Constraint-Aware Algorithms for Heterogeneous Power Module Layout Synthesis and Reliability Optimization

Imam Al Razi [a], Quang Le [b], H. Alan Mantooth [b], Yarui Peng [a]
[a] Computer Science and Computer Engineering Department,[b] Electrical Engineering Department
University of Arkansas, Fayetteville, AR, US
ialrazi@email.uark.edu, yrpeng@uark.edu

***Abstract*—A constraint-aware layout engine is developed for PowerSynth to explore heterogeneous power module layout synthesis and optimization considering reliability. For this purpose, the corner stitching data structure with constraint graph evaluation is extended for power module layouts with generic, scalable, and efficient algorithms to place and route heterogeneous components including active devices, sensors, controllers, and other passive components. Unlike VLSI, in power modules design, layout compaction is not the optimum target because of thermal and reliability issues associated with high voltage and current. Therefore, in this layout engine, both design and reliability constraints are honored while generating layout solutions by evaluating constraint graphs and randomizing edge weights. Compared with existing work, the proposed algorithms can process a broader range of layouts with a higher geometrical complexity within a few minutes. In addition, the produced layouts are both reliable and design-rule-check (DRC)-clean, which improves both time complexity and layout quality.**

***Keywords*—PowerSynth, corner stitch, constraint graph, Multi-Chip Power Module, algorithms, layout optimization.**

I. Introduction

Recently, inspired from VLSI, power module layout design automation is gaining attention as power module layout has been deemed key to realizing the maximum performance for wide band gap (WBG) technologies (i.e., GaN and SiC). The existing design procedures are performed manually using iteration-based sequential design steps that result in local optimum solutions rather than the global one [1, 2]. Recent studies try to extend VLSI placement-and-routing (P&R) concepts into power module layout design automation [3–5]. However, there are some fundamental differences [3] in the design procedure between VLSI and power module layouts because of thermal and reliability issues.

The sequence pair method is used in [3] for representing and optimizing placement of components in a power module layout. Routing is simplified by combining signal and power traces with devices and considering the entire device as a single component. This simplification leads to reduced flexibility and may give worse results compared to designs with detailed routing. For routing optimization, a 1-D binary string is produced randomly by applying a uniform binary distribution. This randomization approach leads to iteratively checking electrical connectivity and is a potentially time-consuming step.

PowerSynth [5] is a Multi-Chip Power Module (MCPM) layout synthesis tool that exploits multi-objective optimization to generate optimized layout solutions automatically. To begin with, the tool takes layer stack information including layout dimensions, material properties and simple system specification parameters such as ambient temperature and switching frequency, etc. This information serves for the reduced order models characterization process for both electrical and thermal as described in [5, 6]. These models provide a high prediction accuracy with an acceptable computational time during the optimization process. Along with the layer stack information, an abstract layout representation namely symbolic layout is used as the input. This symbolic layout consists of lines and points representing traces, devices or lead connections, whose coordinates are normalized. Then, each trace or component is stamped into a matrix representation to effectively update the trace dimension and component locations during the optimization process. However, this layout generation method does not consider any design constraints during the layout generation phase. Therefore, once a layout is generated, it has to go through an iterative DRC checking process. In some complicated layout cases more than 90% layout solutions are discarded due to DRC failures. Also, this DRC step could be time-consuming on 3-D layouts with many heterogeneous components and limits geometric configurations due to many built-in assumptions. To address these limitations, a new constraint-aware layout engine is required to handle higher geometrical complexity with generic approach.

Both studies mentioned above use the genetic algorithm for finding optimal solution sets using different evaluation metrics. In [3], parasitic inductance, resistance, and footprint area are considered as layout performance metrics, whereas in [5], electrical (parasitic inductance, resistance, capacitance) and thermal (average and maximum temperature) parameters are used. Both methodologies consider only homogeneous power systems consisting of only power transistors and diodes rather than heterogeneous layouts with components such as passive elements (capacitor), gate drivers, and EMI filters. To explore a larger solution space with enhanced flexibility, these heterogeneous components should also be optimized with power modules simultaneously. Therefore, to the best of our knowledge, there is no existing work on automatic power module layout optimization considering heterogeneous components, design constraints, and reliability constraints with a scalable and generic algorithm.

Furthermore, recent improvements in wide band gap device technologies have led to higher-voltage-rating applications ranging from 2.5 kV to 15 kV [7–10]. While these improvements allow more compact layouts and a higher power density, in recent studies more attention is drawn to reliability metrics such as creep/strike and partial discharge (PD) [11–14]. One of the solutions to improve reliability is applying insulating materials with high dielectric strength between traces [14]. While this method prevents PD activity, a lot of care is taken to choose an appropriate material for good thermal and mechanical reliability. Another solution is simply increasing the gaps between traces to ensure a safe PD distance. For instance, in [9], a 10 kV 120 A SiC Half-bridge module has been designed based on a commercial Si IGBT module footprint to ensure a reliable trace distance. Similarly, a study on a 15 kV bridge rectifier [10] has shown

This material is based on work supported by The National Science Foundation under Grant No. EEC-144954804 and Army Research Lab Contract No. W911NF1820087. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of the National Science Foundation and Army Research Lab.

