MTBoM: Metal Trace to Bill of Materials Generation for PCB Reverse Engineering

Lubaba Nahar*, Jeyavijayan Rajendran†, Yiorgos Makris* and Carl Sechen*
†Dept. of Elect. & Comp. Engineering, Texas A&M University, College Station, TX, USA
*Dept. of Elect. & Comp. Engineering, University of Texas at Dallas, Richardson, TX, USA
Email: lubabam.nahar@utdallas.edu, jv.rajendran@tamu.edu, yiorgos.makris@utdallas.edu, and carl.sechen@utdallas.edu

2022 IEEE 15th Dallas Circuit And System Conference (DCAS) | 978-1-6654-9885-2/22/$31.00 ©2022 IEEE | DOI: 10.1109/DCAS53974.2022.9845644

***Abstract*—In this work, we developed an automatic tool for printed circuit board reverse engineering (PCB-RE) in which it is assumed that the PCBs are devoid of any components or silkscreens, with only the wiring traces being accessible on the various layers. This models the scenario where the PCBs are damaged and/or discarded. We demonstrate that our PCB-RE tool extracts the correct Bill of Materials (BoM) for ten different PCBs by analyzing primarily the metal layers. The proposed PCB-RE tool MTBoM detected every integrated circuit (IC) on the PCBs with no false positives and no false negatives. This scheme also identifies passive components, such as resistors, capacitors, and inductors.**

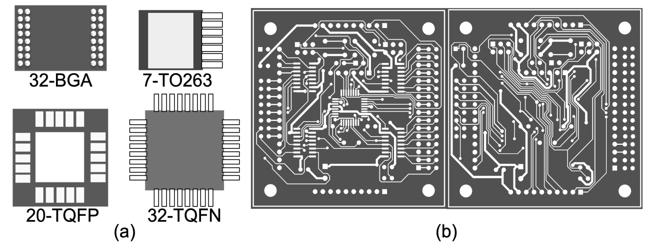


Fig. 1. Metals on PCBs: (a) footprints of PCB components (ICs) in 4 different packages; (b) metal traces of a random PCB [lighter area refer to the metal].

I. Introduction

In recent years, nondestructive PCB Reverse Engineering (RE) by X-ray tomography [1] indeed advances the process of PCB-RE automation. PCBs are assembled with ICs, power sources, connectors/headers, and other passive components, connecting metal traces in between them. Whether each of these components comes in a package or not, the metal must be placed according to its footprints on the PCB to ensure proper electrical and mechanical support; additionally, metal routed traces and vias connect various components providing signal connectivity with minimum resource allocation. MTBoM utilizes these two pragmatic metal elements: i) component placements with unique footprints and ii) trace-routes that confirm the signal connectivity, as key clues for BoM generation. In Fig. 1, four random footprints of different IC packages are shown on the left, illustrating their uniqueness. On the right, metal traces for 2 layers of a reverse-engineered PCB vivifying PCB metal's impression.

In this work, we focused on PCB RE automation, in which the Bill of Materials (BoM) is automatically generated, given solely the various metal traces. Our tool is called MTBoM, for metal trace to BoM generation. For a legacy PCB, if a component to be replaced is missing, MTBoM is the solution. This paper provides the following contributions and findings:

- We developed an automated PCB-RE tool MTBoM and proposed its heuristic considering metal data is available either by a destructive [2] or nondestructive PCB-RE method.
- For the first time, Bill Of Materials derived solely from metal traces has been presented; we provide the formulation of a one-shot IC detection method.
- We demonstrated the effectiveness of our tool on ten different PCBs, and in all cases, we can extract the correct BoM.

II. PCB Security vs PCB RE

Supply chain risk is a known concern in software acquisition and hardware procurement; PCBs are no exception. Precedent PCB security work showed how PCBs are becoming more vulnerable to malicious insertion during design or fabrication in untrusted design or fabrication facilities [3] and how PCBs can be exploited during in-field tampering attacks [4], [5].

