A Fast Algorithm for Power Grid Design*

Jaskirat Singh Sachin S. Sapatnekar
Department of Electrical & Computer Engineering
University of Minnesota
Minneapolis, MN 55455
{jsingh,sachin}@ece.umn.edu

ABSTRACT

This paper presents an efficient heuristic algorithm to design a power distribution network of a chip by employing a successive partitioning and grid refinement scheme. In an iterative procedure, the chip area is recursively bipartitioned, and the wire pitches and the wire widths of the power grid in the partitions are repeatedly adjusted to meet the voltage drop and current density specifications. By using the macromodels of the power grid constructed in the previous levels of partitioning, the scheme ensures that a small global power grid system is simulated in each iteration. A post-processing step at the end of the optimization is employed to maximize the alignment of wires in adjacent partitions. The effectiveness of the scheme is demonstrated by designing various power grids with real circuit parameters and realistic input current values. The proposed algorithm is able to design power grids comprising thousands of wires and more than a million electrical nodes in about 7-16 minutes. The proposed design scheme as compared to a multigrid-based power grid design scheme saves about 7%-12% of wire area.

Categories and Subject Descriptors
B.7.2 [**Hardware**]: INTEGRATED CIRCUITS-Design Aids
General Terms
Power Grid Design, Optimization
Keywords
Locality, macromodel, bipartitioning, wire pitch

1. INTRODUCTION

Increasing operating frequency, power density and lower supply voltages have made the design of a high performance power distribution network a challenging problem. While the decrease in the widths of the metal wires has led to an increase in the wire resistance, the currents through these wires have also drastically increased due to the increase in chip densities. As a consequence, it has become essential that the power delivery network is carefully designed to prevent the loss of circuit performance due to IR drop and failures due to electromigration (EM).

*This work was supported in part by the SRC under contract 2003-TJ-1084 and the NSF under award CCR-0205227.

In view of this, there have been various schemes proposed for power grid design. Typically, the choices available to a supply net designer are to (i) appropriately size the supply net wires [1–5], (ii) perform topology optimization, i.e., to assign suitable pitches to the power grid wires and/or determine the optimal assignment of the pins to the pads and placement of pads on the power grid [6–10], and (iii) add decoupling capacitors [11, 12].

In general, the power distribution network design schemes can be divided into the following two categories:

(A) An iterative optimization heuristic, employing an explicit circuit analysis step in the main optimization loop to determine the violating parts of the power grid circuit [7], [11].

(B) A mathematical optimization scheme formulated as a general nonlinear program and solved with the aid of nonlinear optimization techniques [1–5], [8].

The desirable characteristic of the supply network design methods based on scheme A is the guaranteed accuracy of the final solution, ensured by the process of performing an explicit circuit simulation step in each iteration, to detect and fix the IR drop and EM violations. However, these methods typically have large run times as each simulation of a power network, comprising hundreds of thousands of electrical nodes, is extremely time consuming. For the supply net design methods built on scheme B, usually, the circuit analysis is implicitly carried out by making the circuit constraints, i.e., the Kirchoff's current and voltage laws, as a part of the constraints set. In the original form, the solution to such a nonlinear problem formulation is known to be computationally intensive which makes it extremely prohibitive for power network design problems involving millions of design variables. Hence, to achieve efficiency these methods typically either employ some constraint relaxations to transform a general nonlinear optimization problem to special forms of nonlinear programs such as the convex program, which can be efficiently solved or introduce some approximations to reduce the problem size. Although, these methods provide solutions which are more efficient than the ones based on the explicit circuit simulation schemes, the final solutions are inherently subject to inaccuracies due to the relaxations and approximations introduced in the original nonlinear problem formulation.

In this paper, we propose a novel, fast yet accurate algorithm to design the power distribution network in the form of a non-uniform power grid. The power grid design procedure achieves efficiency by using the *notion of locality* in the power grid design. The locality idea for power grid analysis was proposed in [13]. The concept of locality in grid design is based on the observation that to construct the power grid locally in a specific region of the chip, it suffices to focus on the details of that specific region only. The regions of chip that are *far away* from the specific region, are not likely to affect the local grid design in the specific region. Hence, the far away regions can be abstracted as coarse models in the local grid design step. Figure 1 illustrates the concept of locality in power grid design. A violating grid region, i.e., one that violates the constraints, is shown by the dashed rectangle in Figure 1(a). This can be fixed by adding more wires locally in and around the violating region. Due to the locality property, it is reasonable for the details of parts of power grid in the regions far away from the violating region to be abstracted away, as seen in Figure 1(b). The use of these abstractions, or *macromodels*, for parts of the power grid circuit has two main advantages. Firstly, this results in much smaller system sizes for the power grid circuit, which leads to efficient circuit analysis. Secondly, by focusing the grid design effort on the details of the local region only, the search space for choosing the optimal design parameters, e.g., the wire pitches and the wire widths, is greatly reduced.

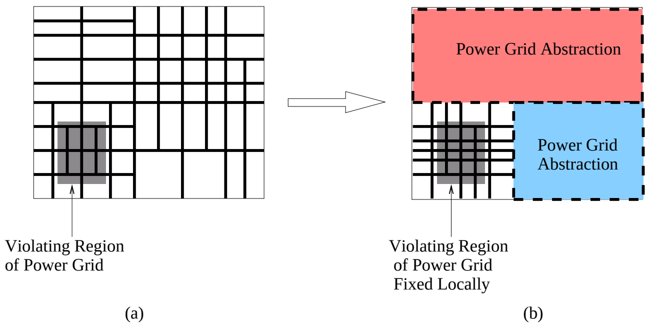
Power Grid Abstraction
Power Grid Abstraction
Violating Region of Power Grid
Violating Region of Power Grid Fixed Locally
(a)
(b)

Figure 1: The concept of locality in power grid design. (a) A detailed power grid with a violating region shown with the shaded rectangle. (b) Details of regions of power grid far from the violating region abstracted away and the violations fixed locally.