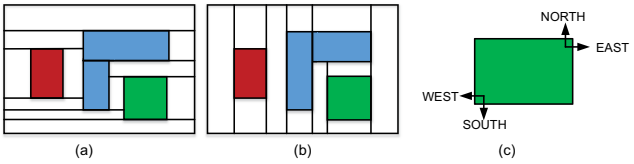

Fig. 1. (a) Horizontal corner-stitched plane, (b) vertical corner-stitched plane, and (c) a corner stitch tile.

that the pad spacing has to be greater than 3 mm to ensure a safe partial discharge threshold. Based on these design practices, both design and reliability constraint-aware algorithms are proposed in this paper for generating DRC-clean layout solutions with voltage-and-current-dependent reliability constraints.

In this paper, reliability-aware physical design algorithms are proposed to generate a broader range of heterogeneous layouts with a higher geometrical complexity. For this purpose, the corner stitching data structure [15] with the constraint graph [16] evaluation methodology has been extended for power module layouts. All common technology constraints such as minimum width, minimum spacing, minimum enclosure, minimum extension and user-defined reliability constraints such as voltage-dependent spacing and current-dependent trace width are preserved in constraint graphs. From corner-stitched layouts, all of these constraints are mapped into constraint graphs to maintain horizontal and vertical correlations among components. The graphs are evaluated using the longest path algorithm that is widely used in physical design algorithms in VLSI [17]. These solutions are further optimized using cost functions including electrical parameters and thermal metrics to generate a Pareto-front of optimal layouts. The proposed algorithms have four modes of operations for enhanced flexibility in handling heterogeneous components. Our study demonstrated that the proposed methodology with planar data structures is preferable for heterogeneous power module layout optimization with improved reliability, flexibility, scalability, and efficiency.

II. Corner Stitching and Constraint Graph

The corner stitching data structure is widely used in the Magic VLSI tool [18] for layout representation. Constraint graphs are very popular in VLSI floorplan compaction problems. A short review of corner stitching and constraint graph is presented here.

A. Corner Stitching

The corner stitching data structure is used to represent non-overlapping rectangles called tiles. Two important features make corner stitching better than others: a) both empty and occupied areas are represented explicitly, and b) each tile contains four pointers to preserve neighboring information. Using two top right and two bottom left corner pointers (shown in Fig. 1 (c)) the whole layout area can be traversed efficiently . To find horizontal and vertical constraints properly from corner stitched layouts, two corner stitched layouts (shown in Fig. 1 (a), and (b)) are created using maximal horizontal and vertical rules respectively. The rules are: a) each tile must be as wide (tall) as possible, b) after satisfying rule (a) each tile must be as tall (wide) as possible. Due to the linear time complexity of related operations such as point finding, tile creation, area searching and the convenience of obtaining necessary design constraints, we have found this elegant data structure to be best suited for power module layout representation.

B. Constraint Graph

There are two graphs: horizontal constraint graph (HCG) and vertical constraint graph (VCG) that encode the set of constraints, one for horizontal constraints and one for vertical constraints. In the constraint graph each vertex represents a coordinate in the corner-stitched layout. Each edge between any pair of vertices represents the relative locations between them. These graphs are weighted directed acyclic graphs (DAGs), where weights represent minimum constraint values. For example, if there is an edge between vertex i and j with the weight w_{ij}, it reflects the inequality:

$$j \quad i \quad w_{ij} \tag{1}$$

In VLSI floorplan compaction problems, these graphs are evaluated using the longest path algorithm to have the minimum-sized layout without DRC violations [19]. But in the case of power modules, the evaluation algorithms have to be designed such that not only the minimum sized layout, but also variable sized layouts can be produced to address thermal and reliability issues.

III. Methodology

A. Existing Methodology

The existing layout engine of PowerSynth takes a symbolic layout as input of the power module consisting of lines and points. The engine converts the symbolic layout data into a 2-D matrix. Each entry in the matrix has a list of pointers to the layout objects. From this matrix, a design parameter list is derived and used for optimization. The number of design parameters in the layout engine is restricted by definition. So, this layout engine has less flexibility in layout variation. All the gaps among components are fixed, and each component has a one-dimensional variable width to be either horizontal or vertical. The most important issue is with the design rule checking. This layout engine checks each design rule one-by-one after layout generation, which is not efficient as many solutions are discarded. Also, it can only generate fixed-sized layout, which limits the solution space.

B. Proposed Methodology

1) Data Structure: Due to inefficiency and restrictions with the matrix-based methodology, we propose the corner stitching data structure with constraint graph methodology for physical design of heterogeneous power modules. In the proposed methodology, initial layout information is taken from the user to generate horizontal and vertical corner-stitched layouts. Tile insertion function uses point finding, splitting, merging functions to insert a new component. Here, each component is represented as a rectangle of individual type. Therefore, heterogeneous components can be easily represented and the number of components are not bounded, which ensures the scalability of the proposed data structure. As the corner stitching data structure is a planar data structure originally designed for VLSI, the basic version does not allow overlapping of tiles. However, overlapping is a must for proper representation of the layout structure of power module, because devices (such as power FETs and diodes) are normally placed on top of traces. Therefore, the basic tile insertion function has been modified to allow overlapping of tiles.