Though RE may lead to cloning, counterfeit, or Hardware Trojan [HT] insertion, which are significant concerns for the government, it leverages the assurance of system integrity. Furthermore, RE is the only option to repair a legacy system when its design manual no longer exists.

Several PCB-RE studies have been published for the last two decades. In [6] Longbotham first demonstrated X-ray imaging as a powerful reverse-engineering tool. Grand [2] described different methods that fully reverse-engineer a device by extracting images of all PCB layers. In recent times, Asadi [1] described PCB reconstruction using advanced image processing with X-ray tomography and generated an unfolded copy of all layers of a PCB.

The generic PCB RE steps are illustrated in Fig. 2. Aiming to recover the implementation and its functionality from an assembled PCB, reverse engineering can generate the schematic by re-producing each metal layer's layout. We can perform it

Authorized licensed use limited to: Univ of Texas at Dallas. Downloaded on October 09,2022 at 04:05:02 UTC from IEEE Xplore. Restrictions apply.

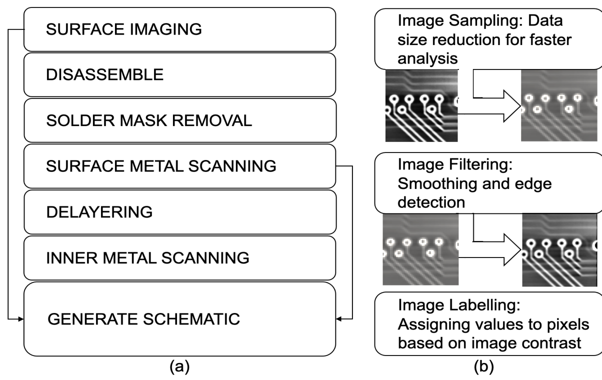


Fig. 2. PCB RE: (a) basic PCB-RE steps and (b) advanced image processing steps for non-destructive PCB RE [1] [lighter area refers as metal].

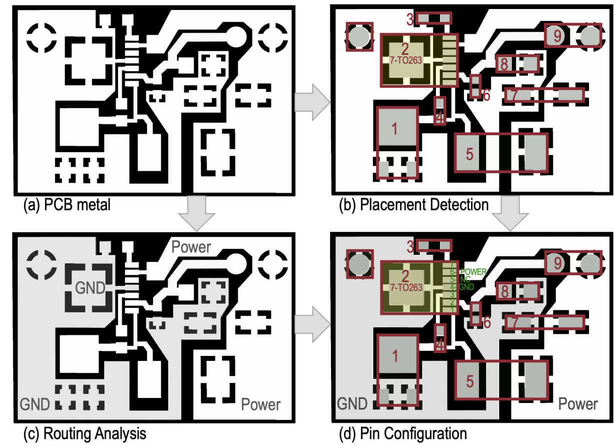


Fig. 3. Motivational example: (a) PCB metals in white on a black PCB; (b) identified footprints are maroon-boxed and numbered during placement detection, #2 was identified as an IC in the 7-TO263 package; (c) GND is grayed, power is tagged as a result of routing analysis; (d) partial pin configuration for IC component #2 is derived.

by orchestrating the steps in Fig. 2(a) [2]; Fig. 2(b) illustrates the additional image processing steps required instead of physical destructive delayering for the non-destructive methodology by X-ray tomography [1].

However, as we show in this work, assuming metal data can be produced using any of the above techniques, we automate correct BoM generation even though all component information is absent.

III. MTBoM

A. Objective

The goal is to restore the BoM, i.e., all components on a PCB. We explore all relevant websites for available off-the-shelf product specifications to gather the required information. Anticipating the exclusion of any silkscreen or any component part-number information, we examine if it is possible to reproduce the detailed component list only from metal data.

Our first challenge is identifying the dedicated exact component placement metal from the rest, as they can be submerged beneath the trace routes. The trace routes are analyzed next to determine major signals such as GND or power, as well as basic signals.

