' is the point succeeding σ', and the vertical inversion occurs. (b) $D_{\sigma\to\sigma'}$ is in the minor direction. In this case, only if σ_M'' and σ_m'' do not exist, σ_i'' could be the succeeding point to σ', and the horizontal inversion occurs.

is in the coordinate system (each time we have four directions: up, down, left, and right), σ'' may come from three directions, except $D_{\sigma\to\sigma'}$. Those three directions are further classified as *major*, *minor*, and *inversive* directions. Hence, we extend σ'' to a set $\{\sigma_M'', \sigma_m'', \sigma_i''\}$ to specify the positions of σ''. It should be noted that at least one in $\{\sigma_M'', \sigma_m'' \sigma_i''\}$ exists.

LEMMA 4. **(Inversion Detection)** *The inversion can be detected in the following two situations.*

1. *$D_{\sigma\to\sigma'}$ is in the **major** direction of d_σ, and σ_i'' exists.*
2. *$D_{\sigma\to\sigma'}$ is in the **minor** direction of d_σ, and σ_i'' exists but neither σ_M'' nor σ_m'' does.*

PROOF 2. *According to Lemma 2, we need three segments to determine an inversion, but here we are given d_σ and thus know the states of S_x and S_y; so we do not need the first two segments for determining the states of S_x and S_y. Now, we use Figure 8 for a clear illustration, in which d_σ is set to RU. Note that because the traversal is counterclockwise, the voltage island occupies the region above the segment connecting σ and σ' in Figure 8 (a) and the region on the left of the segment connecting σ and σ' in Figure 8 (b). In Figure 8(a), $D_{\sigma\to\sigma'}$ is in the **major** direction, and σ_M'', σ_m'', and σ_i'' are detected. It is obvious that σ_i'' is the point succeeding σ', and the vertical inversion occurs. In Figure 8(b), $D_{\sigma\to\sigma'}$ is in the **minor** direction, and σ_M'', σ_m'', and σ_i'' are detected. In this case, only if σ_M'' and σ_m'' do not exist, σ_i'' could be the succeeding point to σ', and the horizontal inversion occurs.*

• Step 5: Setting $d_{\sigma'}$

We have been given d_σ and the inversion after σ'. According to Table 1, we set $d_{\sigma'}$.

• Step 6: Confirming the current segment

The segment connecting points σ and σ' has been confirmed as a contour segment. We let $\sigma = \sigma'$ and $d_\sigma = d_{\sigma'}$ to repeatedly search succeeding points in the contour.

• Step 7: Determining the termination

If σ is not the starting point, we repeatedly find succeeding points until coming back to the starting point. If σ is the starting point, the algorithm terminates, and the contour is composed of the points which have been set as point σ.

4.1.3 An Example of Contour Search Algorithm

In Figure 9(a), we search the contour starting at point p_1 which is the first element in H_y and set $d_{p_1} = RU$. Then, we search point p_2 as the succeeding point in the major direction R specified by d_{p_1}. We check the inversion after p_2, but no inversion is detected. Hence, we set d_{p_2} to RU according to Table 1. Next, since p_2 is not the starting point, we go back to the third step and repeatedly find the succeeding point.

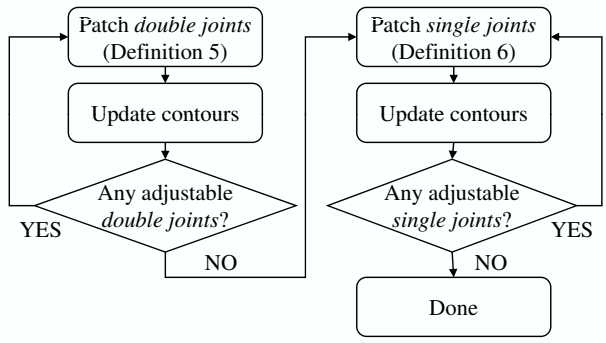

Figure 10: Power-ring corner-patching flow.

In Figure 9(b), due to the page limitation, we skip the description of the third step which has been presented in the aforementioned example. In the fourth step, an inversion after p_3 is detected because $d_{p_2 \to p_3}$ is in the minor direction of d_{p_2}, and σ_i'' (p_5) exists but neither σ_M'' nor σ_m'' does (the horizontal inversion is detected), as described in the second situation in Lemma 4. Then, according to Table 1, d_{p_3} is set to UL. Again, since p_3 is not the starting point, we go back to the third step and repeatedly find the succeeding point.