2) Incorporated Constraints: For power module design, two types of constraints are considered: design constraints and reliability constraints. Design constraints are minimum constraints imposed by the technology, to ensure proper fabrication of the module. Reliability constraints are considered to address issues related to high voltage and current such as partial discharge and thermal.

i. Design Constraints: The following design constraints are considered to have DRC-clean layouts. An illustration of the constraints are shown in Fig. 2.

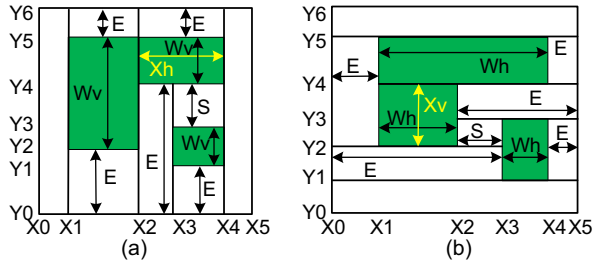

Fig. 2. Design constraints from (a) vertical, and (b) horizontal corner-stitched layout.

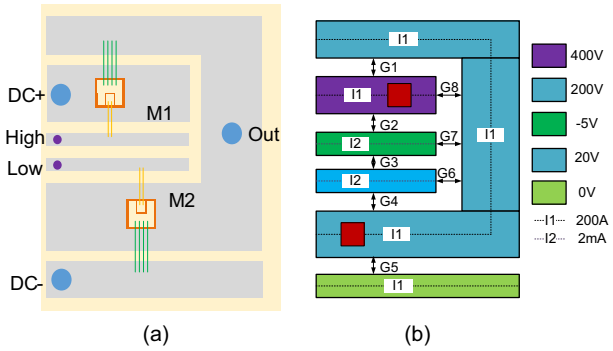

Fig. 3. (a) Half bridge power module layout and (b) associated reliability constraints: G1 through G8 are voltage-dependent minimum spacing, and traces have current-dependent minimum width.

Minimum Width: Minimum width is associated with each component in the power module. Some components can have a different minimum width along x and y axis. All horizontal widths are taken from horizontal corner stitch and vertical widths are taken from vertical corner stitch.

Minimum Spacing: Minimum spacing value is considered between two components. When there are multiple components in between same vertices maximum value determines the spacing to ensure DRC-validity. All horizontal spacing and vertical spacing are taken from horizontal corner stitch and vertical corner stitch respectively.

Minimum Enclosure: To ensure proper connectivity, some components are required to be surrounded by some other components underneath, with a spacing known as minimum enclosure. For example, when a device is placed on top of a trace, there should be a minimum enclosure of the trace to the device.

Minimum Extension: In some cases, there may be L-shaped or T-shaped components, where one leg extends in the direction perpendicular to the components routing direction. For those cases, the minimum extension rule appears. Horizontal extensions can be found from vertical corner stitch, whereas vertical extensions can be found from horizontal corner stitch.

ii. Reliability Constraints: The following reliability constraints are considered to minimize partial discharge, current crowding, field focusing and increase the reliability of the power module. An illustration of the constraints are shown in Fig. 3.

Minimum Width: In a power module, power traces can carry very high current, whereas signal traces carry very low current. So, for power and signal traces, there should be different minimum widths. As these minimum widths are current-dependent, user-defined minimum width rule is considered in these cases. Between two minimum widths (design constrained and reliability constrained), the higher value is applied to the graph as the corresponding edge weight. For example in Fig. 3, two current loadings require two different minimum trace widths apart from the design constraints.

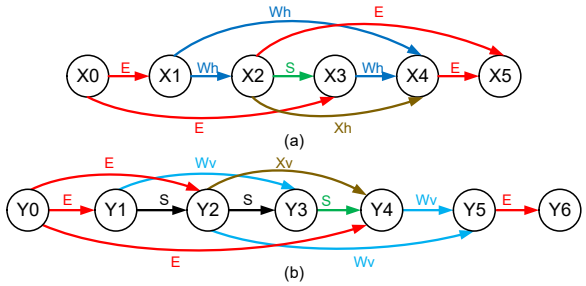

Fig. 4. (a) HCG and (b) VCG of the layout in Fig. 2. Here E, S, W, and X stand for min enclosure, spacing, width, and extension constrains, respectively. Label h and v indicate the direction.

Minimum Spacing: To minimize the effects of partial discharge in the layout, voltage-dependent spacing rule is applied. This minimum spacing depends on the voltage difference over the dielectric. To comply with the dielectric breakdown voltages, minimum trace spacing is calculated based on the voltage drop, so that current crowding, field focusing, and partial discharge can be mitigated. Same as current-controlled constraints, the higher value between design and reliability constraints dominates. For example in Fig. 3, each gap is subjected to a voltage-dependent constraint apart from the minimum spacing value imposed by the technology. All vertical gaps are found from vertical corner-stitched layout while horizontal gaps are from horizontal corner-stitched layout.

