Chapter 5

Methods to Localize Shorts Between Power and Ground Circuits

Mark Braverman[1], Shengyuan Chen[2], Marcio Gameiro[3], Nadine Gärtner[4], Yasong Jin[5], Edward Keyes[6], Fadil Santosa[7], Haris Widjaya[8]

Report prepared by Fadil Santosa (santosa@ima.umn.edu)

5.1 Introduction

In the competitive world of microprocessor design and manufacturing, rapid advancements can be facilitated by learning from the components made by one's closest rivals. To make this possible, Orisar Inc. (formerly Semiconductor Insights) provides reverse engineering services to integrated circuit (IC) manufacturers. The process produces a circuit diagram from a chip and allows the manufacturer to learn about a competitor's product. These services are also used to determine if any intellectual property infringements have been committed by their competitor.

Reverse engineering of integrated circuit is made difficult by the shrinking form factor and increasing transistor density. To perform this complex task Orisar Inc. employs sophisticated techniques to capture the design of an IC. Electron microscope photography captures a detailed image of an IC layer. Because a typical IC contains more than one layer, each layer is photographed and physically removed from the IC to expose the next layer. A noise removal algorithm is then applied to the pictures, which are then passed to pattern recognition software in order to transfer the layer design into a polygonal representation of the circuit. At the last step a knowledgeable human expert looks at the polygonal representation and inputs the design into a standard electronic schematic with a CAD package.

[1]University of Toronto
[2]University of British Columbia
[3]Georgia Institute of Technology
[4]Clemson University
[5]University of Kansas
[6]Orisar Inc.
[7]University of Minnesota
[8]Simon Fraser University

At the first few stages of the process, errors can be introduced into the polygonal representation of the design. Errors can exist in the form of noise or simply be the result of a physical obstruction on the layer being photographed. These errors can introduce invalid components or connections within the device, and the most catastrophic of these errors results when the power and ground networks of an IC are tied directly together. Since the typical first step in determining component functionality is the determination of power and ground, these short circuits or *shorts* must be identified and corrected before further analysis can occur. It should be emphasized that it is a very rapid process to determine the existence of a short however finding the *location* of a short is another matter.

In the current work flow, workers at Orisar Inc. visually inspect the polygonal representation of the circuit in order to find errors. To completely remove all the shorts one person needs to spend one day inspecting the polygonal data by running signal tracing. To make the search faster, the search area is narrowed down by one quarter by running signal tracing on a quarter of the IC at a time, the shorts can then be quickly narrowed down to small sub-regions.

However this process, performed manually, is very time consuming. We propose a method which can be easily automated thereby saving valuable worker time and accelerating the process of reverse engineering.

5.2 Problem Description

A modern IC design contains in the order of millions logic gates packed into a very small area. The gigantic number of devices, together with minimal space, create a problem for designers to connect their logic gates efficiently. This is the main reason for the multi layered approach to IC manufacturing. A typical IC has one layer dedicated to the placement of logic gates and multiple layers of wiring. This is necessary because all the gates on the gate layer are packed as densely as possible, leaving no room for interconnecting wires and connection to the power supply.

In a modern IC there are many more wiring layers than logic gate layers, and typically adding a layer to an IC translates directly into an increase in the cost of manufacturing. Thus any manufacturer tries to pack the wires as densely as possible within the layer without violating the design rules. Because an IC consists predominantly of wiring, it is crucial to get the wiring design transfered properly. There are three types of wiring we can find within an IC.

- Power lines which we will denote as V_{dd}. These lines carry the positive charge into the components of the IC.
- Ground lines which we will denote as V_{ss}. These lines carry the negative charge.
- Signal line which interconnect the various components.
- Vias, this is the inter-layer wiring that connects wires to wires, and wires to logic gates.

The two main wires that we will focus on are the power lines and the ground lines. On a normal IC, the power lines and the ground lines form two disconnected components. Unfortunately nature is against us in this process. The errors mentioned in the introduction can add wires that are not part of the real IC design, and create inadvertent connections, or shorts, between the power lines and the ground lines.