In Figure 9(c), in the fourth step, no inversion is detected and thus d_{p_5} is set to UL. Similarly, since p_5 is not the starting point, we go back to the third step and repeatedly find the succeeding points. The algorithm would terminate when we come back to p_1.

4.2 Corner-Patching Algorithm

After searching the power ring (contour) of each voltage island, we try to minimize the number of corners (see Definition 3) for the *via* reduction in the power rings by patching whitespaces into circuit blocks. It should be noted that the resulting power rings will not overlap, because we use only whitespaces for patching to minimize the via count. See Figure 10 for the corner-patching algorithm flow.

We first introduce single and double joints defined in Definitions 5 and 6 and illustrated in Figure 11.

DEFINITION 5. **(Double joint in a contour)** *If we **individually** extend vertical and horizontal contour segments of a voltage island, a double joint is a rectangle which is not part of the voltage island and is enclosed by **one extended** and **three original** segments.*

DEFINITION 6. **(Single joint in a contour)** *If we **simultaneously** extend vertical and horizontal contour segments of a voltage island, a single joint is a rectangle which is not part of the voltage island and is enclosed by **two extended** and **two original** segments.*

Moreover, single and double joints may be filled by whitespace. In Figure 12(a), for example, suppose that double joint d_1 in the left voltage island has been filled. Consequently, two single joints individually exist in the two voltage islands. Single joint s_1 associated with the right voltage island could be filled up by a complete whitespace; however, the single joint associated with the left voltage island cannot be filled because there is no complete whitespace that can fill the single joint. The complete whitespace for the single joint associated with the left voltage island is shown by the oblique-line region in Figure 12(b).

Now, we show the optimality of the corner-patching algorithm. We shall first show the correctness of the algorithm flow and then show the optimality of the algorithm.

LEMMA 5. **(Correctness of the corner-patching algorithm flow)** *It is impossible that filling single joints would induce double joints.*

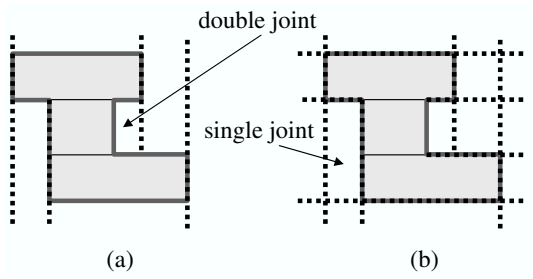

Figure 11: (a) An example double joint. Extending the *vertical* contour segments, we have a double joint on the right-hand side of the voltage island. (b) An example single joint. Extending the *vertical* and *horizontal* segments, we have a single joint on the left-hand side of the voltage island.

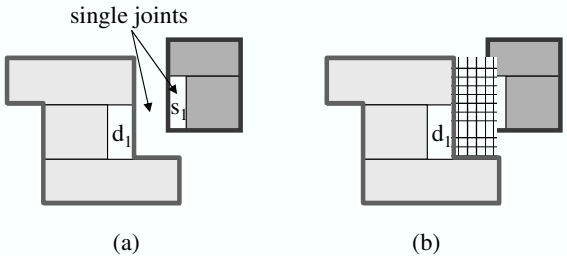

Figure 12: (a) There is no complete whitespace that can fill the single joint associated with the left voltage island; meanwhile there is a complete whitespace that can fill single joint s_1 associated with the right voltage island. (b) The complete whitespace for the single joint associated with the left voltage island is shown by the oblique-line region.

We now present our corner-patching algorithm. We search double joints starting at the point with the smallest y coordinate (if there are multiple points with the smallest y coordinate, choose the one with the smallest x coordinate). Double joints exist in the four situations illustrated in Figure 13(a). Therefore, each time we scan three contour segments to verify if a double joint exists. If we detect a double joint, we immediately try to fill the double joint. If the double joint can be filled, we update the contour and detect new double joints locally; after patching a double joint, new double joints may be induced. If the double joint cannot be filled, we process the next double joint. After filling all adjustable double joints, we try to search and fill single joints which exist in the four situations illustrated in Figure 13(b). The algorithm terminates when all adjustable single joints are filled. We have the optimality claim as follows:

THEOREM 2. **(Optimality of the corner-patching algorithm)** *Given a contour, the corner-patching algorithm minimizes the number of corners in the contour by patching whitespaces.*

PROOF 3. *We first fill all double joints, if any. Then, we fill all single joints, if any. According to Lemma 5, after filling single joints, we do not have any* new *double joints for filling. It implies that we impossibly make any changes to make the current situation better, and thus the current solution is optimal. Therefore, the optimality of the corner-patching algorithm holds.*

5. EXPERIMENTAL RESULTS

We conducted two experiments in this section. Section 5.1 validates the corner effect in power rings on IR drop, and Section 5.2 presents the experimental results from our algorithms.

