SPROUT - Smart Power ROUting Tool for Board-Level Exploration and Prototyping

Rassul Bairamkulov
Department of Electrical and Computer Engineering
University of Rochester
Rochester, New York
rbairamk@ur.rochester.edu

Abinash Roy, Mali Nagarajan, and Vaishnav Srinivas
Qualcomm Technologies, Inc.
San Diego, California
{abinashr,mahaling,vaishnav}@qti.qualcomm.com

Eby G. Friedman
Department of Electrical and Computer Engineering
University of Rochester
Rochester, New York
friedman@ece.rochester.edu

2021 58th ACM/IEEE Design Automation Conference (DAC) | 978-1-6654-3274-0/21/$31.00 ©2021 IEEE | DOI: 10.1109/DAC18074.2021.9586128

***Abstract*—The board-level power network design process is governed by system-level parameters such as the number of layers and the ball grid array (BGA) pattern. These parameters influence the characteristics of the resulting system, such as power, speed, and cost. Evaluating the impact of these parameters is, however, challenging. To estimate the reduction in impedance if, for example, additional BGA balls are dedicated to the power delivery system, adjustments to the board layout and an additional impedance extraction process are required. These processes are poorly automated, requiring significant time and labor. Automating power network exploration and prototyping can greatly enhance the power delivery design process by increasing the number of possible design options. With power network exploration and prototyping, the effects of the system parameters on the electrical characteristics can be better understood, providing valuable insight into the early design stages. SPROUT - an automated algorithm for prototyping printed circuit board (PCB) power networks - is presented here. This tool includes the first fully automated algorithm for board-level power network layout synthesis. Two board-level industrial power networks are synthesized using SPROUT where a ball grid array is connected with a power management IC and decoupling capacitors across four voltage domains. The impedance of the resulting layouts is in good agreement with manual PCB layouts while requiring 95% less design time.**

I. Introduction

The increasing performance requirements of integrated systems rely on a stable power delivery system [1]. Violating power integrity specifications can produce a variety of issues ranging from performance degradation to a device malfunctioning. The robustness of integrated systems to voltage ripples is significantly less due to lower operating voltages and higher speeds [2]. On-chip current fluctuations can produce drops in the load voltage, jeopardizing system performance. The board-level power network is an important part of the power delivery system connecting a package or IC with the power management IC and on-board decoupling capacitors, critical to suppressing low frequency fluctuations [1].

A design flow for a board-level power delivery system is shown in Fig. 1. The quality and design time of a power network are influenced by system specifications, often set at the start of the development process. These specifications, such as the thickness of the metal layers, location and impedance characteristics of the decoupling capacitors, and the BGA ball arrangement, inform the floorplan and component placement process. After printed circuit board (PCB) floorplanning and component placement, the power rails are generated. An electrical model of the power network layout is extracted next. If the impedance of the power network is greater than the specified target impedance, the power network is iteratively adjusted until the target impedance is achieved. Changing system-level parameters after the impedance is extracted, however, typically requires significant time to manually modify the layout and repeat the impedance extraction and verification processes.

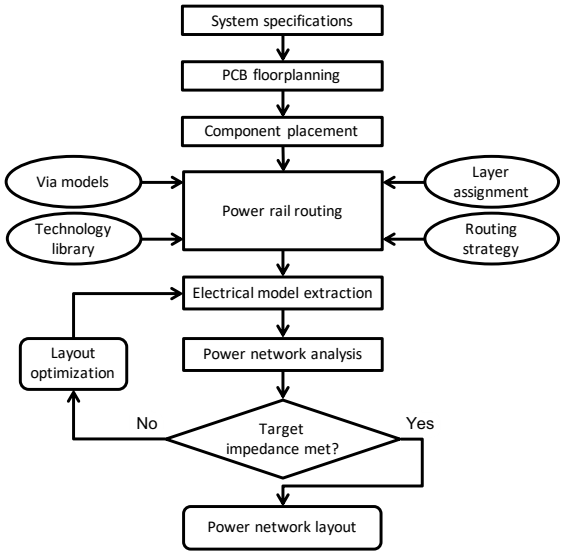

Fig. 1. Conventional design flow for power delivery networks for printed circuit boards.

Due to the lack of information during early stages of the design process, quantifying the effects of the system parameters on the power delivery system is complicated, increasing the likelihood of adjustments at later stages of the design process [3]. Incorrect system specifications, such as an insufficient number of decoupling capacitors, can lead to significant modifications in the layout, incurring significant time and labor during the iterative refinement process. Early exploration and analysis of layout prototypes can greatly enhance the power network design process by providing an estimate of the electrical performance of a system in response to a set of specifications.

Despite numerous studies describing board-level signal network synthesis [4], [5], automated board-level power network layout synthesis has received minimal attention in the literature. Existing research has focused on enhancing and guiding the power network analysis and manual layout processes. In [1], [6], [7], [8], [9], [10], for example, the decoupling capacitance and electromagnetic compatibility of a PCB are analyzed. In [11], [12], [13], [14], methods for predicting the electrical characteristics of a board-level power network are presented. Efficient methods for the analysis of existing PCB layouts are presented in [15], [16].