Now we will show an example of a comparatively more straightforward case how we ascertain the BoM with a pragmatic description. Later on, we describe the overall flow for our automation algorithm, the systemic details of the frameworks, and its heretofore results.

B. Motivational Example

Consider a single-metal-layer PCB, as shown in Fig. 3(a). We perform two main stratagems: i) placement detection and ii) routing analysis.

The first step is to identify the pins, which are floating pieces of shaped (rectangular/round) grayed metal polygons in Fig. 3(b). After clustering neighboring pins, it is possible to extract corresponding component packages. We pinpointed the package for IC component #2 as being the 7-TO263 package, ascertained from its footprint. However, if we search ICs in a PCB component website such as Digikey [7] for the 7-TO263 package, hundreds of ICs will appear; thereupon, our goal is to extract the correct one.

As for the next step, we performed trace routing analysis. We first anticipated that the most spread trace is ground (GND) (marked in gray in Fig. 3 (c)). Furthermore, this trace is connected to the thermal pad and pin-4 of that detected IC of the 7-TO263 package. We then anticipated that the second widest trace is power, connected to pin-6. Meanwhile, pin-5 is not connected (NC) or floating. Among the ICs in Digikey's 7-TO263 list that partially satisfies the expected pin configurations, as shown in Fig. 3(d), we extracted those with a one-shot trial. We determined that the IC component #2 is LM2676, a voltage regulator [8], and cross-verified all other components on the PCB with the specification schematic, as seen in Fig. 4.

C. Overview of the Approach

Fig. 5 shows the basic steps for MTBoM. First we need to perform metal data processing to define all the metal segments and to differentiate pins from the rest of the metal segments (i.e., traces, planes, and vias). Then the main steps

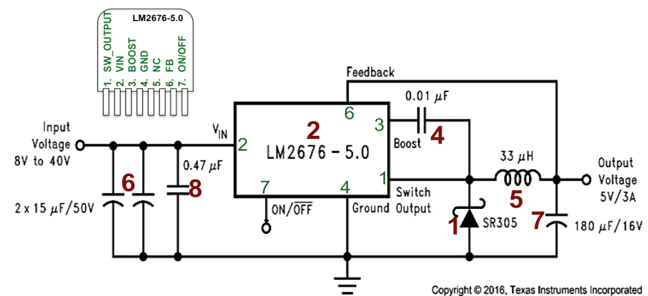


Fig. 4. Retrieved schematic of motivational example; numbers in maroon stand for components as marked in the PCB layout in Fig.3, and the greens refer to the IC pin orientation

Authorized licensed use limited to: Univ of Texas at Dallas. Downloaded on October 09,2022 at 04:05:02 UTC from IEEE Xplore. Restrictions apply.

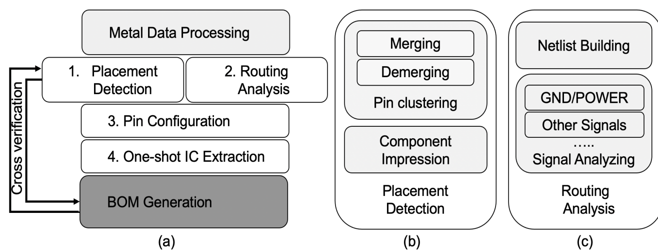


Fig. 5. MTBoM basic steps

are executed, which are: 1) placement detection, 2) routing analysis, 3) pin configuration and 4) one-shot IC extraction. The idea here is to find out the main IC/ICs first. Once all the ICs are extracted as a result of MTBoM, the bill of materials (darkened in Fig. 5(a)) can be generated using the IC-specification documents.

Pseudo code for the MTBoM top-level is demonstrated in Algorithm 1, which can be viewed as sequential executions of subroutines for MTBoM basic steps. We start with the set of all metal segments ($Metals$), the set of all IC packages found via a web crawler ($PackageLib$), and the specifications of the pin configurations for all of those IC packages ($SpecPinTables$). As we process the metal data, (line 1 in Algorithm 1) we distinguish $Pins$ and define them along with all other metal segments ($Metals$), i.e., traces, planes and vias.