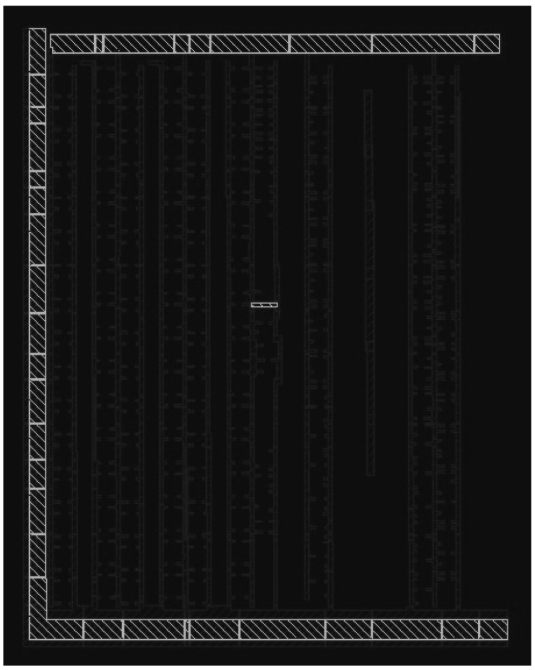


Figure 5.1: A polygon representation of wires in an IC.

The wires in the IC are represented as a set of overlapping polygons. Each polygon contains a list of vertices and a signal number. The number of vertices in each polygon can be very large although many of the polygons are quadrilaterals. The polygons are not guaranteed to be convex, and they can vary greatly in size. The wires are typically laid out in a regular fashion, that is wires run either parallel or or orthogonal to other wires.

In the next section we will explain the approaches that we have considered to solve this problem, and the choices we made concerning the final algorithm.

5.3 Approaches to the Problem

5.3.1 A Network Flow Method

If current can flow from a V_{ss} pin to a V_{dd} pin, it must go through one of the shorts. Normally a short is much narrower than a legitimate wire. Given the fact that the wider a wire is, the more capacity it bears, shorts become the bottleneck of the network. This idea is already used in manual inspections where power is applied between two pins and a thermal image of the circuit is made to identify hot regions.

We would like to simulate this process in a computer. The first step would be to convert the circuit into a network. Our thought is to translate a given circuit into a capacitated network where each edge in the network is a piece of wire. This leave the problem of estimating the capacity of a wire. Several ideas were proposed:

1. Starting from the scanned image, we may regard every black pixel as a node of the network, and assign each edge capacity one. This approach leads to a massive network, which may be too

expensive to compute.

2. Starting from the polygons representation of a wire, determine the direction of flow and estimate the width of the wire. This process can be very complex with many special cases as a polygon could be as simple as a quadrilateral and as complicated as a nonconvex n-gon.

3. Another idea is to extract the network from the scanned image by scanning through all pixels vertically and locating junctions and edges. Even though the idea is very attractive, the group felt that it was unlikely that an algorithm to do this can be completed in the time given.

We remark that upon transforming the circuit into a finite capacity network, we can use network flow techniques to locate the shorts. The max-flow of such a network is limited by the shorts, and thus the min-cut will be on the edges of these shorts. We assume that there are only a few shorts and the sum of the capacities of all these shorts are less than that of a narrow wire.

5.3.2 Signal Processing to Locate H-Junctions

The observation that shorts tend to make **H**-junctions between two legitimate wires lead us to consider pattern recognition algorithms. It was felt that such an algorithm, though it can identify false positives, could be valuable in localizing the search. After some consideration, it was felt that such a method is not sufficiently general.

5.3.3 Exploiting the Incidence Matrix

Since the incidence matrix encodes the connectivity of the circuit, it seems possible that we can devise a way by which the matrix is manipulated so that it separates out into power and ground parts, with the shorts connecting them. To be able to do this, we must develop an objective functional such that when it is minimized, the matrix segregates as desired. This appealing idea deserves further study.