3) Constraint Graph Creation: From each corner-stitched plane, the constraint graph is created by iterating over entire design and reliability constraints. Each corner-stitched plane is traversed from left to right to create the HCG and bottom to top to create the VCG, which results in horizontal and vertical weighted directed acyclic graphs (DAGs). When there are both design and reliability constraints, the greater constraint value takes effect. The HCG, and VCG created from Fig. 2 are shown in Fig. 4. To keep the relative locations between neighbor vertices, the black spacing edges are added between them in the VCG. These are non-orthogonal constraints. For the given example, there is no vertically orthogonal relationship between Y1, Y2 and Y2, Y3. Therefore, those edges cannot be found directly from vertical corner-stitched layouts. These constraint graph weights are randomized to create new layout solutions. As the graphs are created from planar corner-stitched layout, the relative locations of all components are preserved in all the solutions.

C. Operating Modes and Evaluation Algorithms

The algorithms can serve four purposes to have better flexibility. Therefore, four operating modes are summarized in Table I. The existing layout engine has only fixed floorplan size solutions, whereas the proposed one is giving three more options that can generate more candidates for the designers to choose the optimum solution.

1) Mode-0: Mode-0 produces the minimum-sized layout that reflects the maximum possible power density for a certain layout. To evaluate the constraint graph, the longest path algorithm (shown in Algorithm 1) is used. In this algorithm, the source vertex is set at location 0. This means the reference coordinate of a layout is (0,0). For the rest of the vertices in the topological order, incremental locations from the source are calculated. Using the same algorithm, the longest path from the source to any vertex can be determined. While calculating the minimum location of a vertex, vertices traversed are on the longest path from source to that vertex. So, these traversed

TABLE I
SUMMARY OF OPERATING MODES

Mode	Purpose	Evaluation Methodology
0	Minimum sized layout	Minimum constraint values
1	Variable floorplan layouts	All Weights are randomized with minimum constraints. No maximum constraints
2	Fixed floorplan layouts	All Weights are randomized with minimum constraints. Some have maximum constraints
3	Fixed floorplan with fixed component locations	

vertices constitute the longest path. The maximum distance from the source to each vertex is set as the minimum location of that vertex. This algorithm has a time complexity O(E), where E is the number of edges in the graph. The evaluated graph gives the minimum location of each component in the layout.

Algorithm 1: Constraint Graph Evaluation(G)

```
A = Adjacency Matrix(G)
Location = f g
for i = 0 to length(A) do
    Predecessors= list of predecessors of vertex i
    if i == 0 then
        Location[i] = 0
    else
        Value=[ ]
        for j = 0 to length(Predecessors) do
            V=Location[j] + A[j][i]
            Value.append(V)
        Location[i] = max(Value)
Return Location
```

2) Mode-1: Mode-1 can produce variable-sized layouts. In this mode, all edge weights of the constraint graph are randomized based on minimum constraint values. Gaussian distribution is used to vary each edge weight within the limit of (min, c min), where c is a constant. The average is set to a value close to minimum constraint value, and the standard deviation is adjusted accordingly. Then the constraint graph is evaluated using the longest path algorithm and all vertex locations are determined. Each evaluated constraint graph determines the component location of the layout. The whole procedure is iterated over N times to generate N number of layouts.

3) Mode-2 and 3: Mode-2 generates layouts having fixed floorplan size. In the fixed area it randomizes the component location and generates different solutions. Whereas in mode-3, not only the floorplan size but also any component location can be fixed.

In general, if there is a connected weighted DAG and some vertices locations are required to fix at some values, Algorithm 4 is the proposed solution for such case. This algorithm can be used to solve the special case of the problem where the only initial source and sink vertices locations are to be fixed. This is the fixed floorplan problem and is solved in Mode-2 evaluation. However, this algorithm is generalized such that not only initial source and sink vertices but also single or multiple intermediate vertex or vertices can be at fixed locations. In Mode-3, the locations given for the components must be greater than or equal to the minimum locations.

Some terminologies and concepts used in the algorithm are explained here:

Fixed vertex: A vertex is fixed if it has the same minimum and maximum location. In other words, if a vertex location is determined or pre-defined that is called a fixed vertex. Each fixed vertex can be treated as a potential source or sink vertex for a path in the DAG.

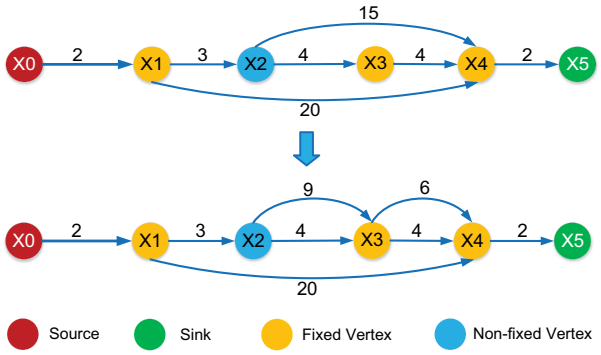

Fig. 5. An example showing edge splitting process.