Placement detection comprises lines 2-6. In the first step, pins are clustered based on their proximity, pitch and alignment with neighboring pins, as well as their shape. The set $ClusteredPins$ results from a routine PinClustering, in which each cluster of pins represents a component. ComponentImpression then uses the component footprint information to determine its type and package. In the motivational example in Fig. 3(b), after performing this step, we found 9 clusters of pins, i.e., 9 unidentified components. As an outcome of this step, the $Component.Type$ of component#2 is identified as IC and $Component.Package$ is determined to be 7-TO263.

While performing Routing Analysis (lines 7-8), we first execute NetlistBuilding which finds the sets of interconnected metal segments (nets). Then we perform SignalAnalysis, which tags (or names) the nets and corresponding metal segments. Conditionally, metals are tagged such as GND or VDD. As shown in Fig. 3(c), the most spread net is tagged as GND, and the second most as VDD.

During Pin Configuration (line 10 of Algorithm 1), we seek the partial pin configuration for each component. The objective is to name the pins of the ICs using i) already tagged metal segments connected to the pins and ii) analyzing the connectivity to other components. In Fig. 3(d), the thermal pad and pin-4 of component#2 are found as GND, pin-5 is floating, i.e., NC (not connected), and we also preserve its connectivity to other components.

Now for the final MTBoM step, One-Shot IC Extraction, as we have derived the partial pin configuration of all the ICs in the PCB, we compare it with the $SpecPinTables$, which contains the pin configuration of various ICs available. Finally, all the ICs, as outcomes of One Shot IC Extraction (line 14 of Algorithm 1), that satisfy the partial pin configuration are returned. Once we have the information of all the exact ICs on the PCB, we exploit their specifications to predict the bill of materials, as shown in Fig. 5. For example, in the motivational example, LM2676 [8] was matched with the partial pin configuration of component#2; its connectivity is cross verified with the schematic in its spec-sheet to determine the remaining components.

Algorithm 1 MTBoM

Require: $Metals$, $PackageLib$, $SpecPinTables$

1: Identify $Pins$ by **MetalProcessing**($Metals$)
2: Do **PinClustering**($Pins$) by merging neighbours according to their shape, pitch-in-between, alignments and by demerging if applicable
3: **for** each $ClusteredPins$ **do**
4:    Create a $Component$ in the PCB
5:    Define $Component.Package$, $Component.Type$ using **ComponentImpression**($ClusteredPins$) by comparing component to each package in $PackageLib$
6: **end for**
7: Find the set of $nets$ by **NetlistBuilding**($Metals$)
8: Tag the $nets$ and metal segments using **SignalAnalysis**($Metals$)
9: **for** each $Component$ **do**
10:    Perform partial **PinConfiguration** according to the connectivity of the pins to tagged metals
11: **end for**
12: **for** each $Component$ on the PCB **do**
13:    **if** $Component$ is an IC **then**
14:       Find all the $MatchedICs$ for each PCB IC by **OneShotICExtraction**($SpecPinTables$,$Component$) according to partial pin configuration matching
15:    **end if**
16:    Append $MatchedICs$ to $MatchedICs_List$
17: **end for**
18: **return** $MatchedICs_List$

IV. Framework

We have developed an automated tool using Python 2.7.9 [9], where the input is the metal layers and the outputs are all the components, along with all the specific ICs that would have been mounted on the PCB. We will present a comparatively comprehensive description of our basic steps in this section.

A. Input Preparation

Our framework requires as inputs: 1) digitized $Metals$, which includes all metal segments on all layers of the PCB under test, 2) $PackageLib$: a package library containing available IC-package data, and 3) $SpecPinTables$: specification of the pin-outs of the catalog ICs.

Authorized licensed use limited to: Univ of Texas at Dallas. Downloaded on October 09,2022 at 04:05:02 UTC from IEEE Xplore. Restrictions apply.