The idea of working with detailed local models and abstractions of far away regions is especially favorable in the case of flip-chip packages where the power pads are distributed throughout the chip area by using C4 bumps. For a flip-chip package, most violations of power grid in a specific regions can be fixed by locally modifying the power grid just in and around the violating regions. Due to the availability of enough pads around the violating region, the power grid wires in the local region contain the path of least resistance for the current to flow from the nearest V_{dd} pads to the violating nodes. For chips with wire-bond packages, the strategy of local modification of the grid may not work because of the concentration of power pads on the chip periphery. However, even in the wire-bond case it is advantageous to work with coarse models of parts of the grid initially, and then iteratively refine the grid in different regions.

Based on the notion of locality in power grid design, we propose an efficient and accurate grid design procedure. The method employs and iterative scheme of recursively partitioning the chip area and constructing the power grid locally in the partitions by adjusting the wire pitches and the widths. The outline of our method is as follows:

- We present a heuristic algorithm to design a supply grid that meets the IR drop and the EM constraints. The optimization to construct the power grid is carried out under DC conditions using an iterative refinement scheme. In our implementation, the power grid is designed for the top two metal layers of the chip, but the same procedure can be easily extended to design a power network spanning multiple layers of metal.
- We start off the design procedure by dividing the chip area into two partitions. The power grid in the two partitions is then constructed by placing thick or very wide wires at an initial pitch. The pitches in the two partitions are then repeatedly reduced until the initial grid meets the constraints.
- In each of the subsequent iterations, a previous partition is further divided into two smaller partitions and a refined power grid is reconstructed *locally* in these smaller partitions. The solution of the grid designed in the previous iterations is used to guide the design of the power grid in the current iteration.
- We use a previously proposed macromodeling technique [14] to construct abstractions of partition of the power grid. Employing a hierarchical circuit analysis method in each iteration, to determine the nodes and branches which violate the reliability constraints, ensures the accuracy of grid design. Since, in each iteration we construct the macromodels of only two partitions, and reuse the macromodels formed in previous iterations, the analysis step is very fast.
- At the end of the optimization, a post processing step is used to maximize the alignment of wires in adjacent partitions.

2. PRELIMINARIES

A power grid comprises metal wires running in the orthogonal directions and spanning multiple layers (typically, 5-8 for current microprocessor designs). The wires in two consecutive layers of metal are electrically connected to each other by using vias. The wires in the top-most metal layers are electrically connected to the V_{dd} pads. The power pads are located either on the peripheral power ring as in the case for a chip with a wire-bond package or are distributed over the entire chip area, using C4 bumps, as in the case of flip-chip package. This system of pad connections and network of metal wires carrying currents from the V_{dd} pads to the underlying gates in the functional blocks, can be modeled as an equivalent electrical circuit comprising of millions of nodes. Under DC conditions, as illustrated in Figure 2, the power grid can be modeled as a resistive mesh, with the pads replaced by voltage sources. As seen in the figure, the wires are replaced by their equivalent resistances and the worst-case switching activities of the gates in the underlying functional blocks determines the loading currents.

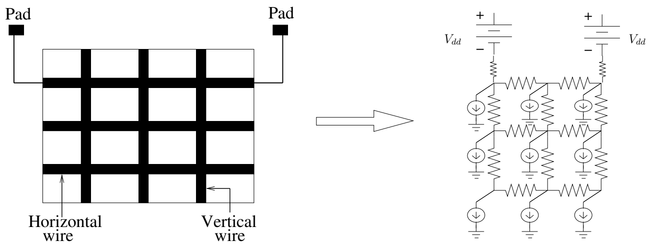

Figure 2: A power grid and its equivalent circuit model.

Using the circuit model as shown in Figure 1, the branch resistance *r* of branch *i* can be expressed as:

$$r = \rho_s \frac{p}{w_i} \tag{1}$$

where ρ_s is the sheet resistivity, w_i is the width of the wire segment corresponding to the branch *i* and p is the pitch of the power grid wires, which is the same as the length of the wire segment. The pitch of the power grid wires could be different for different layers and in fact, may not even be constant for a given layer.

For all nodes *j* and all branches *i* in the power grid circuit, the following constraints must be satisfied:

1. **The IR drop constraint**: $V_j > V_{spec}$
2. **The current density constraints (EM)**: $|I_i| < \sigma w_i$

The voltage of node *j* is denoted as V_j, the branch current of branch *i* is denoted as I_i and σ is the specified current density for a fixed thickness (height) of the metal layer.

In the following sections of the paper, we describe the design procedure to construct a power grid that meets the above two constraints of IR drop and EM. The inputs to our power grid design problem are the number of power pads and their precise connection locations to the grid wires in the top layer, the placed functional blocks and an estimate of worst case currents drawn by the gates in the functional blocks. The amount of currents draw by each of the functional blocks can be determined by current estimation techniques such as the ones proposed in [15]. We use the power grid circuit model as shown in Figure 2. Each node j in the power grid circuit is loaded by a constant current source c_j. Given the current estimate I_{f_k} drawn by a functional block k, the values of constant current sources loading the power grid are chosen in such a way that the sum of all current sources at node locations lying over the functional block k add up to the functional block current I_{f_k}.

3. POWER GRID DESIGN PROCEDURE

3.1 Grid refinement

A power grid covering the entire chip area comprises thousands of wires. The equivalent circuit of such a power grid, as the one shown in Figure 2, contains millions of electrical nodes, making it prohibitive to simulate. To achieve efficiency in the circuit analysis step, it becomes essential to work with a coarse initial power grid representation, which yields a power grid circuit of manageable size, and then iteratively refine the coarse model. In our power grid design procedure, we construct the coarse grid by using unrealistically thick power grid wires initially, and then subsequently improve the coarse grid model by a *grid refinement* operation, in which, a power grid containing a few thick wires is replaced by a grid comprising of many thinner wires. The advantage of using a grid with thick wires is that it greatly reduces the system size as there are fewer electrical nodes in the equivalent circuit model of Figure 2.