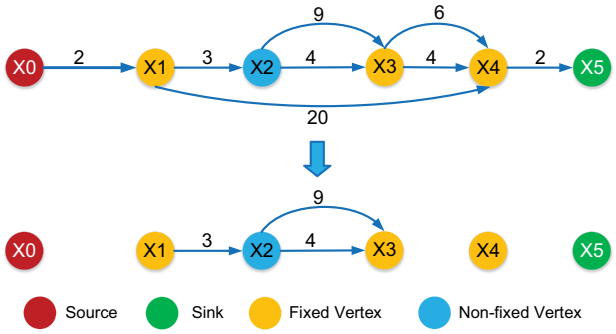

Fig. 6. An example showing edge removal process.

Non-fixed vertex: If a vertex location is not pre-defined or determined yet, it is a non-fixed vertex and needs to be evaluated.

Fixed edge: For an edge, whose weight must always be fixed, it is called a fixed edge. For example, the edges associated with device widths are fixed edges as the device width is always a constant.

Edge splitting: If there is a fixed vertex in between the initial source and sink vertices of a graph, edge splitting operation can be applied to reduce computational efforts. If any edge satisfies: a) It has both source or sink fixed, and b) it bypasses any fixed vertex that edge is a candidate for splitting. So, if such edge is found that edge is associated with at least two fixed vertices. Since a fixed vertex has a known fixed location the distance between those two fixed vertices is always a constant. Therefore, the total weight of the edge can split into two parts: one from the source to intermediate fixed vertex and another is from the intermediate fixed vertex to the sink vertex. Edge splitting concept is illustrated using Fig. 5, where the edge from X2 to X4 weighting 15 satisfies splitting conditions. That edge is divided into two parts: one between X2 and X3 with a weight of 9, another between X3 and X4 with a weight of 6 as both X3 and X4 are fixed vertices with a constant distance 6.

Edge removal: If any edge has both the source and the sink fixed that constraint value is of no use. Therefore, that edge can be removed from the graph to reduce the graph size. Also, any fixed edge in the graph is removable. Shown in the Fig. 6 example, X0, X1, X3, X4, X5 are fixed vertices. So, all edges associated with these vertices are removed.

Graph splitting: If there are fixed vertices in between the initial source and sink vertices, and no edge is bypassing that fixed vertex, then that graph can split into two connected sub-graphs. Since a fixed vertex is a potential source and sink that fixed vertex can be used as a sink to the left sub-graph and a source to the right sub-graph.

Randomization: Randomization is used to vary the edge weights

Algorithm 2: Find_Loc (G,Source,Sink)

```
Path=Longest_path(G,Source,Sink)
Values=list of minimum constraint values in the path
Total=sum(Values)
Max=Location[Sink]-Location[Source]
Range=Max-Total
Variable=Random_value(Range,Value)
for i=0 to length(Path) do
    if Path[i] not in Location then
        L=Location[Path[i-1]]+Variable[i-1]
        Location[Path[i]]=L
```

as well as locations of the components. If there is no fixed room for randomization, each constraint value in the graph is randomized using minimum constraint values. On the other hand, if the room is fixed for a certain path in the graph, to ensure design constraint satisfaction, sum of minimum constraint values of that path is subtracted to determine the actual range of randomization. Then the problem is similar to randomly distributing a constant value into several parts having the same expectation. This way, the component width or location is varied without violating any constraint.

The supporting functions for Algorithm 4 are discussed in Algorithm 2 and 3. Algorithm 2 is used to evaluate the single-source-single-sink graph that represents fixed floorplan sized layout. When a graph has only two vertices (source and sink) at fixed locations, this algorithm is used to evaluate the rest of the vertex locations in the path. First, it finds the longest path from the source to the sink. Then it finds the available quota for randomization. Variable is returned from the function Random_value, which generates the distributed random values. Finally, the for loop finds each non-fixed vertex incremental location in the path using the randomized values.

Algorithm 3 is used to evaluate the graph having multiple sources and sinks that means both floorplan size and any of the components are at fixed locations. This case happens when there is at least one intermediate fixed vertex in between initial source and sink. Due to the intermediate fixed vertex, while calculating the location of the other non-fixed vertices, extra constraints are added. This algorithm also starts with finding the longest path from the initial source to the initial sink vertex. Then in the longest path for each non-fixed vertex, all possible minimum and maximum locations are stored in Min_val and Max_val dictionaries respectively. After that, each non-fixed vertex location is found using randomization between the minimum and maximum location. As soon as one vertex location is calculated, it becomes another source and sink vertex for the rest non-fixed vertices. So, each newly fixed vertex is appended to the list of sources and sinks. This procedure is iterated until all vertices in the path have fixed locations. The performance of the algorithm increases with the increasing number of fixed vertices and the worst case has no intermediate fixed vertex. So the time complexity is O(E), where E is the number of edges in the graph.