We have designed a metal parser based on 2D-Cartesian coordinates comprising both the English and metric system that automatically yields for each metal layer:

i) Pins with their shape, width, and location

ii) Traces with the start and endpoints

iii) Metal-planes with their perimeters

iv) Vias with the shape, width, location, and its connecting layers

We have manually prepared an extensive IC-package library, called $PackageLib$ in Algorithm 1, that includes data concerning various aspects of packages, such as pin arrangements (e.g., dual-inline, array, square, with or without thermal pad), assembly technology (e.g., through-hole, surface mounted) and pin pitch (spacing between two pins) such as TQFP, TSSOP, SOT23, SO, UCSP, TQFN, SC70, BGA, etc.

To create an extensive IC pin-out specification table, $SpecPinTables$, we have built a web-crawling tool in Python 2.7 [9] that extracts all the available PDF-specs for the various packages of all ICs it can locate on the Digikey website. Then we have utilized PDFminer [10] as a PDF-parser to build the pin-out tables for any available IC for each specific package.

B. Pin Clustering

Pins are located on the top and bottom metal layers; any piece of shaped metal (mostly rectangular, but also can be square, round or oval) that may be connected to other metal traces and is aligned with another piece of metal of the same type are identified as candidate pins. Here our objective is to cluster the pins that belong to the original component such that each cluster represents the footprint of a component.

Our first approach is to cluster all the pins that are next to each other which have the same size, shape and maintain a certain pitch. In most cases this reveals the footprint of the component. This method works when the pin arrangement is single-in-line (e.g., headers or connectors) or arrayed (e.g., packages such as Ball Grid Array [BGA]). We adapt a merging tool to identify Dual-in-Line (e.g., Thin Shrink Small Outline Package [TSSOP]), Quad in Line (e.g., Thin Quad Flat No Leads [TQFN]), and any package that contains a thermal pad. After pin clustering, the total number of mounted components on the PCB becomes known.

We can cluster any IC component correctly with this pin clustering method. However, for the case of a few non-IC components, if two components of equal pin-pitch are placed maintaining that pitch, it would be detected as a single component. We applied a self-correcting pattern matching demerging tool, which can separate two or multiple merged components in case there exists a third component (boxed in green in Fig. 6(b)) with the same footprint as the original merged components.

Fig. 6 shows how components that are incorrectly merged (boxed in red in Fig. 6(b)) for benchmark ATMEG168 can be unmerged (boxed in black) applying the pattern matching tool. Here our pin clustering results for all other PCBs are shown in Fig. 7, where each bar stands for each PCB measuring the correctly identified components in percentage. Note that the summations of each bar are 100% for all cases, which shows all pins are clustered properly.

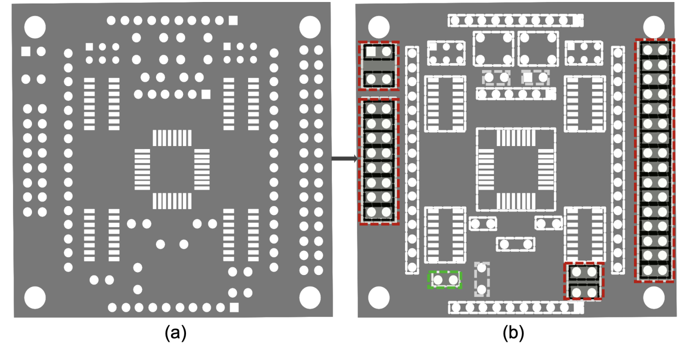


Fig. 6. Pin clustering applying on PCB-ATMEG168, topside pins are laid in light-gray in (a); (b) shows the clustering