5.3.4 A Monte Carlo Approach

We convert a circuit into a graph by letting the polygons representing the circuit be vertices. If two polygons are connected, then the vertices representing them are connected by an edge. With this description, we can define the length of a path between a pair of polygons. The nice feature of shorts is that it must appear on a path starting from a polygon on the V_{dd} (power) side to a polygon on the V_{ss} (ground) side or vice versa. To utilize this observation, we randomly pick a pair of starting and end points, and compute the shortest path between them using the breadth first algorithm. We repeat this procedure a large number of times, and we count the number of times each polygon appeared in all of these paths.

We claim that the polygons from a short will have a higher number of visits that other polygons, thus allowing us to isolate potential shorts. This approach, as well as an implementation, are described in the next section.

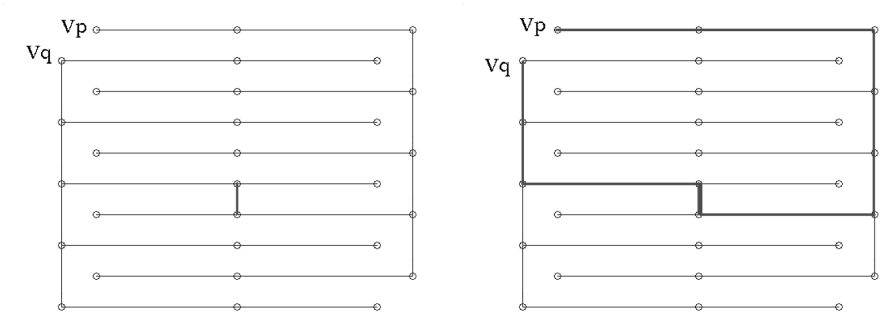


Figure 5.2: The shortest path between v_p and v_q in a simple circuit.

5.4 A Monte Carlo Algorithm

5.4.1 Description of the Algorithm

To make it possible to apply graph theory algorithms to the problem, we represent the circuit as an undirected graph G. The vertices of the graph correspond to the polygons representing the circuit and the edges represent the connection between adjacent polygons. For two vertices v_p and v_q, corresponding to polygons P and Q, there is an edge connecting v_p and v_q if and only if the polygons P and Q touch each other. A path between two vertices v_p and v_q in G is a path between the polygons P and Q in the circuit.

If a circuit has a short, then any path going from a polygon in V_{dd} (power) wire to one in V_{ss} (ground) wire must pass through a short. Since there are only few shorts in the circuit, we would expect many paths to go through each of the shorts. Our algorithm randomly chooses pairs of vertices in the circuit, connects them through a shortest path, and keeps track of the number of visits to each vertex of the circuit. While any path between these vertices would work, we choose the shortest path (see Figure 1.2). We expect that the shorts would be the most visited areas in the circuit.

We keep a counter for the number of visits to each of the vertices. At each iteration of the algorithm we randomly choose two vertices v_p and v_q. We then find a randomized shortest path from v_p to v_q using a randomized version of the **breadth first search** (BFS) algorithm. Then we increase the counter of each of the visited vertices by 1.

We make a large number of iterations, and output a colour map highlighting the most visited areas. Then we manually search for the shorts in the highlighted areas.

The Algorithm
1) N = number of trials;
2) For each p, $counter(v_p) = 0$;
3) **For** i = 1 **to** N
    3.1) randomly choose vertices v_p and v_q;
    3.2) find one of the shortest paths from v_p to v_q,
        $v_p := v_0, v_1, v_2, \ldots, v_K =: v_q$ using a randomized BFS;
    3.3) **For** j=0 **to** K
        3.3.1) $counter(v_j) = counter(v_j) + 1$;
    Endfor
Endfor
4) Output a colour map with the values of $counter(v)$ for each v, the polygons with high counter values are the primary suspects to be the shorts.

Randomized BFS is an algorithm which randomly chooses one of the shortest paths from V_P to V_Q. Details will be explained in the end of the section.