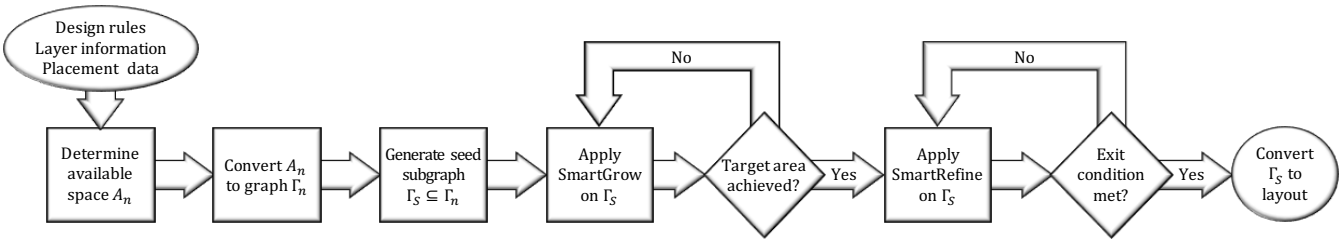

Fig. 2. Overview of SPROUT algorithm. The available space A_n is converted into an equivalent graph Γ_n. The subgraph seed Γ_n^S is generated by SPROUT and expanded using the SmartGrow algorithm described in section II-D. After achieving the target area, the nodes in Γ_n^S are rearranged using the SmartRefine algorithm to enhance the electrical characteristics. The final subgraph is converted into a physical layout.

The Smart Power ROUTing algorithm (SPROUT) for the exploration and synthesis of board-level power delivery network prototypes is presented here. Using the proposed algorithm, the multiple power nets from the power management IC (PMIC) are connected to the BGA balls and decoupling capacitors. The effective resistance of the resulting shapes is minimized, while obeying topological and electrical design rules. SPROUT enables early exploration and assessment of the effects of different system parameters on the electrical characteristics by producing a prototype of the PCB power network. Many system specifications can be evaluated by automatically generating a prototype for each set of system parameters. The tool greatly increases the efficiency of the power delivery design process by exploring performance-cost tradeoffs, thereby increasing the likelihood of satisfying the target impedances. Furthermore, the final PCB layout can be guided by the synthesized prototype, further enhancing the design process. In industrial case studies, similar electrical characteristics are achieved as compared with manual routing while significantly reducing the design time. The rest of the paper is organized as follows. The proposed power routing algorithm is described in section II. Validation of the algorithm on two industrial PCB systems is described in section III followed by the conclusions in section IV.

II. SPROUT ALGORITHM

A printed circuit board is composed of interleaved metal and dielectric layers. The connections between layers are provided by vias. Layer information, design rules, and placement data are inputs to the power routing process, as shown in the overview of the SPROUT algorithm depicted in Fig. 2. The thickness and material of each layer are specified as layer information. The placement data characterizes the floorplan with parameters such as the location and nets of the vias and components, reserved whitespace, and predefined polygons. Collectively, these inputs specify an initial floorplan. The objective of the power routing process is to produce a metal shape connecting a corresponding set of disjoint regions within a layout while minimizing the impedance and complying with geometric and electrical constraints. Importantly, the resulting layout is not a final topology used for fabrication but a synthesized exploratory prototype to estimate the impedance of the layout corresponding to the system level parameters.

In this paper, graph-based algorithms are utilized to route the power nets. A graph structure is frequently applied in routing VLSI systems [17], [18]. The routing process commences with a geometric analysis, as described in section II-A. The available space for each routed net A_n is determined using cell placement and geometric constraints. This available space is converted into a graph Γ_n, as described in section II-B. The source and target locations are identified and connected using the seed subgraph Γ_n^S, as discussed in section II-C. A subgraph is grown by utilizing the SmartGrow algorithm described in section II-D. The SmartRefine algorithm improves the electrical characteristics of the subgraph Γ_n^S based on the heuristics discussed in section II-E. The routing process is completed by converting the resulting subgraph into a physical layout, as described in section II-G.

A. Available routing space

The routing process starts with processing the input information, including the layer thickness and material characteristics, existing shapes, reserved whitespace, and location of the PMIC, inductors, capacitors, and BGA balls. The available routing space for a specific net is determined in three stages. The components, vias, and predefined shapes are initially converted into a set of polygons S. Each polygon s_i is characterized by four parameters, the net n_i, layer l_i, geometry g_i, and buffer b_i. During the routing process, the buffer area is not accessible by other nets, thereby ensuring no electrical connection exists between the different nets, as shown in Fig. 3b. To illustrate the interaction between the physical geometry and buffer, consider the example structure shown in Fig. 3. The three via pads have buffer zones that prevent other nets from being routed nearby. The placement of the connection shown in Fig. 3b is invalid since the V_{DD} connection intersects the V_{SS} buffer. To avoid these intersections, the connection is curved downward, as shown in Fig. 3c, producing a valid layout.