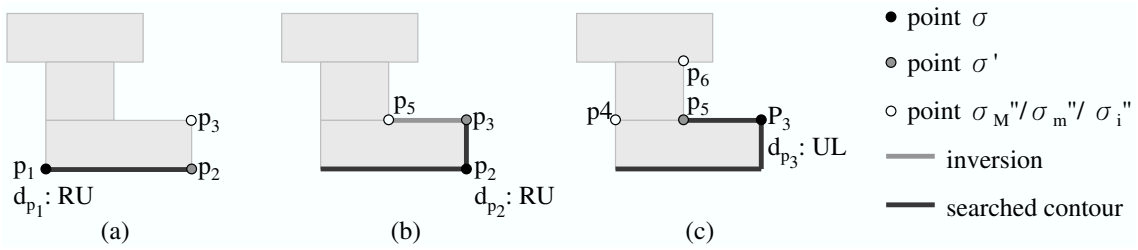

Figure 9: (a) The counterclockwise traversal begins at p_1 whose direction pair d_{p_1} is set to RU. p_2 is found in the major direction specified by d_{p_1}, and d_{p_2} is set to RU because no inversion is detected. (b) p_3 is found in the minor direction specified by d_{p_2}, and d_{p_3} is set to UL because the horizontal inversion is detected, as described in the second situation in Lemma 4. (c) p_5 is found in the minor direction specified by d_{p_3}, and d_{p_5} is set to UL because no inversion is detected.

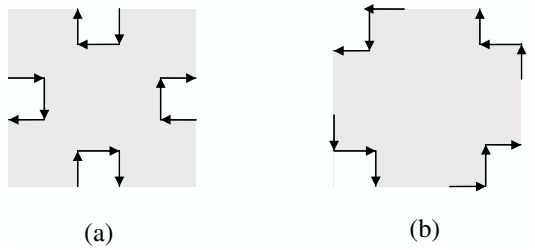

Figure 13: (a) Four situations in which double joints exist. (b) Four situations in which single joints exist.

5.1 Corner Effect Validation

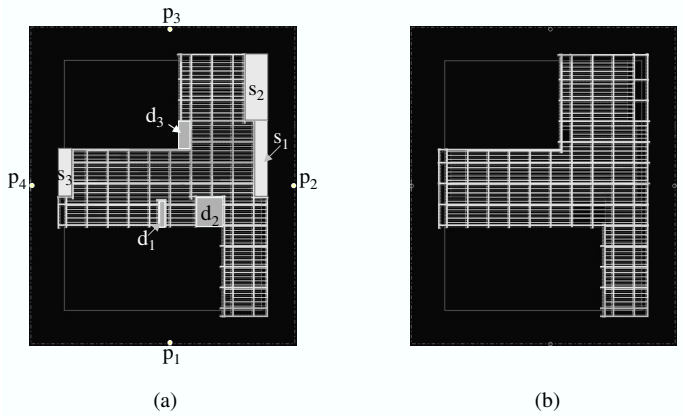

Figure 14: Layout of the r2000_soc circuit. (a) Three double and three single joints are placed in the power ring, and four power pads are attached in the four boundaries of the chip. (b) Three double and three single joints are filled.

The corners in a power ring induce vias and increase IR drop. To justify this claim, we tested various numbers of corners in a power ring, and then individually evaluated the IR drop by Cadence SOC Encounter v6.2. We took the r2000_soc circuit from Opencores [12] and synthesized it by Synopsys Design Compiler v2007.12 with the TSMC 0.18μm technology. The circuit layout contain three double and three single joints in its power ring, and the four power pads are attached to the four boundaries of the chip.

Table 2 reports the experimental results. Column "power pads" gives the power pads we used, and columns "3d3s", "2d3s", "1d3s", "0d3s", "0d2s", "0d1s", and "0d0s" give the IR drops for d_1, d_2, d_3, s_1, s_2, and s_3 being filled one by one. It is obvious from Table 2 that minimizing the number of corners can significantly reduce IR drop.