5.4.2 Analysis of the Algorithm

The main idea is that every time we choose v_p from V_{dd} and v_q from V_{ss} or v_p from V_{ss} and v_q from V_{dd}, the path from v_p to v_q must pass through one of the shorts of the circuit. Next we estimate the probability that the chosen v_p and v_q are in different wiring nets? Suppose that the portion of V_{ss} polygons in the net is s, and the portion of V_{dd} polygons is d, $s + d = 1$. Then the probability that we choose v_p from V_{dd} and v_q from V_{ss} is ds and the probability that we choose v_p from V_{ss} and v_q from V_{dd} is sd, so the probability that v_P and v_q are chosen from different wiring nets is $2ds$. This probability is almost 50% if s and d are relatively close to each other. We expect this to be the case most of the time. The following table summarizes the estimated outcomes

The portion s of V_{ss} in the net	The portion d of V_{dd} in the net	The probability that v_p and v_q are chosen from different nets.
10%	90%	18%
20%	80%	32%
30%	70%	42%
40%	60%	48%
50%	50%	50%
60%	40%	48%
70%	30%	42%
80%	20%	32%
90%	10%	18%

If there are k shorts in the circuit, then each of the short points gets visited

$$\frac{2 \cdot d \cdot s \cdot N}{k}$$

times on average, where N is the number of trials. We have

$$\frac{2 \cdot d \cdot s \cdot N}{k} \approx \frac{N}{2 \cdot k}$$

when s and d are close to each other. This number is much larger than the number of visits to an average polygon in the circuit. So the shorts should get highlighted by the algorithm.

A possible improvement to the algorithm would be discarding particularly short paths. A typical path connecting a point in V_{ss} with a point in V_{dd} going through one of the shorts and is generally longer than a path connecting two points on the same side of the net. Hence discarding short paths would discard more paths which connect vertices on the same side, thus yielding a better performance of the algorithm.

Note that the algorithm is extremely parallelizible. All the trials can run completely independently. So the algorithm can be executed simultaneously by arbitrarily many machines in parallel, and the visits statistics can be collected at the end of the run.

5.4.3 The Randomized BFS Algorithm

We use the classical BFS (breadth first search) algorithm for finding a shortest path in an undirected, unweighted graph. It is a classical algorithm described in many introductory algorithms books. A good exposition of the algorithm can be found in Cormen [1]. We modify this algorithm in order to choose a random shortest path.

After scanning the graph from the source s and labelling all the vertices according to their distance from s, we backtrack from the target t in order to find the shortest path from t to s. Whenever we have more than one way to backtrack, a random choice is made amongst the possibilities. This procedure diversifies the possible routes in congested areas and thus avoids false positives.

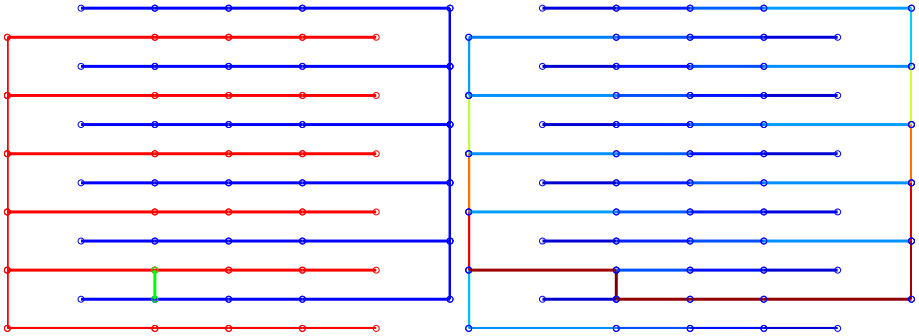
5.5 Performance of the Randomized Algorithm

On a typical circuit we will have approximately 10^5 nodes in the graph, with the number of shorts in the order 10^1. The testing is done in increasing node density to approximate real cases, this is important because our method works under the assumption that the number of shorts are a few orders of magnitude smaller, thus the probability of it being hit by our randomized tracing algorithm is higher than a real node. The following are the results.