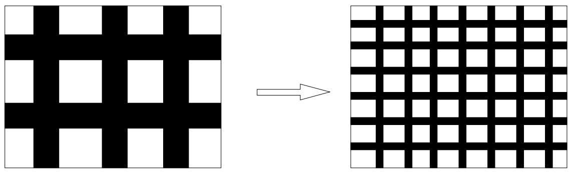

Figure 3: A power grid with thick wires and large pitch refined to a grid with thinner wires and smaller pitch.

Figure 3 depicts the grid refinement operation. Since the refined grid has wires of smaller widths as compared to the wires of the coarser power grid, the wire resistances of the refined power grid are higher. However, the higher wire resistance is compensated by the fact that there are more number of wires in the refined power grid as compared to the coarser grid. Having more number of wires in the grid implies that there are more paths for current to flow from a V_{dd} pad to any node, which decreases the effective resistance of the power grid circuit. Similarly an increase in the current densities of the thinner wires in the refined grid is offset by a decrease in the magnitude of current flowing through each branch, due to the increase in the number of wires.

3.2 Power grid design by recursive bipartitioning

We apply the *divide and conquer* approach to our power grid design procedure by successively dividing the chip area into smaller regions or partitions and iteratively constructing power grids in the smaller partitions using the notion of locality in power grid design. The recursive bipartitioning idea works on the strategy of solving a small local power grid design problem in each step, comprising of choosing optimal pitches for two partitions such that the new grid constructed in the two partitions meets the specifications, and the previously constructed grids in the other partitions maintain their correctness in terms of meeting the IR drop and EM constraints.

Figure 4 illustrates the recursive bipartitioning process. As shown in the figure, the process of partitioning the chip area to iteratively construct the power grid is equivalent to growing an *almost complete* binary treereferred to as the *partition_tree*. Each element in the partition tree contains the information about the power grid, i.e, the wire width and the pitches, in a previously processed partition, represented as the non-shaded tree elements in Figure 4, or the two partitions being processed in the current iteration, referred to as the *active partitions*, shown as the shaded tree elements in Figure 4. The height of the partition_tree determines the current level of partitioning. A new partition is constructed by making a vertical (horizontal) cut, by introducing a vertical (horizontal) *partition wire* at the half length (half width) of the parent partition. Consecutive levels of partitioning alternate the cut directions between vertical cuts and horizontal cuts. The process of adding two children to a parent node in the tree is equivalent to the grid refinement operation described in Section 3.1. In other words, when two children nodes are added to a parent node in the tree, the coarse power grid in the partition corresponding to the parent node is replaced by finer grids of the children nodes.

In the following sections we explain the recursive bipartitioning idea for power grid design in detail.

3.3 First level of partitioning

As seen in Figure 4(b), the first level of partitioning is equivalent to adding two child nodes to the root of the partition_tree. In this step, we start with the full chip area and divide it into two parts by adding a vertical power grid partition wire at the half chip length.

Figure 5 depicts the division of chip into two halves or partitions. Choosing an initial wire width and pitch for the two partitions, such that the worst case voltage drop is about twice the specified drop, we construct an initial grid in the two partitions and then iteratively decrease the pitches of power grid in the two partitions until there are no IR drop and EM violations. The circuit analysis to detect the violations is performed using the macromodeling approached proposed in [14]. A macromodel is an abstraction of a linear network and has the following transfer characteristic:

$$\mathbf{I} = A \cdot \mathbf{V_p} + \mathbf{S}, \quad \mathbf{I}, \mathbf{V_p}, \mathbf{S} \in R^m, \quad A \in R^{m \times m} \qquad (2)$$

where, m is the number of ports in the system, **I** is a vector of currents flowing into the system through the ports, V_p is the vector of voltages at the port nodes, A is the admittance matrix and **S** is a vector of currents from each port to the reference node.

Using the macromodeling idea, all nodes in the two partitions except the port nodes are abstracted away. The port nodes are those

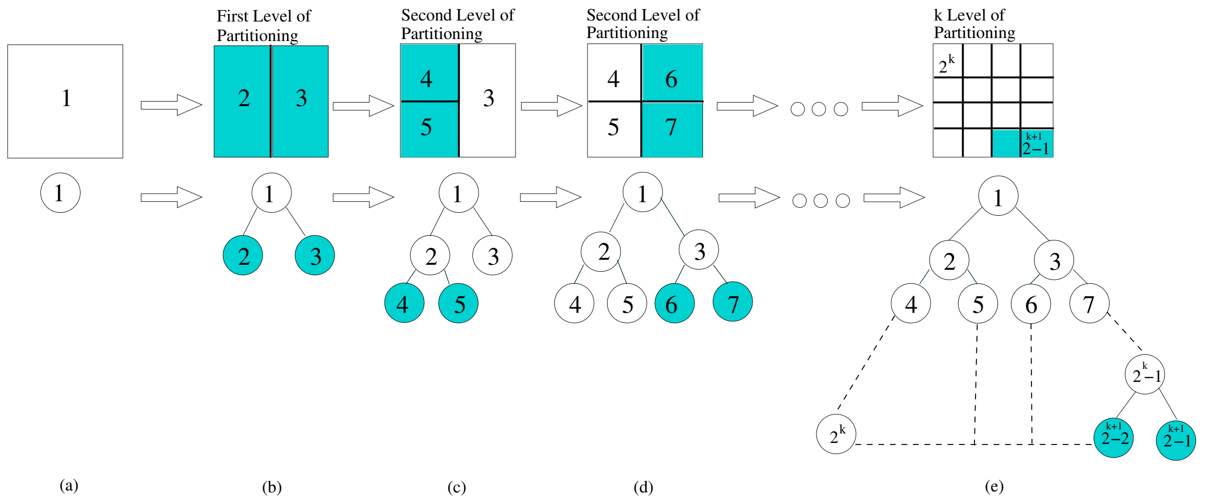
First Level of Partitioning
Second Level of Partitioning
Second Level of Partitioning
k Level of Partitioning
(a)
(b)
(c)
(d)
(e)

Figure 4: The recursive bipartitioning process to design the power grid. Each partition cut is equivalent to adding two elements in the binary *partition_tree*. The height of the tree represents the level of partitioning. The two shaded elements refer to the two partitions where the new power grid is designed in the current iteration.