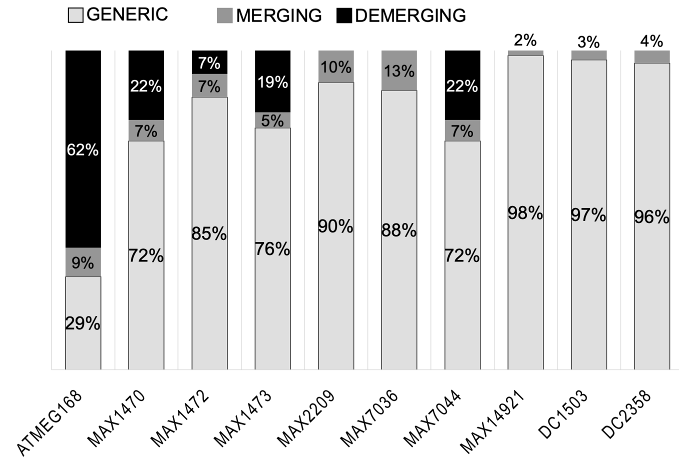


Fig. 7. Pin clustering correction result on all PCBs in the testbench

C. Component Impression

Once pins are clustered to represent each component, we can now determine its type (IC, non-IC) and sub-type (power management units, resistor/capacitor, inductor, headers/connectors) by analyzing the component footprints.

A key goal here is to determine the IC package, as our search procedure on the Digikey Website [7] begins with it, which greatly refines the search. IC packages have significantly different footprints compared to the other PCB components, making it easy to identify them.

D. Routing Analysis

We can define a metal trace on a particular metal layer as a metal line with a start-point, an end-point, and a width; likewise, a metal plane can be defined with its perimeter. Vias between metal layers can be identified with their location, shape, and size. First, to complete the netlist for the entire PCB circuit(s), we build the metal netlist by marking all the traces, planes, pins, and vias connected with a unique net number.

We analyze how metal planes are spread, paying particular attention to how thick the traces are. Furthermore, how much

Authorized licensed use limited to: Univ of Texas at Dallas. Downloaded on October 09,2022 at 04:05:02 UTC from IEEE Xplore. Restrictions apply.

area each metal plane contains. The most significant metal segment is identified as ground (GND, VSS, AGND). Then, we mark the second-largest metal plane as VDD and the third-largest metal plane as VDD2, such that they would be candidates for any power supply pins (e.g., VIN, VOUT, VCC, VA). If any unique metal structures for Radio Frequency (RF) pins are used, we tag the pins connected directly (or through a resistor) to those metal structures as RF. We tagged each metal trace, pin, plane, and via according to the identified type.

E. Pin Configuration

Pins are configured according to the trace type it is connected to. On the other hand, if any pin is open or only connected to a floating trace, it is identified as a Not Connected (NC) pin. We identify candidate pins for Clock (CLK) or Enable (EN), as they are comparatively thicker and common among all ICs in a PCB. If a group of pins in a component have the same destination, we identify them as data.

Our matching tool comprises seven pin-configuration matching steps for: i) GND, ii) VDD, iii) CLK/EN, iv) NC, v) RF, vi) Data, and vii) pins that are eligible to be shorted internally.

F. One-shot IC extraction

IC extraction uses the extensive IC pin-out specification table, $SpecPinTables$, described earlier, that has the pin-out tables for any available IC for each package. During the IC extraction procedure, we apply a matching tool to compare the pin configuration of the ICs in the PCB, considering all the possible rotations of an IC for a specific package, with all the pin tables in $SpecPinTables$.

When we examine if a pin satisfies GND matching or VDD matching, we perform two-way verification: i) if a pin in the spec is named as GND or any power, it has to be tagged in the PCB accordingly; ii) if a pin is connected to GND or VDD in the PCB that cannot violate the connection in the spec, for example, if a pin is grounded in the PCB, but that pin is named as CLOCK in PDF-spec, it would not satisfy GND matching and that IC would be excluded from our search. In all other cases, we perform one-way verification; if any pin specified in a spec is named CLK, EN, NC, RF, it must be tagged accordingly in the PCB. For data bus matching, if at least half of the data pins in the spec travel together on the PCB, it is still considered as a data bus match. For the case of checking if pins are eligible to be shorted, we check whether those pins are named the same in the spec; or if they are the feedback pins from outputs to the negative inputs of op-amps.