Table 2: Comparison of IR drops in the power rings containing various numbers of double and single joints. The power rings are shown in Figure 14.

power pads	3d3s	2d3s	1d3s	0d3s	0d2s	0d1s	0d0s
p1	15.456	15.440	13.895	13.769	13.455	13.417	13.39
p1–p2	5.363	5.343	4.643	4.520	2.723	2.667	2.665
p1–p3	2.871	2.852	2.447	2.403	1.736	1.716	1.714
p1–p4	1.631	1.626	1.593	1.586	1.177	1.163	1.075

5.2 Corner Number Minimization by Whitespace Patching

Our algorithms was implemented in the C++ programming language and experimented on a workstation with an AMD-64 2.2 GHz CPU and 8 GB memory running Linux. The benchmark circuits are taken from the GSRC floorplanning benchmarks. Other assumptions in our experiments are listed below. Note that the efficiency and correctness of our algorithms are not affected by these assumptions.

- Four available supply voltages are used in the benchmarks and are randomly assigned to blocks. Note that previous works have shown that two or three supply voltages are sufficient for practical circuit designs [6].
- The positions of blocks are fixed, and they are given in the original benchmarks; that is, our algorithm is applied at the post-floorplanning stage.

Table 3 reports our experimental results. Column "#islands" gives the numbers of the voltage islands, and columns "original," "after double-joint patching," and "after single-joint patching" give the numbers of corners before and after patching double and single joints. The experimental results show that our algorithm can reduce the number of corners by 33% on average. In addition, the *total* running time for the 16 GSRC circuits is less than one second. For the biggest benchmark circuit n300 that contains 300 blocks, 57 voltage islands, and 862 corners, our algorithm can reduce 298 corners in only 0.06 seconds. The results show that the proposed algorithms are efficient and effective.

Figure 15 shows the resulting layouts of n30b and n50b. Figures 15(a) and (d) show that our contour search algorithm can search the power rings aligning to the outer boundaries of the voltage islands correctly. Figures 15(b) and (e) show that our double-joints patching algorithm can fill the double joints and reduce the number of corners from 94 and 140 to 80 and 110 (reduced by 14 and 30), respectively. Figures 15(c) and (f) show that our single-joint patching algorithm can fill the corners and further reduce the corners from 80 and 110 to 62 and 88 (reduced by 18 and 22), respectively.

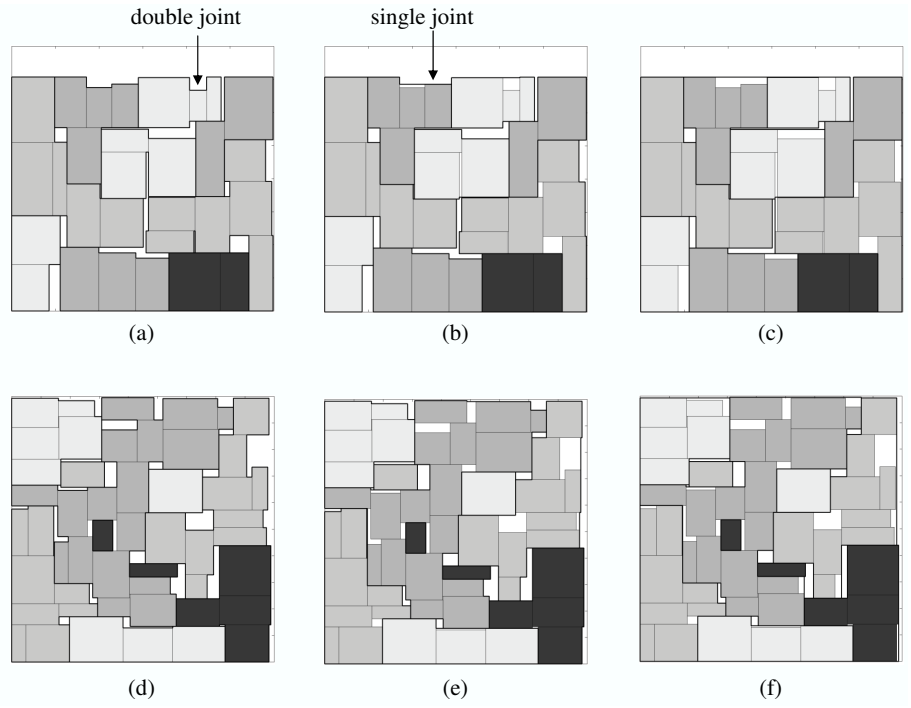

Figure 15: The layouts of n30b and n50b. The power rings are shown by the bold lines. (a) and (d) show that our algorithm can search the power rings (contour) of the voltage islands correctly. (b) and (e) show the results after patching double joints. (c) and (f) show the results after patching single joints.