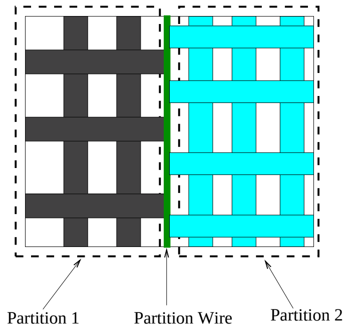

Figure 5: Chip divided into two partitions and the power grid constructed in the two partitions.

that lie on the partition wire through which the two partitions connect to each other. Using the matrix algebra described in [14], we construct the macromodels given by the parameters $(A, \mathbf{S})$ for the two partitions. Following the hierarchical circuit analysis approach, the macromodel parameters are stamped in the MNA equations of the global system:

$$M \cdot \mathbf{X} \;=\; \mathbf{b} \qquad (3)$$

- M is the matrix containing the conductance links between the pads and the wires in all the partitions and the stamps of macromodel parameter A for each partition.
- $\mathbf{X}$ is the vector of voltages at the port nodes of the partitions.
- $\mathbf{b}$ is vector containing the stamps of macromodel parameter $\mathbf{S}$ for each partition.

The global system is solved and subsequently the node voltages within the two partitions are determined to check for any IR drop and EM violations. In case a partition does not meet the voltage drop and current density specifications, the wire pitch of the partition is reduced by a factor β and the process of creating the macromodels and solving the power grid system by hierarchical analysis is repeated. It is worth emphasizing that at this first level of partitioning we use *unrealistically* thick wires, e.g., wires having widths between 80 and 200 μm. Hence, the system size of each partition is fairly small, e.g., 1000 to 3000 nodes. As a consequence, the iterative process of constructing new macromodel parameters $(A, \mathbf{S})$ after decreasing the wire pitch and simulating the grid repeatedly is extremely fast. At the end of this step, we obtain a coarse power grid constructed in the two partitions that span the entire chip area.

3.4 Second level of partitioning

We use the power grid constructed at the first partitioning level to guide the grid design in the next level. First, the power grid constructed in the left partition is ripped up. Then, a horizontal cut is made in the left partition by introducing a horizontal power grid partition wire at half width of the chip. Hence, the left partition is further divided into top-left and bottom-left partitions. As seen in Figure 4(c), the second partitioning level step begins by adding child nodes to the parent nodes 2 and 3 in the partition_tree or equivalently growing the almost complete binary tree to the next level. Leaving the previously constructed grid in the right partition intact, a finer grid is designed in the top-left and the bottom-left partition.

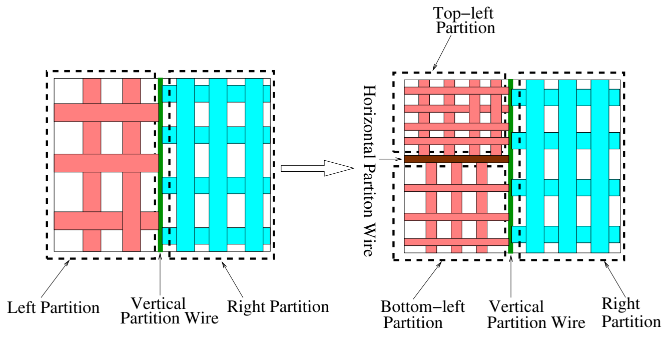

Figure 6: The second partitioning level in the grid design procedure

Figure 6 shows the step of partitioning the chip at the second level in the power grid design procedure, in which the left partition is split into top-left and bottom-left partitions.

The grid design procedure for these two partitions is exactly similar to the one employed in the first level of partitioning to construct the grids in the left and the right partitions. We start off the grid construction in the top-left and bottom-left partitions by constructing an initial grid with wire width chosen as some factor, $\gamma < 1$, of the wire width of the parent node, i.e., the grid designed in the left partition at the first partitioning level. This is precisely the grid refinement technique explained in Section 3.1 and shown in Figure 3, with the difference that we now also maintain a coarse macromodel for the right partition. The grid of the left partition with thicker wires and larger pitch is replaced with the grids in partitions top-left and upper-left having thinner wires but smaller pitch. The macromodel parameters $(A, \mathbf{S})$ are calculated for the top-left and upper-left partitions and they are stamped in the global system of equation (3) along with the macromodel parameters of the right partition which were previously computed in the first partitioning level. Again, using the hierarchical analysis technique, the IR drop and EM violations are detected for the power grid in the top-left and bottom-left partitions. The violations are corrected by putting more wires in the partitions by reducing the pitch by a factor β.

In addition to the voltage drop and the current density specifications, there is an additional requirement that the grids of the top-left and bottom-left partitions must meet. It is essential that the process of ripping up the original grid of the left partition and replacing it with the grids constructed in the top-left and the bottom-left partitions does not render the grid of the right partition ineffective in terms of meeting the specifications. To inspect if the correctness of the grid in the right partition is maintained, we could completely solve the right partition power grid again. However, this would be costly in terms of runtime as we would need to check the voltages of all the nodes of the grid within the right partition each time the pitches of the top-left or bottom-left partitions are decreased. To avoid this high simulation cost, we make use of the abstraction of the power grid of the right partition effectively. The voltages at only the port nodes of the right partition grid are evaluated. These voltages are compared with the port voltages of the right partition power gird obtained at the end of first level of partitioning and a grid violation, referred to as *previous_grid_violated*, is flagged if the maximum change in the port voltages is greater than a specified value, MAX_CHG_{spec}. In the event of such a violation, the pitches of the top-left and bottom-left are further decreased to have more wires in the power grid in order to maintain the correctness of the previously designed grid in the right partition. By this process, it is ensured that the port voltages of the previously correct grid design are not significantly disturbed by the new power gird which replaces the previous grid. By employing this strategy of checking for previous_grid_violation flags, we use the knowledge of the grid constructed at the previous level of partitioning to guide the grid design in the current partitioning level.