In the first experiment we have one short in between the two interdigitated power and ground wires as shown in Figure 5.3. The power wires are coloured red and the ground coloured blue. The short is shown as the green line in Figure 5.3a. The result of our algorithm, with 5000 trials, is shown in Figure 5.3b, it is evident that our algorithm has succeeded in localizing the area of the probable short shown by the red wires in the picture.

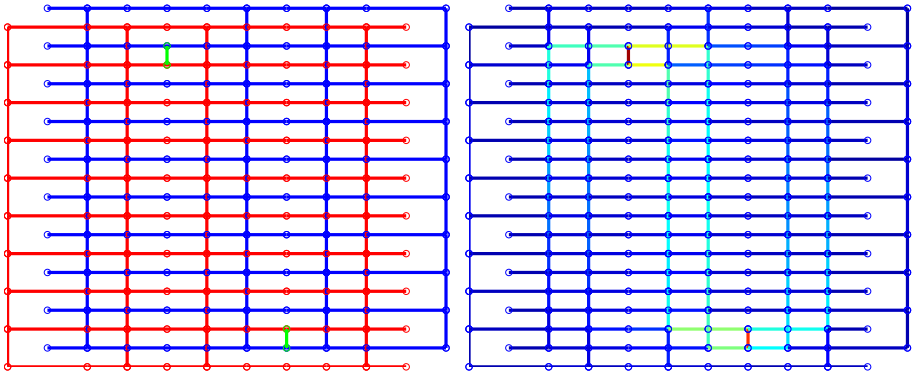
In the next experiment we tested our algorithm on a circuit with a higher node density. The circuit contains two shorts. It is clear that our algorithm is able to find a local area where the short occurs (see Figures 5.4). Here, $N = 5000$ trials as well.

In the next set of figures we ran the algorithm on a circuit that has 3 shorts. After a first run involving 5000 trials, only one short was found (Figure 5.5b). We repeated the run after removing the



(a) Initial figure with one short. (b) Final figure.

Figure 5.3: Localization of 1 short.



(a) Initial figure with two shorts. (b) Final figure.

Figure 5.4: Localization of 2 shorts in 1 run.

short found in the first run and identified a second short in Figure 5.5d. A final run, after removing the second short was able to determine the location of the third and final short (Figure 5.5f).

We have numerical evidence that our algorithm coupled with simple heuristics is capable of removing all the shorts that exist within a given circuit model. We have also tested our algorithm on real data that the company uses.

Figure 5.6a shows a case from a real circuit that has been annotated and identified by a knowledgeable human expert. The white lines in Figure 5.6b are areas our algorithm has identified as possibly containing shorts. This illustrates that our algorithm correctly identified the local areas where shorts occur.

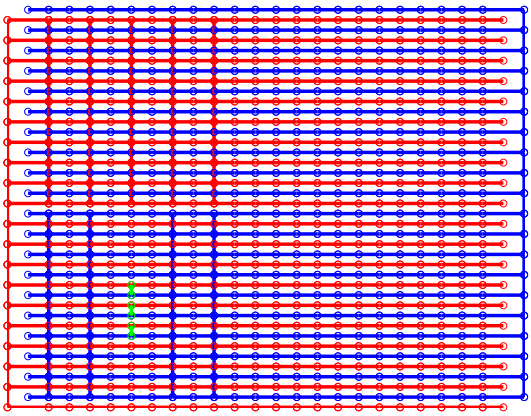
5.6 Conclusions

The team has implemented and tested a Monte Carlo approach capable of localizing power to ground short circuits. Any valid wiring network consists of two unconnected power and ground networks. The key observation is that given any two randomly chosen points in the wiring network, there exists a relatively high probability that the shortest path joining these two points will contain a short circuit if it exists. Our algorithm exploits this relatively high probability of traversing a short and has been shown to be capable of detecting multiple shorts of both simulated and actual IC power wiring networks.