After the layout is converted into a set of buffered polygons, the available space A_n is determined for each net n. This available space is initially set to the full layout space U. Those blockage and buffer polygons belonging to different nets are removed from the space, as shown in Fig. 4,

$$A_n = U \setminus \bigcup_{n_j \neq n} b_j. \tag{1}$$

Operators $\setminus$ and $\bigcup$ denote, respectively, polygon subtraction and union operations. Efficient algorithms for polygon clipping

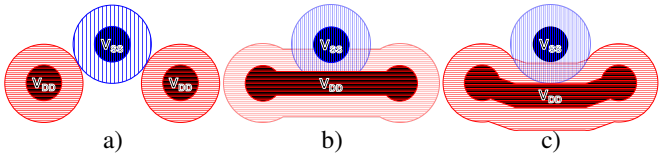

Fig. 3. One V_{SS} (vertical hatch) and two V_{DD} (horizontal hatch) via pads (dark), and buffers (light). a) Initial layout. b) The connection to the V_{DD} vias is invalid since the buffer around the V_{DD} connection overlaps the V_{SS} via, and the V_{DD} connection overlaps the V_{SS} via buffer. c) Example of valid routing. Neither the V_{DD} nor the V_{SS} buffer intersects with the vias or connections of a different net. Note that the V_{DD} connection can be placed in the buffer around the V_{DD} vias because both the via and connection belong to the same net.

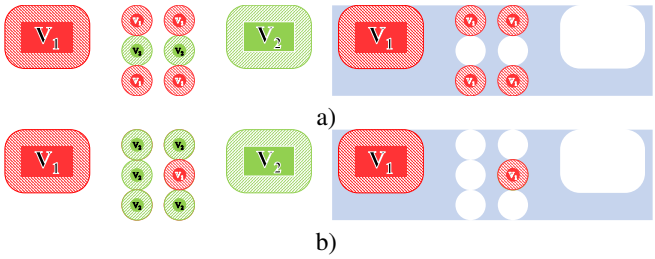

Fig. 4. Available space (shaded) for V_1 in two layouts. a) Layout (left) where routing from the pad on the left to four vias is possible, as evident from the connected available space (right). b) Layout (left) where connecting a pad with a via is not possible within a single layer due to the disjoint available space (right).

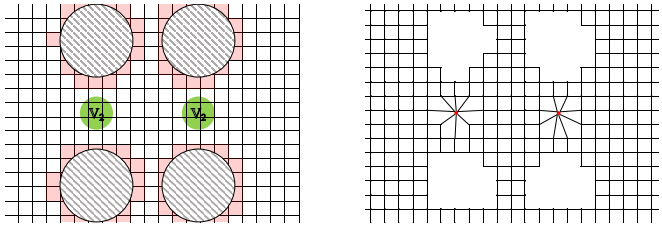

Fig. 5. Conversion of the available space for net V_2 into an equivalent graph. a) The available space is split into unit cells. Cells with irregular shapes are shaded. b) Equivalent graph. The tiles overlapping vias are treated as a single node. Nodes are not generated in prohibited areas.

are reported in [19], [20] and require negligible runtime in case studies.

The location of the terminal nodes is externally supplied using the set of routed terminals $T_n = \{t_n^1, \ldots, t_n^k\}$. After the removal process, the available space can become disjoint, leaving no valid path between terminals in T_n. If the routing terminals belong to different parts of a disjoint space, routing multiple layers utilizes existing vias or requires insertion of new vias (see Fig. 4b). In this scenario, the routing process is decomposed into several single layer routing steps.

B. Equivalent graph

To enable graph-based routing, the available space A_n is converted into an equivalent graph $\Gamma_n = (V_n, E_n)$. This available space is initially divided into tiles a_i. To repeat the contour of the boundary and avoid routing within prohibited areas, the boundary cells of A_n are irregular in shape, as shown in Fig. 5a. The shape of the cell is stored to consider the area of the routed shape. Note that the size of a_i is a design parameter determined from the routing specifications. A smaller cell size will produce smooth routed shapes, likely exhibiting superior impedance characteristics at the cost of significantly higher computational time. Each cell of the polygon is mapped into a vertex v_i of Γ_n.

Vertices corresponding to the adjacent cells are connected with edges. The weight of the edges represents the conductance between adjacent unit cells. Accurately determining the conductance requires costly analysis techniques such as the finite element method [21]. To minimize the effective resistance, a more efficient heuristic is proposed; specifically, the conductance is assumed to be proportional to the length of the overlap between cells.

The final step maps the set of polygons T_n into a set of corresponding terminal vertices $\Theta_n = \{\theta_1, \ldots, \theta_k\}$. Each polygon $t_i \in T_n$ is initially checked for intersections with vertices in V_n. The cells intersecting the polygon t_i are added to set θ_i. The nodes in each θ_i are identified and merged into a single terminal node, as shown in Fig. 5b.