3.5 Grid refinement by recursive bipartitioning

In the process of growing the partition_tree of Figure 4(e), when each time two child nodes are added to a parent node in the partition_tree, the coarser power grid in the partition corresponding to the parent node, is converted to finer grids in the two active partitions corresponding to the child nodes. The wire widths of the children nodes are some factor, $\gamma < 1$, of the widths of the parent nodes.

With the growth of the partition_tree, the number of partitions in the current partitioning level increases exponentially. As a result, the number of port nodes and thus, the size of the global system of equation (3) grows rapidly. This adversely affects the runtime of the design procedure as in each iteration, following a pitch decrease in any one of the active partitions, it is required to construct and factorize the global matrix M, whose dimensions are rapidly increasing. To overcome this problem, we employ the *port approximation* technique suggested in [7] to reduce the macromodel sizes. By this approach, some of the port nodes located on the partition wires are collapsed. The port approximation scheme helps in checking the fast increase of the global system of equation (3) at the cost of reasonable simulation errors. However, the accuracy of the final solution is not compromised since the port approximation technique is switched off during the final few iterations of the design procedure.

The addition of two new children nodes in the partition_tree can only take place if the following are satisfied:

1. IR drop and EM constraints, for the new power grid being constructed in the two active partitions, are met. These violations are detected by the hierarchical circuit analysis step.

2. There are no previous_grid_violation flags set as described in Section 3.4. These violations are detected by checking the port voltages of all the neighboring partitions of the two active partitions. If the maximum change, between the new port voltages of the neighboring partitions and the port voltages of the neighboring partitions before constructing the new grid in the active partitions, exceeds a specified value, MAX_CHG_{spec}, the previous_grid_violation flags are set to true.

3. The wire pitches in the two active partitions are greater than the minimum wire pitch, p_{min}.

In order to fix the violations 1 and 2, the pitch of one or both the active partitions is decreased depending on which of the two active partitions are exhibiting these violations. The minimum pitch violation is an undesirable breakdown in our power grid design procedure. This occurs when the grid refinement operation does not work, i.e., the increase in the wire resistance by replacing a power grid having thicker wires with a power grid having thinner wires cannot be compensated by having more and more number of wires in the replacement grid. These minimum pitch violations cannot be fixed locally by modifying the grid in the active partitions. In this case, we need to traverse to the other tree elements which are the neighbors of the active partitions and add more wires in the neighboring partitions by decreasing the pitches of the power grid of the neighboring partitions. Fixing the minimum pitch violations thus adversely affects the runtime of the grid design procedure. However, we found out empirically that if the width reduction factor, $\gamma < 1$, is chosen not too small, e.g., if $\gamma \in [0.65, 1)$, such breakdowns are very rare events. We stop the optimization procedure at the end of k levels of partitioning. The value of k is determined by the specifying the minimum partition size.

3.6 Post processing for wire alignment

At the end of k-level partitioning, we have designed a power grid in the 2^k partitions of the chip, comprising of wires with potentially different wire pitches in each partition. As a result, the wires in the adjacent partitions maybe offset with respect to each other. To alleviate the misalignment situation, we introduce a post processing step to our grid design procedure.

Figure 7 illustrates the post-processing idea. As seen in Figure 7(a), a power grid with four partitions has misaligned wires in the

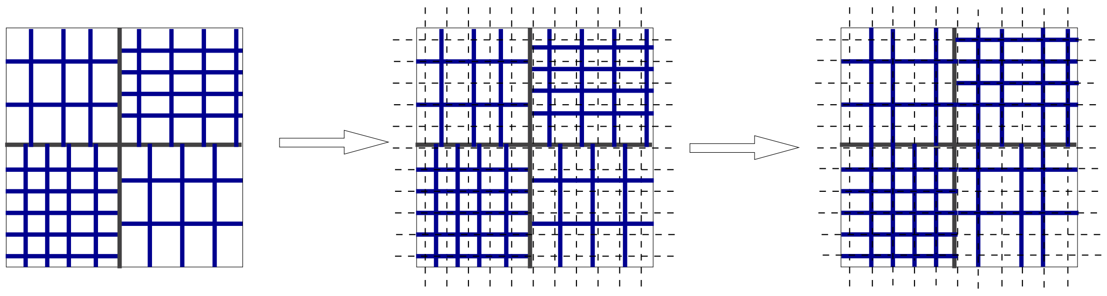

Figure 7: The post processing step to align the power grid wires in different partitions. (a) Power grid wires in adjacent partitions are misaligned. (b) A minimum pitch virtual grid, shown with dashed lines is constructed over the entire layout area. (c) The power grid wires are moved to the nearest position on the virtual grid. The wires in adjacent partitions are better aligned now.

adjacent partitions. To rectify this, a uniform and continuous virtual grid is superimposed over the layout area. The pitch of the virtual grid is chosen to be the minimum pitch of wires in all the partitions. The virtual grid is represented by dashed lines in Figure 7(b). In the next step, the power grid wires in all partitions are moved to the nearest location on the virtual grid. The virtual grid thus acts as a place holder for the real power grid wires. As seen in Figure 7(c), the wires in the adjacent partitions are better aligned at the end of the post-processing step. To make sure that the small movement of power grid wires does not affect the correctness of the grid, we perform a complete simulation by the hierarchical analysis technique. We found out in our experiments that the post-processing step hardly ever introduced any violations in the power grid. In the very few cases of grid not meeting the constraints after the post processing step, the violations are easily fixed by adding more wires in the violating partitions. The extra wires are still added on the vacant positions of the virtual grid to increase the wire alignment.