In the Algorithm 4, edge splitting and edge removal are performed. Then if there is an intermediate fixed vertex, it partitions the graph from the initial source to that intermediate fixed vertex and from that fixed vertex to the initial sink. With multiple fixed vertices, more partitions are generated. A list of parts (Connected_parts) is created considering connectivity among different parts. For each connected partition, the procedure Eval_Loc is called as it reduces to a multiple-source-multiple-sink problem. Otherwise, a list of connected sub-graphs from Parts_list is made, which ensures there is no intermediate fixed vertex in each sub-graph. Then it reduces to a single-source-single-sink problem and is evaluated using Find_Loc procedure. After evaluation, some vertices are fixed and edge removal is performed again. The process is iterated until all vertices in the graph have fixed locations. To keep track of the locations, a global dictionary is maintained in which keys are vertices and values are locations correspond to the vertices. The overall time complexity should be O(E) as each edge is visited once in the whole procedure.

For mode-2, floorplan width and height are taken from the sink vertex location of the HCG and VCG respectively. By default the source vertex location is set at 0. Therefore, this is a single-source-

Algorithm 3: Eval_Loc($G, Sources, Sinks$)

```
start=Sources[0], end=Sinks[0]
Path=Longest_path(G,start,end)
Fixed=list of fixed vertices in the Path
Non-fixed=list of non-fixed vertices in the Path
while length(Non-fixed)>0 do
    Min_val=fg
    for each source in Sources do
        for each vertex in Non-fixed do
            Update Min_val where key=vertex, value=list of
              distance from source to that vertex
    Max_val=fg
    for each vertex in Non-fixed do
        for each target in Sinks do
            Update Max_val where key=vertex, value=list of
              distance from that vertex to target
    vertex=Non-fixed.pop(0)
    v1=max(Min_val[vertex]), v2=min(Max_val[vertex])
    Location[vertex]=random(v1,v2)
    Append the vertex to Sources and Sinks
```

Algorithm 4: Fixed_Loc(G)

```
while all vertices are not fixed do
    E= Dictionary of edges which are potential split candidates
     foreach edge in E do
        split_edge(edge)
    Fixed-vertices= list of vertices having fixed locations
    foreach edge in G do
        if edge is between a pair of fixed vertices then
            G.remove(edge)
    Parts_list=list of parts of G having source and sink pair in
     Fixed-vertices
    Connected_parts= list of parts from the Parts_list which are
     connected to each other
    if Connected_parts found then
        foreach part in Connected_parts do
            Sources= list of all potential sources in part
            Sinks= list of potential sinks in part
            Eval_Loc(G,Sources,Sinks)
    else
        Sub-graphs=list of connected sub-graphs in Parts_list
        foreach sub-graph in Sub-graphs do
            Source=source of the sub-graph
            Sink=sink of the sub-graph
            Find_Loc(G,source,sink)
    Fixed-vertices= list of vertices having fixed locations
    foreach edge in G do
        if edge is between a pair of fixed vertices then
            G.remove(edge)
```

TABLE II
RUNTIME ANALYSIS OF PROPOSED ALGORITHMS, WITH 3015 LAYOUTS GENERATED FOR OPTIMIZATION

Case #	Tile #	Vertex #	Edge #	Runtime (s)		
				Mode-0	Mode-1	Mode-2
1	8	18	36	0.0159	2.4432	3.5713
2	15	28	68	0.0165	4.10597	7.1645
3	34	43	165	0.0294	7.7506	12.2803
4	72	63	311	0.0551	11.9551	34.1308

single sink constraint graph evaluation problem. The algorithm shown in Algorithm 4 is used here to find all vertex locations. As there is no intermediate vertex at a fixed location, no Connected_parts is found, and Location-finding algorithm(shown in Algorithm 2) is used for evaluation with an O(E) time complexity.

Mode-3 allows fixing components (such as leads) at pre-defined locations which are very useful for packaging purpose. As leads are used for external connections, those positions should not be varied from layout to layout. In case of mode-3 evaluation, width, height, and some other component locations are taken as input in the form of a dictionary. Due to intermediate vertices at fixed locations, at least one multiple-source-multiple-sink case appears thus Algorithm 3 is used. After sufficient edge splitting and removal, the graph is split into single-source-single-sink sub-graphs. Therefore, the Mode-3 operation is also performed in O(E) time complexity.

All of these algorithms are used to evaluate constraint graphs. Each evaluated constraint graph gives all vertex locations that determine component positions in the layout. The current layout engine of PowerSynth randomly changes widths of the components and generates new layouts without any knowledge of design constraints, and has to discard most generated solutions due to DRC-failure. Also, the spacings between components are not considered as variables. The current layout engine also does not allow different minimum width assignment for the same component, which makes it incapable of differentiating signal and power traces. Therefore, the reliability constraints can not be considered at all.

IV. EXPERIMENTAL RESULTS

To evaluate the runtime of the proposed algorithms, several layouts with a different number of components and different geometrical complexity are chosen. The runtime summary for different operating modes of the algorithms is shown in Table II.