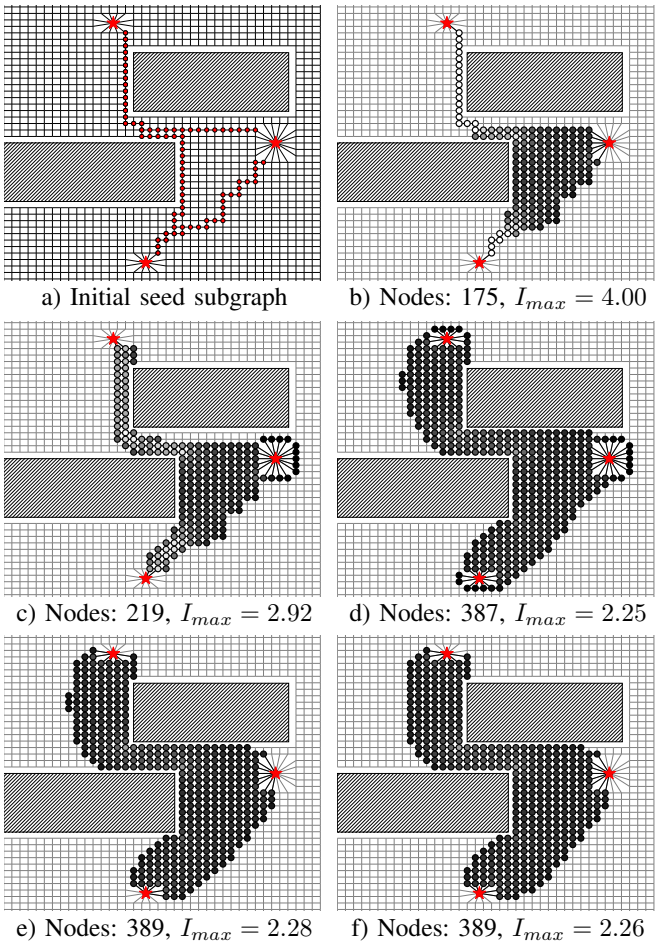

Fig. 6. Example of graph-based routing process between three terminals. a) Initial seed subgraph. b) Voidless subgraph after filling the internal voids. c) Initial stage of subgraph growth, and d) final stage of subgraph growth. The areas with large current are reinforced with new nodes. e) Initial stage of the refinement process. Areas with small current, specifically the nodes near the terminals, are replaced by the nodes in areas of current crowding, *i.e.*, closer to the obstacles. f) Final stage of the refinement process. The reduction in impedance is negligible and triggers termination of the algorithm.

C. Seed subgraph

Once the equivalent graph is generated for each routed net, the power routing process reduces to determining the subgraph $\Gamma_n^S = (V_n^S \subseteq V_n, E_n^S \subseteq E_n)$. The graph-based routing process starts with establishing an electrical connection between terminal nodes using a seed graph. Shortest path algorithms, such as Dijkstra [22] or Bellman-Ford [23], efficiently provide an initial seed. Once an initial connection is established, the seed graph is reinforced using the SmartGrow algorithm described in section II-D.

The process starts with determining the shortest path for each pair of nodes in Θ_n (see Fig. 6a). All nodes belonging to the shortest path are merged into a polygon. The vertices in Γ_n inside the exterior boundary of the polygon are included in Γ_n^S along with the corresponding edges, as shown in Fig. 6b.

Algorithm 1 Given available space graph Γ_n, seed subgraph Γ_n^S, and set of terminals $\Theta_n \in \Gamma_n^S$, add k nodes from Γ_n to Γ_n^S to reduce the impedance of the subgraph.

procedure SMARTGROW($\Gamma_n, \Gamma_n^S, \Theta_n, k$)
  $V_n^c \leftarrow V_n \setminus V_n^S$
  $[\Theta_n]^2 = \{\theta' \subseteq \Theta_n \mid |\theta'| = 2\}$
  $N_{pairs} \leftarrow \#[\Theta_n]^2$
  $I \leftarrow$ NODECURRENT(Γ_n^S, Θ_n)
  $I^c \in \mathbb{R}^{|V_n^c|}$
  for $p \in V_n^c$
    $I_p^c \leftarrow \sum_{j \in N(\Gamma_n, p), j \in \Gamma_n^S} |I_j|$
  for $i = 1, 2, \ldots, k$
    $V_n^S \leftarrow V_n^S \cup \{p \mid I_p^c = max(I^c)\}$
  $\Gamma_n^S \leftarrow G_n[V_n^S]$
  return Γ_n^S

D. Growth stage

The seed subgraph generated in the previous stage is highly resistive. The impedance between the terminal nodes can be reduced by adding nodes to the subgraph. To maximize the benefits from adding new nodes, the SmartGrow procedure is proposed (see Algorithm 1). Regions near nodes with low current are unlikely to benefit from additional nodes. Reinforcing those regions with large current results in a significantly lower impedance. To identify the regions with high and low current, a node current metric is proposed here.