4. THE COMPLETE ALGORITHM

The psuedocode of the main loop of the power grid design algorithm is presented in Algorithm 1. We use the binary tree data structure to model successive partitioning of the chip area. This is represented by the *tree* array in Algorithm 1. Each element of the tree array represents a node in the partition_tree of Figure 4, and contains the information about the partition dimensions, and the wire width and pitch of the power grid constructed in the partition corresponding to the tree node. In lines 2-8 of the psuedocode, the root node of the tree, which represents the full chip area, is initialized. The outer while loop, extending from lines 10-47, performs the recursive bipartitioning and grid construction process. Inside the outer loop, a parent partition is divided into two children partitions in lines 11-15. The first inner while loop, in lines 18-32, performs the task of constructing the local power grid in the two active partitions, subject to reliability constraints. Within this loop, the steps of constructing the macromodel parameters $(A, \mathbf{S})$ for the two active partitions, and the circuit analysis by the hierarchical approach are represented in lines 21 and 22, respectively. In case the local grid, constructed in the two active partitions, does not meet the reliability constraints the pitches of the partitions are reduced by a factor δ, as shown in lines 26-31. The second inner while loop, shown in lines 35-41, ensures that the grid constructed in the active partition does not render the previously constructed grids in other partitions ineffective. This is made sure by checking the port voltages of neighboring partitions of the two active parti-

Algorithm 1 Power Grid Design

```
1: Power_Grid_Design(func_block_currents,power_pads_pos)
2: /*Initialize the root node of the partition tree*/
3: tree[1].wire_width=$W_{init}$;
4: tree[1].wire_pitch=$P_{init}$;
5: tree[1].width=chip_width;
6: tree[1].length=chip_length;
7: i=2;
8: cut_dir=0; /* cut_dir = 0/1 for a ver/horz cut*/
9: /*Begin Outer While Loop*/
10: while (i < MAX_NUM_PARTITIONS) do
11:   parent_index=i/2;
12:   $[child_1, child_2]$=Divide(tree[parent_index],cut_dir);
13:   /* Decrease the width, $\gamma \in (0,1)$ */
14:   $child_1$.wire_width = $\gamma$ * tree[parent_index].wire_width ;
15:   $child_2$.wire_width = $\gamma$ * tree[parent_index].wire_width;
16:   spec_met_flag_this_part_level = 0;
17:   /*Begin Inner While Loop*/
18:   while (spec_met_flag_this_part_level ==0) do
19:     $child_1$.wire_pitch= $\beta_1$* tree[i].wire_pitch; $\beta_1 \in [0,1)$
20:     $child_2$.wire_pitch= $\beta_2$* tree[i].wire_pitch; $\beta_2 \in [0,1)$
21:     $[(A_1, \mathbf{S_1}), (A_2, \mathbf{S_2})]$=make_pow_grid($child_1, child_2$);
22:     $[spec\_met\_flag_1, spec\_met\_flag_2]$=
        solv_grid($(A_1, \mathbf{S_1}), (A_2, \mathbf{S_2})$)
23:     if ($spec\_met\_flag_1$ AND $spec\_met\_flag_2$) then
24:       spec_met_flag_this_part_level =1;
25:     end if
26:     if (!$spec\_met\_flag_1$) then
27:       $\beta_1$=$\delta * \beta_1$; $\delta \in (0,1)$ /*Decrease the pitch*/
28:     end if
29:     if (!$spec\_met\_flag_2$) then
30:       $\beta_2$=$\delta * \beta_2$; $\delta \in (0,1)$ /*Decrease the pitch*/
31:     end if
32:   end while
33:   /*End Inner While Loop*/
34:   prev_grid_viol_flag=1;
35:   while (prev_grid_viol_flag ==1) do
36:     $[prev\_grid\_viol\_flag]$=chk_other_parts($child_1, child_2$);
37:     if (prev_grid_viol_flag == 0) then
38:       $\beta_1$=$\delta * \beta_1$; $\delta \in (0,1)$ /*Decrease the pitch*/
39:       $\beta_2$=$\delta * \beta_2$; $\delta \in (0,1)$ /*Decrease the pitch*/
40:     end if
41:   end while
42:   /* Add two children nodes to the tree*/
43:   tree[i]=$child_1$;
44:   tree[i+1]=$child_2$;
45:   i=i+2;
46:   cut_dir=!cut_dir;
47: end while
48: /*End Outer While Loop*/
49: do_post_processing_of_grid_to_align_wires();
```

tions. Lines 43-46, which are a part of the outer while loop, add the two children node to the parent node, after a satisfactory local grid has been designed in the two active partitions. At the end of the design procedure, the post processing step is employed to improve the alignment of power grid wires in the adjacent partitions.

5. EXTENSION TO MULTIPLE LAYERS

We have used the proposed power grid design scheme to construct a power grid in the top two metal layers. The wires added to the grid, in each iteration, are in both horizontal and vertical directions in the top two metal layers. We assume equal pitches of the wires in horizontal and vertical directions within a partition, but, clearly this is not a restriction in the proposed scheme. The power pads are assumed to be connected to the top metal layer which has wires running only in the horizontal direction.

The same approach can be easily extended to design a power grid in multiple layers of metal. In the case of multiple metal layers, the partitioning step would comprise of introducing partition wires in multiple layers, and would result in partitions that would span multiple metal layers. In this case, the number of active partitions would be more than two and the wires added in the partitions would be added in each layer separately. The wires in different layers can be of different sizes as typically, the wires in top two metal layers are much wider than the ones in the intermediate metal layers. The step of checking for previous_grid_violated flags would now entail evaluation of port voltages of the neighbors of the active partitions in each layer. Other steps in the proposed design procedure remain the same while designing a multi-layered power grid.

6. EXPERIMENTAL RESULTS

The proposed power grid design scheme was implemented in C using a sparse matrix library [16], and design of several power networks were tested. The input to our power grid design procedure is a floorplan with functional block current estimates and the locations and number of the power pads on the chip. The output is a non-uniform power grid that meets the IR drop, EM and minimum pitch constraints. We could find only two real benchmark floorplans [17], [18] for a microprocessor chip in which the functional block currents could be determined. These are the floorplans of ALPHA 21364 microprocessor chip. The block currents of the functional blocks in these floorplans were estimated from the given power consumption estimates of each functional block, in 130nm technology, using a V_{dd} of 1.2V. The functional block current I_{f_k}, of a block k is computed as $I_{f_k} = \mathrm{Power}_k/V_{dd}$, where Power_k is the total power consumption of block k. Due to the paucity of real full chip level benchmark floorplans, with functional block current estimates, we randomly generated floorplans and assigned realistic block currents to various functional blocks in the floorplans. The block currents were assigned by assuming the total power consumption of the chips to be between 40-80 Watts and distributing the total power consumed randomly between the various functional blocks. For each of our experiments, we assume an availability of 400-600 power pads, distributed either throughout the chip, as in the case of a flip-chip package, or 200-300 pads located on the chip periphery, as in the case of a wire-bond package. The circuit parameter values, sheet resistivity (ρ_s), current density (σ) and minimum wire pitches (p_{min}), were taken from [19] and [20] for power delivery to a 2cm×2cm chip in 130nm technology with V_{dd} = 1.2V. The voltage constraints for the power grids, i.e., V_{spec} was 1.08V, i.e., 90% of V_{dd}. The experiments were performed on P-4 processor, Linux machines with a speed of 2.4 GHz and 2GB RAM.