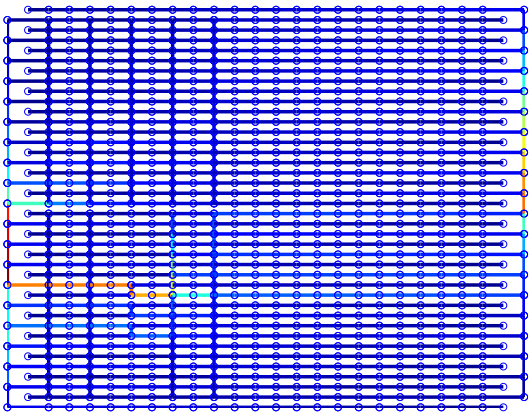
There are several ways in which the proposed method can be improved. These include:

- Make better use of path information.
- Incorporate knowledge embodied in the incidence matrix of the graphs.
- Exploit properties of the circuits, such as line widths.

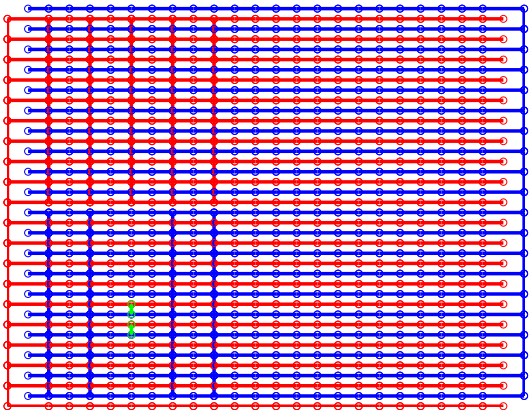
We believe that other ideas explored in this report deserve further investigation.



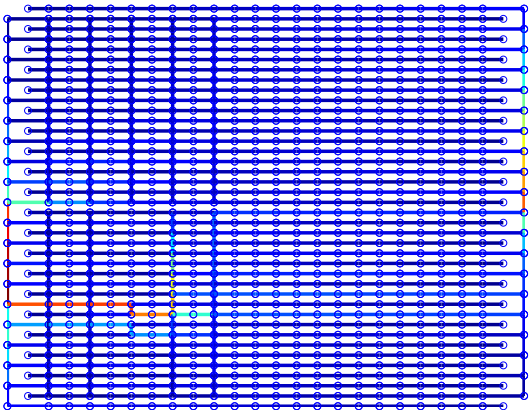
(a) Initial figure with 3 shorts before 1st run.



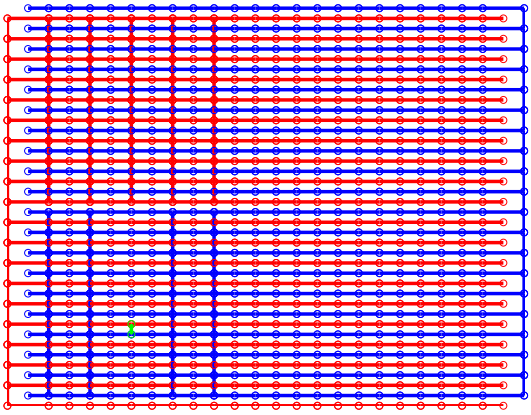
(b) Final figure with 3 shorts after 1st run.



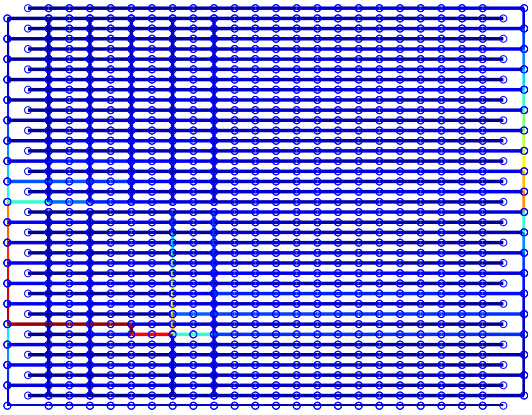
(c) Initial figure with 3 shorts before 2nd run.



(d) Final figure with 3 shorts after 2nd run.