The node current metric is evaluated in three stages. The set of nodes not belonging to Γ_n is initially $V_n^c = V_n \setminus V_n^S$. The next stage evaluates the current metric. A unit current is injected into each pair of terminals in Θ_n. The voltage distribution resulting from each pair is

$$V = L^{-1}E, \tag{2}$$

where $V \in R^{(n-1)\times n_{pairs}}$ is the matrix of node voltages, $L \in R^{(n-1)\times(n-1)}$ is the grounded Laplacian matrix, and $E \in R^{(n-1)\times n_{pairs}}$ is the matrix of current injection with 1 and -1 at, respectively, the source and ground terminals. The voltage distribution corresponding to each pair of terminal nodes determines the current through each edge of the subgraph. The current metric of each node is the sum of the absolute value of the currents in adjacent edges,

$$I_p = \sum_{i=1}^{n_{pairs}} \sum_{j \in N(\Gamma_n^S, c)} \frac{|v_{ci} - v_{ji}|}{r_{cj}}, \tag{3}$$

where $N(\Gamma_n^S, p)$ is the set of nodes adjacent to node $c \in \Gamma_n^S$, and r_{cj} is the resistance between nodes c and j.

The final step of the growth process is reinforcement of those areas with high current. The sum of node currents in nodes adjacent to node $p \in V_n^c$ is

$$I_p^c = \sum_{j \in N(\Gamma_n, p), j \in \Gamma_n^S} |I_j|. \tag{4}$$

Those nodes with the largest I_p^c are likely neighbors of the areas with large node current and are therefore added to the set of vertices V_n^S. The result of the algorithm is a vertex induced subgraph $G_n[V_n^S]$, *i.e.*, the subgraph with edges in Γ_n whose endpoints are both in V_n^S. This process repeats until the desired area is achieved. An example of current distribution within a subgraph is illustrated in Figs. 6b and 6d. Note that new nodes are placed close to those nodes with high current, whereas regions with low current are not reinforced.

Algorithm 2 Given available space graph Γ_n, subgraph Γ_n^S, and set of terminals $\Theta_n \in \Gamma_n^S$, replace k nodes in Γ_n^S with k nodes from Γ_n to reduce the impedance of the subgraph.

procedure SMARTREFINE($\Gamma_n, \Gamma_n^S, \Theta_n, k$)
  $I \leftarrow$ NODECURRENT(Γ_n^S, Θ_n)
  for $i = 1, 2, \ldots, k$
    $m \leftarrow \{p \mid p \in V_n^S, I_p = min(I)\}$
    $V_n^S \leftarrow V_n^S \setminus m$
    $I \leftarrow I \setminus m$
  $\Gamma_n^S \leftarrow$ SMARTGROW($\Gamma_n, \Gamma_n[V_n^S], \Theta_n, k$)
  return Γ_n^S

E. Refinement stage

Despite significant improvements in the impedance of subgraph Γ_n^S during the growth stage, the resulting subgraph can be improved without increasing the area of the shape. Those regions carrying the least current can be removed and replaced by nodes near hot spots, as described in Algorithm 2.

The current metric of the nodes in Γ_n^S is determined using (2) and (3). Those nodes with the smallest current capacity are removed, and the nodes in Γ_n adjacent to the peripheral nodes with the highest current are added to subgraph Γ_n^S. The result of this process is shown in Figs. 6e and 6f. Note that the peak current of the shape is reduced while maintaining the same area.

The number of removed peripheral nodes is a variable parameter set by the designer. A higher number will more quickly converge to a low impedance. However, at later stages of the refinement process, moving a large number of nodes can increase the impedance. Gradually reducing the number of moved nodes to achieve a lower impedance within a target area is therefore recommended.

Viewing the subgraph refinement process from an optimization perspective, the problem can be viewed as

$$\textbf{Minimize} : R(\Gamma_n^S, \Theta_n) \tag{5}$$

$$\textbf{s.t.} : A(\Gamma_n) \leq A_{max}, \tag{6}$$

where $R(\Gamma_n^S, \Theta_n)$ is the impedance of subgraph Γ_n^S with respect to nodes in Θ_n, $A(\Gamma_n)$ is the total area of the polygons in Γ_n, and A_{max} is the area constraint. While incremental changes to the subgraph Γ_n^S lower the impedance of the subgraph, as described in the previous sections, achieving a globally optimal solution using this method is not guaranteed. A reheating technique is therefore proposed in subsection II-F to reduce the probability of converging into a locally optimal subgraph.

F. Subgraph reheating

Most global optimization algorithms use techniques that lower the likelihood of converging into a local minimum. In simulated annealing [24], for example, a higher value of the objective function is chosen with a certain probability to explore the search space. A similar forced exploration procedure is proposed here for subgraph optimization.

The reheating process consists of two stages, dilation and erosion, inspired by image processing operations where the target image is expanded and shrunk by adding and removing nodes along the perimeter of the image [25]. The nodes in V_n^c adjacent to the target subgraph Γ_n^S are initially added to Γ_n^S. Using a node current metric, those nodes with a lower current capacity are removed from the subgraph. Increasing the size of the subgraph and removing nodes can yield a lower impedance since a larger number of different layouts is explored.