We construct the power grid by the proposed scheme for a set of eight benchmark floorplans, both for a flip-chip (FC) and a wire-bond (WB) case. Table 1 shows the results for these power grid constructions. For each experiment for both the FC and the WB case, corresponding to one row in Table 1, the initial power grid in the first partitioning level comprises of very thick wires in the range of 60-100 μm. In subsequent partitioning levels, the width reduction factor, γ is assigned an appropriate value in the interval $[0.65, 1)$ so that at the end of k levels of partitioning, the final value of wire width for the power grid in 2^k partitions is between 2-6 μm. The design procedure is terminated at the end of $k = 7$ levels of partitioning. The value of MAX_CHG_{spec} parameter , to flag the previous_grid_violations, was chosen to be 15mV.

Ckt	# of Blocks	# of Nodes		Wire Area (cm^2)		Run Time (sec)	
		FC	WB	FC	WB	FC	WB
pg-1	17	1557504	1635841	0.0812	0.0852	443	661
pg-2	17	1185921	1216609	0.0783	0.0816	517	787
pg-3	12	1261129	1375929	0.0721	0.0754	653	839
pg-4	16	1050625	1207801	0.0688	0.0738	617	842
pg-5	20	1216609	1343281	0.0704	0.0806	572	805
pg-6	24	1136356	1199025	0.0722	0.0786	683	935
pg-7	20	1640961	1703025	0.0852	0.1022	431	692
pg-8	22	1292769	1364224	0.0840	0.0992	452	671

Table 1: Results of power grids designed by the proposed scheme for both flip-chip and wire-bond cases

The first two rows in Table 1 represent the power grid constructed for the two real benchmark floorplans of ALPHA 21364 chip. The other rows correspond to the power grid designed for the randomly generated floorplans. The second column in the table shows the number of blocks in the floorplan. The next two columns indicate the number of electrical nodes in the final optimized power grid circuit. For each circuit there are more than a million electrical nodes in the final circuit. The optimization is terminated when the worst voltage of all nodes in the final power grid circuit is greater than V_{spec} and all branches meet the current density specifications at the end of k levels of partitioning and the post processing step to align the wires. The worst voltage measured by performing an accurate simulation, at the end of the design procedure without the port approximation technique, verifies the accuracy of the final solution. The wire area consumed by the final power grid is listed in the fifth and the sixth column, for a FC and a WB case, respectively. The last two columns report the run time for constructing the power grid by the proposed design procedure. The run times of the proposed power grid design procedure are in the range of about 7-12 mins for the grids designed for the flip-chip case and about 11-16 mins for the wire-bond case. The order of the run times obtained underscores the efficiency of the design algorithm, considering the fact that for each of the test cases in Table 1, the final power grid for both the FC and the WB case, spanning the entire chip area in two layers of metal, comprises of more than a million electrical nodes.

As seen in Table 1, the proposed scheme performs better for the flip-chip case than the wire-bond case, both in terms of utilizing lesser wire area and faster run times. This can be ascribed to the fact that the notion of locality in power grid design, which is one of the motivating factors of the proposed algorithm, is more pronounced in the case of a flip-chip package, where there are sufficient number of pads near the violating regions. For a wire-bond chip, the fact that the pads around the chip periphery are located far way from the violating regions that may be located at the center of the chip, could make local grid correction step, for fixing the violations, a suboptimal choice. Hence, the procedure has to spend more time and wiring resources to meet the reliability constraints for a wire-bond chip.

Ckt	# of Wires Mgrid Scheme	Run Time (sec)		Wire Area (cm^2)		% Saving in Wire Area
		Prop Method	Mgrid Scheme	Prop Method	Mgrid Scheme	
Ckt-1	1000 × 1000	662	586	0.0701	0.0751	7.2%
Ckt-2	1100 × 1100	673	641	0.0741	0.0801	8.2%
Ckt-3	1150 × 1150	713	681	0.0766	0.0843	10.1%
Ckt-4	1200 × 1200	691	705	0.0759	0.0849	11.9%
Ckt-5	1250 × 1250	758	733	0.0793	0.0866	9.2%
Ckt-6	1300 × 1300	818	775	0.0840	0.0935	12.3%

Table 2: Results of power grids designed for FC circuits by the proposed method and the multigrid-based scheme of [5]

In another set of experiments, we compare the proposed power grid design algorithm with a previous grid design scheme [5]. We implemented a simple version of the multigrid-based power grid optimization scheme of [5] in C++ to compare the results of our proposed power grid design algorithm with this method. Table 2 shows a comparison of the performance of the proposed power grid design algorithm with the multigrid-based technique of [5]. The two schemes are used to design power grids for six randomly generated floorplans for a flip-chip case. The floorplans comprise of 20-40 functional blocks with currents assigned randomly to each block so that the total power consumption of the chip is between 40-80 Watts. Some functional blocks are assigned about 3-4 times more power than the other blocks so that there are distinct high and low current density regions on the chip. The assignment of block currents, to model the high and low current density regions, follows from the observation that most full-chip microprocessor floorplans have about 30%-50% of chip area dedicated to caches which consume much less power than the other functional blocks, e.g., arithmetic and logic units [18]. A uniform power grid is constructed for the six cases using the multigrid-based design scheme. The second column in the table 2 lists the number of horizontal and vertical wires used for the uniform grid construction. Following a series of network reductions, about 8-14 levels of reduction for each circuit, the top level power grid is reduced to a much smaller grid so that the problem size is sufficiently small. The wire sizing solution for reduced network is then obtained by solving a constrained non-linear optimization problem by using a sequential quadratic programing software [21]. The back-mapping to the original network is performed by solving a series of linear programs as formulated in [5].