For mode-1 and 2, 3015 layouts are generated for comparison. Mode-3 operation comparison is not a good choice as it is not possible to fix the same vertex locations for each layout. However, Mode-2 is actually a special case of Mode-3, except that only single source and single sink are fixed. From Table II, it is clear that the proposed methodology has a linear time complexity that ensures scalability for heterogeneous power module layout optimization.

Layout generation efficiency comparison between PowerSynth existing and proposed layout engine is shown in Table III. Three layouts with increasing geometrical complexity are chosen for comparison. As current layout engine can only generate fixed-sized layouts, only Mode-2 solutions of the proposed engine are comparable. From the table it is clear that current layout engine is way less effective than the proposed one. For some cases, current layout engine just fails to generate a single valid layout, whereas the proposed layout engine generates 100% valid layout solutions for all cases with a small runtime overhead.

TABLE III
ALGORITHM EFFICIENCY COMPARISON BETWEEN CURRENT AND PROPOSED LAYOUT ENGINES

Case #	Valid # out of 3015		Total Time(s)		Floorplan Size
	Current	Proposed	Current	Proposed	
1	76	3015	0.74	3.57	30×30
2	1883	3015	1.25	7.16	40×50
3	0	3015	N/A	34.13	98×78

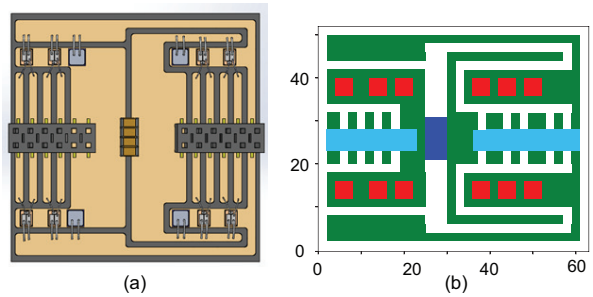

Fig. 7. (a) A half-bridge power module, and (b) its minimum-sized layout.

A. Design-Constrained Layout Solutions

To show layout generation results with only design constraints considered a complex half-bridge power module is chosen (shown in Fig. 7(a)). In all layouts, green components are traces, red rectangles are devices, the dark blue one is the DC link capacitor and the light blue one is the connector for the gate driver. In these layouts, colored blocks represent the room allocated to the corresponding component rather than the actual component dimensions.

Mode-0 result: The minimum sized layout generated using standard design constraint values (shown in Table IV) is shown in Fig. 7(b). Since all components are in the same plane, due to the correlation between vertices in the constraint graph all components are not in the smallest size.

Mode-1 result: From the same initial layout, variable-sized layouts are generated. Two sample results are shown in Fig. 8(a), (b), where floorplan sizes are (165 92) and (145 113), respectively.

Mode-2 result: In case of Mode-2, the floorplan size is (145 140) in the layouts shown in Fig. 9.

Mode-3 result: In case of Mode-3, the floorplan size is (100 100). The capacitor (blue component) is fixed at location (50,40) in both layouts shown in Fig. 10.

B. Design-and-Reliability-Constrained Layout Solutions

To demonstrate the capability of handling reliability constraints such as voltage-dependent spacing and current-dependent width, some constraint values are assumed as shown in the Table V. Using these values, Fig. 12(b), (c) are generated. In Fig. 12 (a) only design constraints are applied, which results in a constant minimum spacing between traces and a minimum floorplan size among three layouts. The layout shown in Fig. 3 has all components assigned with voltage and current rating. This initial layout is used as the input for generating layout solutions by the proposed layout engine. Applying constraint values from Table IV and Table V minimum sized layout is generated (shown in Fig. 12(b)). The voltage difference between traces is rounded up to find the corresponding spacing constraint. For example, the voltage difference over G3 is 25 V, thus it finds 100 V as the closest maximum, which imposes a 1 mm gap. Since the minimum trace-to-trace spacing is 2 mm, as defined by the design constraint, it dominates in this case. Similarly, as G6, G7, G8 are

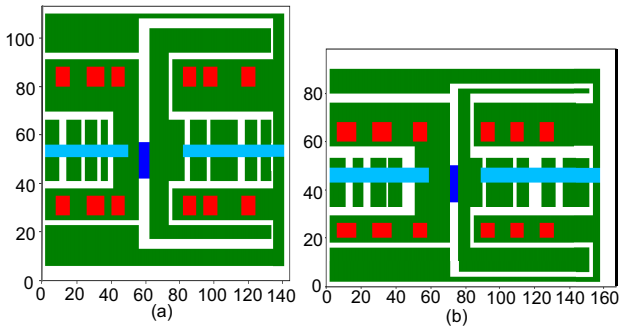

Fig. 8. Mode-1 layouts of the module in Fig. 7.