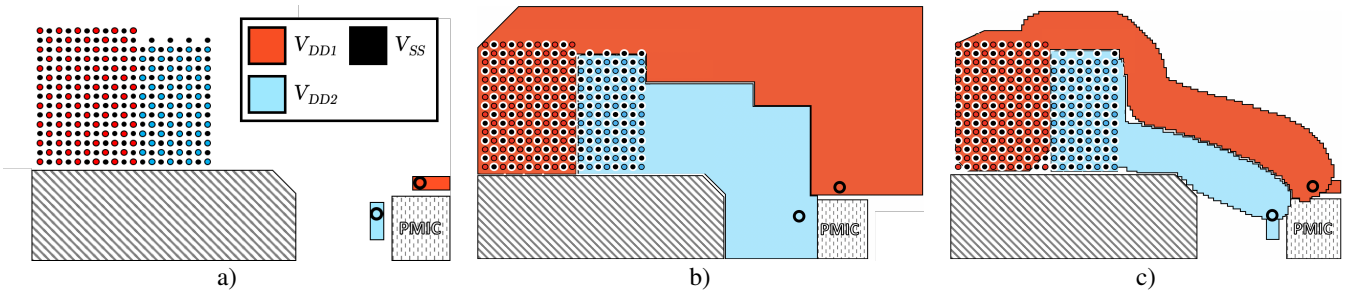

Fig. 7. Automated power routing using SPROUT and manual routing. a) Initial layout. A single PMIC supplies power to rails using two inductors at bottom layer 8. The inductors are connected to routing layer 7 using a via. Blockage is shaded with a diagonal pattern. b) Manually routed layout. c) Layout synthesized using SPROUT

G. Back conversion

A final step in the subgraph optimization process is conversion of the subgraph into a polygon. During creation of Γ_n, each node is recorded as a polygon. Conversion of the subgraph Γ_n^S into a layout is therefore a straightforward placement of the node polygons in Γ_n^S within the layout space. Those nets routed after net n cannot be routed too close to the routed polygon. This polygon is therefore removed from the available space when other nets are routed.

III. Validation of Case Study

Two practical case studies are presented in this section to demonstrate the validity of the proposed tool. In the first case, described in subsection III-A, the layout between the PMIC and two groups of vias is synthesized in a constrained layout. In the second case, described in subsection III-B, connections between the PMIC, capacitor, and a congested group of vias are established for six nets.

A. Two rail system

A portion of a PCB for an industrial wireless application in shown in Fig. 7a. The PCB consists of eight layers. BGA balls are located at the top layer, whereas the PMIC is located at the bottom layer and supplies power to the two power rails, V_{DD1} and V_{DD2}. Each BGA ball is connected to a layer using a dedicated full stack via. The power rails are routed on the seventh (penultimate) layer from the PMIC inductor to the group of BGA vias. Layers two, six, and eight are dedicated ground planes.

The results of the SPROUT synthesis and manual routing processes are shown in, respectively, Figs. 7b and 7c. Regular geometries are primarily utilized in the manual layout. In contrast, the synthesized layout exhibits greater flexibility in the shape of the geometries. The resistance and inductance of the layouts are extracted using a commercial electromagnetic solver tool, as listed in Table I. Note the similarity of the impedance characteristics of the two layouts. By using SPROUT, the inductance of the V_{DD1} rail is reduced by 12%, whereas the inductance of the V_{DD2} rail is increased by 1.47%. The difference in resistance does not exceed 3.1%.

TABLE I
Comparison of normalized impedance between SPROUT and manual routing for the two rail system shown in Fig. 7

	Net	Manual	SPROUT
Normalized inductance @ 25 MHz (picohenrys)	V_{DD1}	100	87.5
	V_{DD2}	136	138
Normalized DC Resistance (milliohms)	V_{DD1}	10.0	10.1
	V_{DD2}	12.7	13.1

B. Six rail system

An example layout of an industrial PCB for a wireless application is shown in Fig. 8. The power rails are routed within a ten layer PCB containing 612 BGA (six power supply nets and 306 BGA for ground). The ninth layer is assigned for routing the power rails whereas layers four, six, and eight are used for ground routing. Each BGA ball is connected to a layer using a dedicated full stack via. Two PMICs are located in the bottom layer. Each PMIC regulates current for three voltage domains.

The six power supply rails are routed and compared to manual layouts. The resulting topologies are shown in Figs. 8b and 8c. Note the visual similarity between the layouts. The DC resistance and loop inductance of each rail are compared in Table II. The loop inductance of the rails generated by SPROUT are 1 to 4% smaller than the manual layout while the difference in DC resistance is below 11%.

IV. Conclusions

The power network design process at the board level is highly influenced by system-level parameters such as the BGA pattern, layer specifications, and placement of the individual components. Changing a floorplan if a target impedance is not satisfied significantly degrades the speed of the development process. To increase the likelihood of satisfying design objectives, the system-level parameters are evaluated to determine appropriate tradeoffs among power, performance, and design time. To accelerate this evaluation process, SPROUT, an automated routing algorithm for power network exploration and prototyping, is introduced here. Based on the node current metric described in this paper, a layout of a power network suitable for impedance extraction is automatically synthesized.