The fourth and the fifth columns in the table show a comparison of the run time of the two schemes. For all the six circuit examples, the total time taken by the multigrid-based scheme to perform the network reduction, solve the non-linear optimization problem and solve a series of linear programs for back-mapping is of the same order of magnitude of the proposed power grid design algorithm. Columns six and seven show the wire area utilized for each of the example circuits by the two design methodologies. The wire area utilized by the proposed heuristic is about 7%-12% less than the grid design method of [5]. In the multigrid-based design method, each column (row) of vertical (horizontal) wire is constrained to have the same wire width in order to reduce the number of design variables for efficiently solving the resulting non-linear program. Since the width of all the wire segments on a column (row) of the wire is determined by the highest current density blocks, the power grid has to be over-designed in the low current density regions of the chip. The proposed design algorithm is run for $k = 10$ levels of partitioning so that the granularity of each partition is of the order of the block size. By designing local power grids in each partition, it is ensured that the wiring resources are utilized in a judicious manner as per the current density requirements. While the solution of the proposed algorithm is very accurate because of employing an explicit circuit analysis step, the inaccuracy in the solution by the multigrid-based method increases with increasing the number of reduction levels beyond a certain number.

7. CONCLUSIONS

In this paper, we have proposed a novel and efficient power grid design procedure. Our method is based on an iterative grid refinement scheme by recursive bipartitioning of the chip area. We use the concept of locality in power grid design to abstract away the details of some parts of power grid by the macromodeling technique. Using the grid abstractions, along with the strategy of constructing an initial coarse grid followed by a successive refinement of the grid, speeds up the circuit analysis step. Experimental results on real and randomly generated realistic test cases show that the proposed power grid design algorithm is considerably fast and has efficient utilization of the wiring resources.

8. REFERENCES

[1] X. D. S. Tan *et al*. Reliability-Constrained Area Optimization of VLSI Power/Ground Networks Via Sequence of Linear Programmings. In *IEEE Trans. on CAD*, volume 22, pages 1678–1684, Dec 2003.

[2] X. D. S. Tan and C. J. R. Shi. Fast Power/Ground Network Optimization Based on Equivalent Circuit Modeling. In *Proc. ACM/IEEE DAC*, pages 550–554, 2001.

[3] X. Wu *et al*. Area Minimization of Power Distribution Network Using Efficient Nonlinear Programming Techniques. In *Proc. IEEE/ACM ICCAD*, pages 153–157, 2001.

[4] T. Wang and C. C. Chen. Optimization of the Power/Ground Network Wire-Sizing and Spacing Based on Sequential Network Simplex Algorithm. In *Proc. IEEE ISQED*, pages 157–162, 2002.

[5] K. Wang and M. M-Sadowska. On-chip Power Supply Network Optimization using Multigrid-based Technique. In *Proc. ACM/IEEE DAC*, pages 113–118, 2004.

[6] M. Zhao *et al*. Optimal Placement of Power Supply Pads and Pins. In *Proc. ACM/IEEE DAC*, pages 165–170, 2004.

[7] J. Singh and S. S. Sapatnekar. Topology Optimization of Structured Power/Ground Networks. In *Proc. ACM ISPD*, pages 116–123, 2004.

[8] T. Mitsuhashi and E. S. Kuh. Power and Ground Network Topology Optimization for Cell Based VLSIs. In *Proc. ACM/IEEE DAC*, pages 524–529, 1992.

[9] H. Cai. Multi-pads Single Layer Power Net Routing in VLSI Circuit. In *Proc. ACM/IEEE DAC*, pages 183–188, 1988.

[10] J. Oh and M. Pedram. Multi-pad Power/Ground Network Design for Uniform Distribution of Ground Bounce. In *Proc. ACM/IEEE DAC*, pages 287–290, 1998.

[11] H. Su, K. H. Gala, and S. S. Sapatnekar. Fast Analysis and Optimization of Power/Ground Networks. In *Proc. IEEE/ACM ICCAD*, pages 477–480, 2000.

[12] H. H. Chen and D. D. Ling. Power Supply Noise Analysis Methodology for Deep-Sub-micron VLSI Chip Design. In *Proc. ACM/IEEE DAC*, pages 638–643, 1997.

[13] E. Chiprout. Fast Flip-Chip Power Grid Analysis via Locality and Grid Shells. In *Proc. IEEE/ACM ICCAD*, pages 485–488, 2004.

[14] M. Zhao *et al*. Hierarchical Analysis of Power Distribution Networks. In *Proc. ACM/IEEE DAC*, pages 482–486, 2000.

[15] H. Kriplani, F. Najm, and I. Hajj. Pattern Independent Maximum Current Estimation in Power and Ground Buses of CMOS VLSI Circuits : Algorithms, Signal Correlations, and Their Resolution. In *Proc. IEEE/ACM ICCAD*, pages 998–1012, 1995.

[16] http://www.netlib.org/c/meschach.

[17] K. Skadron *et al*. Temperature-Aware Microarchitecture. In *Proc. International Symposium on Computer Architecture*, 2003.

[18] W. Liao, L. He, and K. Lepak. Temperature-Aware Performance and Power Modeling. In *Technical Report UCLA Engr. 04-250*.

[19] Semiconductor Industry Association. International Technology Roadmap for Semiconductors, 2001. Available at `http://public.itrs.net`.

[20] J. Cong. An Interconnect-Centric Design Flow for Nanometer Technologies. In *Proc. IEEE*, volume 89, pages 477–480, 2000.

[21] C. Lawrence *et al*. User's gude for cfsqp version 2.4. Institute for Systems Research, University of Maryland, College Park, MD, Tech. Rep. TR-94-16rl, 1996.