Table 3: Comparison of the number of corners before and after patching corners.

Bench-mark	#islands	original, ψ_o	after double-joint patching, ψ_d	after single-joint patching, ψ_s	runtime (sec.)
n10	3	34	26	22	<0.01
n10b	3	28	26	20	<0.01
n10c	3	30	26	20	<0.01
n30	7	82	68	58	<0.01
n30b	9	94	80	62	<0.01
n30c	8	94	78	68	<0.01
n50	14	156	132	106	<0.01
n50b	10	140	110	88	<0.01
n50c	14	162	138	120	<0.01
n100	20	284	218	194	<0.01
n100b	17	290	220	174	0.01
n100c	20	292	218	186	0.01
n200	40	566	452	388	0.02
n200b	45	588	472	408	0.02
n200c	41	592	466	408	0.02
n300	57	862	670	564	0.06
average: ψ_d or ψ_s/ψ_o			0.79	0.67	

6. CONCLUSION

In this paper, we have introduced a post-floorplanning power-ring synthesis and adjustment algorithm. The proposed algorithm enables power ring alignment to the outer boundaries of voltage islands. This is a novel idea of the power-ring synthesis for MSV designs. Further, the proposed algorithm is adaptable to be embedded in MSV floorplanners because of its efficiency. Of course, it is also can be employed to synthesize the power rings after floorplanning.

7. REFERENCES

[1] R. Ching, E. Young, K. Leung, and C. Chu, "Post-Placement Voltage Island Generation," in *Proc. ICCAD*, pp. 641–646, 2006.

[2] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, "Introduction to Algorithm, 2nd Edition," pp. 228, 2001.

[3] Y. Cai, B. Liu, J. Shi, Q. Zhou, X. Hong, "Power Delivery Aware Floorplanning for Voltage Island Designs," in *Proc. ISQED*, pp. 350–355, 2007.

[4] J. Hu, Y. Shin, N. Dhanwada, and R. Marculescu, "Architecting Voltage Islands in Core-Based System-on-a-Chip Designs," in *Proc. ISLPED*, pp. 180–185, 2004.

[5] W.-L. Hung, G.M. Link, Yuan Xie, N. Vijaykrishnan, N. Dhanwada, and J. Corner, "Temperature-Aware Voltage Islands Architetecting in System-on-Chip Designs," in *Proc. ICCD*, pp. 689–694, 2005.

[6] M. Hamada, Y. Ootaguro, and T. Kuroda, "Utilizing Surplus Timing for Power Reduction," in *Proc. CICC*, pp. 89–92, 2001.

[7] W.-P. Lee, H.-Y. Liu, and Y.-W. Chang, "Voltage Island Aware Floorplanning for Power and Timing Optimization," *Proc. ICCAD*, pp. 389–394, 2006.

[8] B. Liu, Y. Cai, Q. Zhou, and X. Hong, "Power Driven Placement with Layout Aware Supply Voltage Assignment for Voltage Island Generation in Dual-Vdd Designs," in *Proc. ASPDAC*, pp. 582–587, 2006.

[9] W.-K. Mak, and J.-W. Chen, "Voltage Island Generation under Performance Requirement for SoC Designs," in *Proc. ASP-DAC*, pp. 798–803, 2007.

[10] Q. Ma and Evangeline F. Y. Young, "VoltageIsland-Driven Floorplanning," in *Proc. ICCAD*, pp. 644–649, 2007.

[11] S. Nahar and S. Sarni, "Fast Algorithm for Polygon Decomposition," in *IEEE TCAD*, vol. 7, no. 4, pp. 473—483, 1988.

[12] Opencores, http://www.opencores.com

[13] K. Usami and M. Horowitz, "Clustered Voltage Scaling Technique for Low-Power Design," in *Proc. ISLPED*, pp. 3–8, 1995.

[14] H. Wu, I.-M. Liu, M. Wong, and Y. Wang, "Post-Placement Voltage Island Generation under Performance Requirement," in *Proc. ICCAD*, pp. 309–316, 2005.

[15] H. Wu, M. Wong, and I.-M. Liu, "Timing-Constrained and Voltage-Island-Aware Voltage Assignment," in *Proc. DAC*, pp. 429–432, 2006.