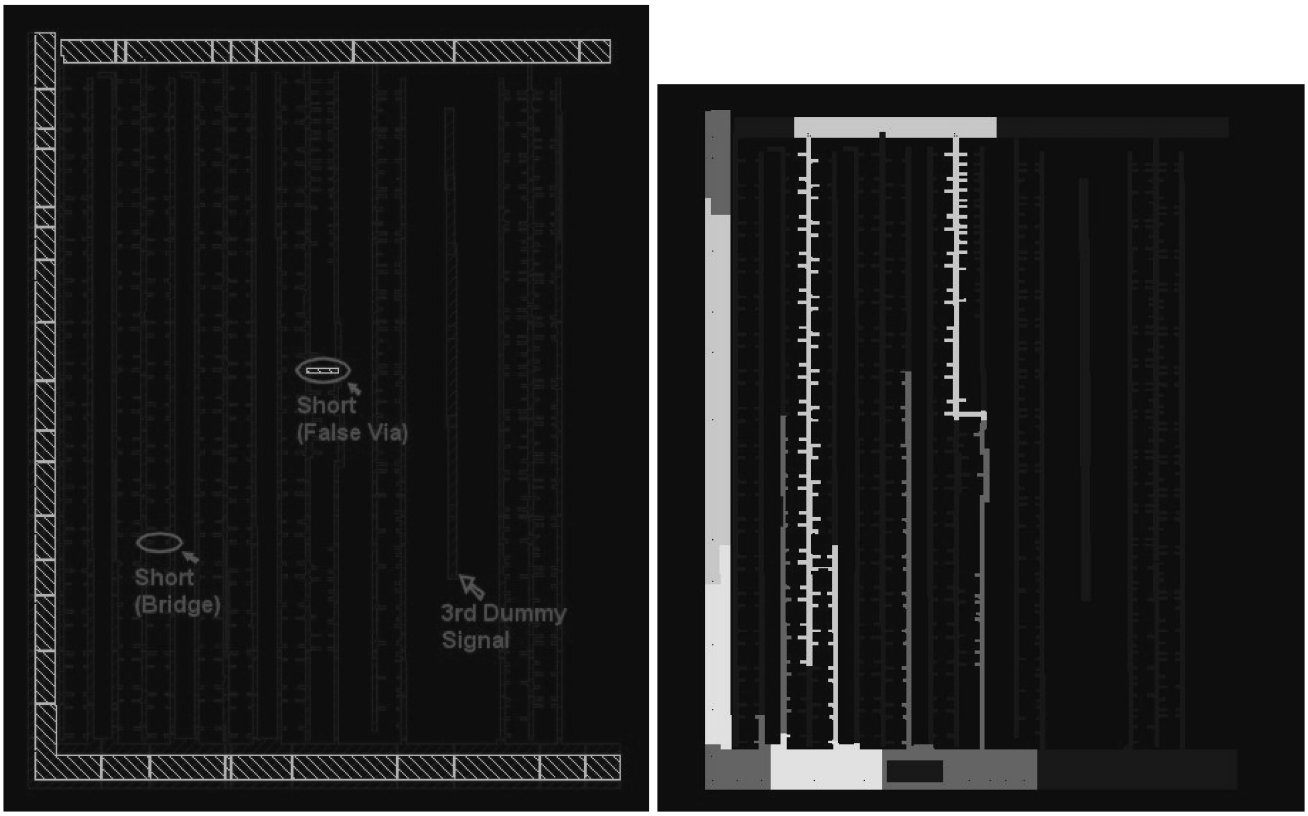


(e) Initial figure with 3 shorts before 3rd run.



(f) Final figure with 3 shorts after 3rd run.

Figure 5.5: Localization of 3 shorts in 3 steps.



(a) The areas where shorts occur.

(b) The highlighted by our algorithm after 5 iterations.

Figure 5.6: The result of our algorithm with real data.

Bibliography

[1] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, *Introduction to algorithms*, McGraw-Hill, New York, (1997).