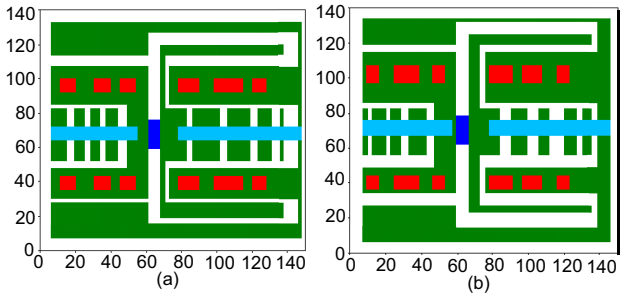

Fig. 9. Mode-2 layouts of the module in Fig. 7.

correlated, 3mm is the maximum spacing imposed by reliability constraint values. According to the design rule, the minimum trace width is 2 mm. For signal traces, carrying a 2 mA current, it is sufficiently reliable. However, power traces have a higher current rating that imposes a higher reliability constraint value. Therefore, power traces have a minimum width of 3 mm. Similarly, Mode-1, 2, and 3 operations can be performed to generate variable, or fixed floorplan layouts. For the same layout, power trace voltages are assumed in kV range with a current of 400 A. Generated solutions are shown in Fig. 12(c).

Applying the proposed algorithms, three different solution sets from Fig. 3(a) are generated. For the first layout set, a fixed trace-to-trace distance of 0.2 mm is applied. This layout set represents the minimum manufacturable requirements without considering reliability constraints. In the second data set, both design as well as reliability constraints are applied. Here, trace-to-trace distance is varied with the voltage difference between traces. The current dependent constraints are also applied in this case. To have reliable layouts rated at 200 A peak current, the min trace width is set to be 2 mm. The third data set is generated using a constant 4 mm trace-to-trace distance to minimize PD and thermal issues. In each case, 1000 candidate solutions with varying floorplan sizes from 2475 mm^2 to 9000 mm^2 are generated for optimization. Then, PowerSynth electrical and thermal models are applied to evaluate the maximum temperature as well as the loop inductance for each layout. Three different Pareto-frontiers for the corresponding data sets are shown in Fig. 11(a). As seen in the figure, the results from three data sets illustrate expected relationships between inductance, temperature and layout area. Inductance increases due to larger conduction loop while temperature tends to reduce with the increased layout area. Due to the smallest fixed gap of 0.2 mm, the first layout set has the largest trace variation while the third case has the smallest. By applying both reliability and design constraints, the second data set provides not only DRC-clean layout solutions with good performance but also ensures higher reliability in terms of thermal and partial discharge.

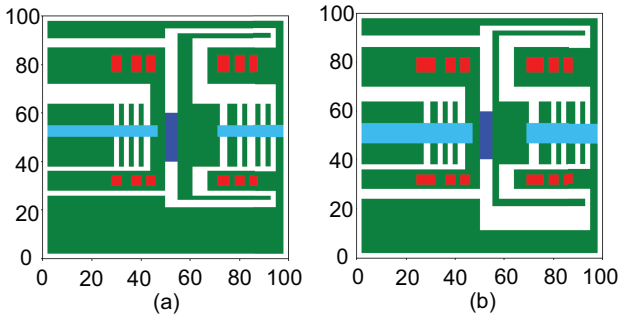

Fig. 10. Mode-3 layouts of the module in Fig. 7.

TABLE IV
DESIGN CONSTRAINT VALUES IN MM.

Constraint Type	Value
Min width of trace	2
Min width of die	4
Min dimension of capacitor	5×10
Min dimension of connector	10×5
Ledge width	2
Trace-to-trace spacing	2
Die-to-trace enclosure	1
Die-to-die spacing	1

This provides designers an opportunity to choose tradeoff between performance and reliability. Third one can ensure better reliability but has high parasitics. Three selected layouts having minimum inductance from three Pareto-fronts are shown in Fig. 11(b), (c), and (d). The first layout has the minimum gap, with the least reliability, whereas the third one is the most reliable but with highest parasitics. Therefore, the second layout is the most optimum in terms of both electrical, thermal and reliability.

V. CONCLUSION AND FUTURE WORK

The corner stitching data structure and constraint graph methodology are highly-suitable for heterogeneous power module layout synthesis and optimization. Constraint graph methodology has better efficiency and scalability over the traditional matrix-based representation methodology used in existing layout design tools. The proposed physical design approach generates DRC-clean layouts and eliminates the DRC validation step in the optimization process to improve efficiency. The algorithms are designed as generic and scalable as possible to handle layouts with complex geometry and heterogeneous components. Also, the reliability constraint-awareness adds a significant feature to the layout generation methodology, which is extremely useful for high-voltage power modules. Though the proposed planar methodology is not giving the actual fabrication-feasible layouts due to coordinate correlation, hierarchical layout representation will solve this problem. In the hierarchical approach, the correlations between components are minimized, which introduces a larger solution space for optimization. The next target is to apply hierarchical constraint graph methodology to generate more feasible layout solutions, exploit different multi-objective optimization algorithms, and also explore 3-D power module layouts optimization.

VI. ACKNOWLEDGMENT

The authors would like to thank Tom Vrotsos, Tristan Evans, Shilpi Mukherjee, and John Alumbaugh for their help throughout the research. We are also grateful to Dr. Fang Luo's research group for their suggestions and assistances along the way.