The primary contribution of SPROUT is automation of layout prototypes, enhancing exploration of the design space. By

TABLE II
Comparison of normalized impedance between SPROUT and manual routing for the six rail system shown in Fig. 8

	Net	Manual	SPROUT
Normalized inductance @ 25 MHz (picohenrys)	V_{DD1}	133	131
	V_{DD2}	103	99
	V_{DD3}	131	127
	V_{DD4}	161	155
	V_{DD5}	152	150
	V_{DD6}	116	114
Normalized DC Resistance (milliohms)	V_{DD1}	15.0	16.8
	V_{DD2}	8.4	9.1
	V_{DD3}	13.0	14.2
	V_{DD4}	18.4	18.2
	V_{DD5}	18.5	18.9
	V_{DD6}	9.2	9.2

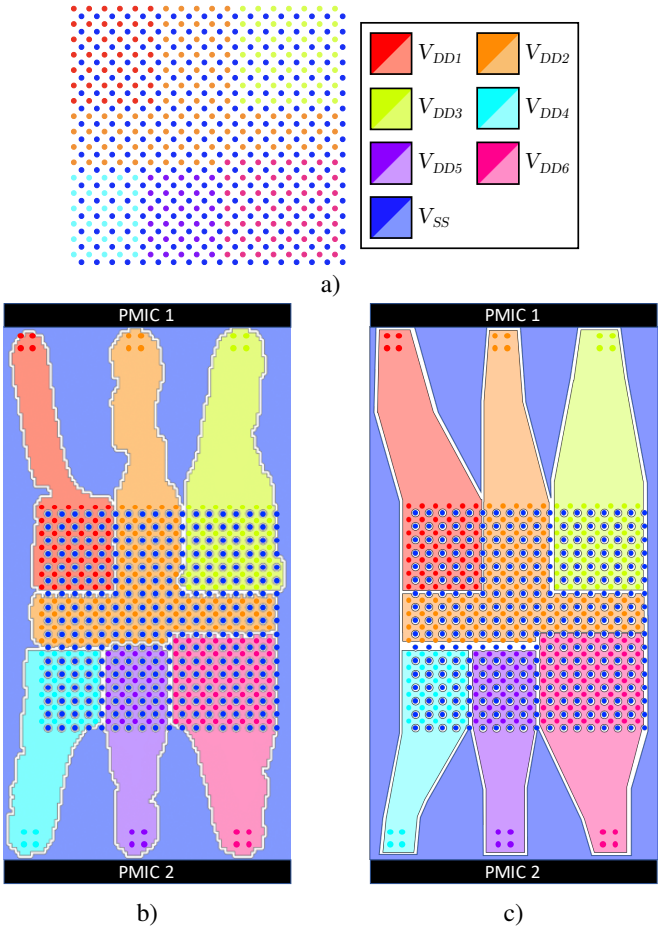

Fig. 8. Comparison between the automated power routed layout using SPROUT and manually routed layout. a) BGA placement, b) layout synthesized using SPROUT, and c) manual layout.

providing greater insight into the layout at early stages of the design process, system parameters can be more accurately determined, reducing the likelihood of not satisfying target impedance objectives. The tool is demonstrated on two industrial wireless applications, routing two and six power rails in a PCB layout. The impedance of the generated layout is in good agreement with manual layout, achieving a 4% average difference in impedance in the two case studies. Furthermore, the rail synthesis process is completed, on average, twenty times faster than a manual routing process, providing significant savings in both time and labor.

REFERENCES

[1] I. Partin-Vaisband, R. Jakushokas, M. Popovich, A. V. Mezhiba, S. Köse, and E. G. Friedman, *On-Chip Power Delivery and Management*, *Fourth Edition*, Springer International Publishing, 2016.

[2] K. Shringarpure, B. Zhao, L. Wei, B. Archambeault, A. Ruehli, M. Cracraft, M. Cocchini, E. Wheeler, J. Fan, and J. Drewniak, "On Finding the Optimal Number of Decoupling Capacitors by Minimizing the Equivalent Inductance of the PCB PDN," *Proceedings of the IEEE International Symposium on Electromagnetic Compatibility*, pp. 218–223, September 2014.

[3] R. Bairamkulov, K. Xu, M. Popovich, J. S. Ochoa, V. Srinivas, and E. G. Friedman, "Power Delivery Exploration Methodology Based on Constrained Optimization," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 39, No. 9, pp. 1916–1924, September 2020.

[4] S.-C. Fang, K.-E. Chang, W.-S. Feng, and S.-J. Chen, "Constrained Via Minimization with Practical Considerations for Multi-Layer VLSI/PCB Routing Problems," *Proceedings of the ACM/IEEE Design Automation Conference*, pp. 60–65, June 1991.

[5] T. Yan, Q. Ma, and M. D. F. Wong, "Advances in PCB Routing," *IPSJ Transactions on System LSI Design Methodology*, Vol. 5, pp. 14–22, February 2012.

[6] M. Popovich, M. Sotman, A. Kolodny, and E. G Friedman, "Effective Radii of On-Chip Decoupling Capacitors," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 16, No. 7, pp. 894–907, July 2008.

[7] B. Archambeault, M. Cocchini, G. Selli, J. Fan, J. L. Knighten, S. Connor, A. Orlandi, and J. Drewniak, "Design Methodology for PDN Synthesis on Multilayer PCBs," *Proceedings of the IEEE International Symposium on Electromagnetic Compatibility*, pp. 1–6, August 2008.

[8] J. Fan, J. L. Drewniak, J. L. Knighten, N. W. Smith, A. Orlandi, T. P. Van Doren, T. H. Hubing, and R. E. DuBroff, "Quantifying SMT Decoupling Capacitor Placement in DC Power-Bus Design for Multilayer PCBs," *IEEE Transactions on Electromagnetic Compatibility*, Vol. 43, No. 4, pp. 588–599, August 2001.

[9] T.-L. Wu, H.-H. Chuang, and T.-K. Wang, "Overview of Power Integrity Solutions on Package and PCB: Decoupling and EBG Isolation," *IEEE Transactions on Electromagnetic Compatibility*, Vol. 52, No. 2, pp. 346–356, July 2010.

[10] T. Hubing, "PCB EMC Design Guidelines: a Brief Annotated List," *Proceedings of the IEEE Symposium on Electromagnetic Compatibility*, pp. 34–36, August 2003.

[11] J. Mohamed, T. Michalka, S. Ozbayat, and G. R. Luevano, "PDN Design and Sensitivity Analysis using Synthesized Models in DDR SI/PI Co-Simulations," *Proceedings of the IEEE Electrical Design of Advanced Packaging and Systems Symposium*, pp. 1–3, December 2018.

[12] S. Yang, Y. S. Cao, H. Ma, J. Cho, A. E. Ruehli, J. L. Drewniak, and E. Li, "PCB PDN Prelayout Library for Top-Layer Inductance and the Equivalent Model for Decoupling Capacitors," *IEEE Transactions on Electromagnetic Compatibility*, Vol. 60, No. 6, pp. 1898–1906, August 2017.

[13] Y. S. Cao, T. Makharashvili, J. Cho, S. Bai, S. Connor, B. Archambeault, L. Jiang, A. E. Ruehli, J. Fan, and J. L. Drewniak, "Inductance Extraction for PCB Prelayout Power Integrity using PMSR Method," *IEEE Transactions on Electromagnetic Compatibility*, Vol. 59, No. 4, pp. 1339–1346, August 2017.

[14] A. V. Mezhiba and E. G. Friedman, "Inductive Properties of High-Performance Power Distribution Grids," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 10, No. 6, pp. 762–776, December 2002.

[15] B. Zhao, C. Huang, K. Shringarpure, J. Fan, B. Archambeault, B. Achkir, S. Connor, M. Cracraft, M. Cocchini, A. Ruehli, and J. Drewniak, "Analytical PDN Voltage Ripple Calculation Using Simplified Equivalent Circuit Model of PCB PDN," *Proceedings of the IEEE Symposium on Electromagnetic Compatibility and Signal Integrity*, pp. 133–138, March 2015.

[16] S. Sun, D. Pommerenke, J. L. Drewniak, K. Xiao, S.-T. Chen, and T.-L. Wu, "Characterizing Package/PCB PDN Interactions from a Full-Wave Finite-Difference Formulation," *Proceedings of the IEEE International Symposium on Electromagnetic Compatibility*, Vol. 2, pp. 550–555, August 2006.

[17] V. F. Pavlidis and E. G. Friedman, "Timing-Driven Via Placement Heuristics for Three-Dimensional ICs," *Integration, The VLSI Journal*, Vol. 41, No. 4, pp. 489–508, July 2008.

[18] J. K. Ousterhout, G. T. Hamachi, R. N. Mayo, W. S. Scott, and G. S. Taylor, "Magic: A VLSI Layout System," *Proceedings of the ACM/IEEE Design Automation Conference*, pp. 152–159, June 1984.

[19] Günther Greiner and Kai Hormann, "Efficient Clipping of Arbitrary Polygons," *ACM Transactions on Graphics*, Vol. 17, No. 2, pp. 71–83, April 1998.

[20] B. R. Vatti, "A Generic Solution to Polygon Clipping," *Communications of the ACM*, Vol. 35, No. 7, pp. 56–63, July 1992.

[21] A. Hrennikoff, "Solution of Problems of Elasticity by the Framework Method," *Journal of Applied Mechanics*, Vol. 8, No. 4, pp. 169–175, December 1941.

[22] S. Peyer, D. Rautenbach, and J. Vygen, "A Generalization of Dijkstra's Shortest Path Algorithm with Applications to VLSI Routing," *Journal of Discrete Algorithms*, Vol. 7, No. 4, pp. 377 – 390, December 2009.

[23] S. H. Gerez *et al.*, *Algorithms for VLSI Design Automation*, Vol. 8, Wiley, 1999.

[24] S. S. Gill, R. Chandel, A. Chandel, and P. S. Sandhu, "Simulated Annealing Based Delay Centric VLSI Circuit Partitioning," *Proceedings of the International Conference on Computer Science and Information Technology*, Vol. 1, pp. 1–4, July 2010.

[25] J. Y. Gil and R. Kimmel, "Efficient Dilation, Erosion, Opening, and Closing Algorithms," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 24, No. 12, pp. 1606–1617, December 2002.