Once the ICs are determined along with their specs, we can use the specs to determine the entire bill of materials for the PCB.

V. RESULTS

A. Experimental Set Up

We applied MTBoM on ten randomly selected PCBs from three different open sources [11], [12], [13]. Table I shows the features and functionality of those PCBs.

Human supervised analysis: We also analyzed matching ICs for their i) type and ii) frequency range of operation. In the case of IC MAX9620 in PCB#8, the tool extracts 8 out of 34 ICs in the 5-SC70 package by partial pin configuration matching. However, only some of those 8 are amplifiers; they perform similar functionality but cover different operating ranges. On the other hand, four others are comparators and a pre-amplifier. With trace analysis, we observed that its OUT pin is shorted to the VIN-, which is not the typical case for a comparator, so this IC should not be a comparator. Therefore, among the remaining 4, only MAX9620 can cover the frequency range of 250 kHz, which is required for another IC MAX11163 in PCB#8 whose operation range is up to 250 kHz. Instead of the IC MAX11163, another candidate, IC MAX11168, was also extracted by MTBoM. If MAX11163 was replaced by MAX11168, whose frequency range is 500 kHz, it would have to be paired with an amplifier that has such a frequency range. However, only MAX9620 can operate at 500 kHz; therefore, MAX9620 remains the only candidate.

Table II shows the conclusive results for MTBoM. When we found additional ICs other than the actual one, we confirmed that none of the additional ICs were a false positive. Instead, those ICs perform the exact functionality with the same features but may cover different operating ranges or are from different manufacturers.

B. Extraction Tool Effectiveness

Fig. 8 uses a stacked bar graph to show the percentage of correctly identified ICs for each PCB after package identification and after application of each of the IC extraction matching steps, which also shows the significance of that particular pin matching step for extracting a particular IC.

Consider, N_{pkg} is the total number of ICs auto-extracted from DigiKey that could use the same package. N_i is the number of extracted candidate ICs obtained after applying pin matching step-i, where i represents the matching criteria. N_f represents the final number of ICs remaining after applying MTBoM.

For example, for PCB#1, the first IC is ATMEG168 in a 32-TQFP package. After package matching $N_{pkg} = 23$, but there is only one actual IC among them ($N_f = 1$), which

TABLE I
PCBs UNDER TEST

PCB#	Main IC	IC Cnt.	Pin Cnt.	Cmp Cnt.	Met. Lyr.	Trace Cnt.	Functionality
1	ATMEG168	5	248	55	2	946	Micro-controller
2	MAX1470	1	205	46	2	1244	Hetero. Receiver
3	MAX1472	2	107	41	4	888	VHF Transmitter
4	MAX1473	1	242	62	2	1619	Hetero. receiver
5	MAX2209	1	28	10	4	515	RF detector
6	MAX7036	1	155	22	4	1104	Receiver
7	MAX7044	2	115	46	4	797	VHF Transmitter
8	MAX14921	4	466	64	4	8045	Status Monitor
9	DC1503	1	131	35	4	2497	trans-receiver
10	DC2358	1	75	25	4	10594	DC-DC w/t LDO

Authorized licensed use limited to: Univ of Texas at Dallas. Downloaded on October 09,2022 at 04:05:02 UTC from IEEE Xplore. Restrictions apply.

TABLE II
MTBoM RESULTS
WE REPORTS FOR EACH PCB#, THE PART# OF THE ICS, IC PACKAGES (PKG), SAME PACKAGE IC-COUNT IN $SpecPinTables$ (N_{pkg}), FINAL IC-COUNT AFTER EXTRACTION (N_f), IF THE ORIGINAL IC REMAIN (ORG. IC), FALSE+ (F+), FALSE- (F-)

PCB#	Part#	Pkg	N_{pkg}	N_f	Org. IC	F+	F-
1	ATMEG168	32-TQFP	23	1	✓	x	x
	74HC595	16-TSSOP	79	1	✓	x	x
2	MAX1470	28-TSSOP	63	1	✓	x	x
3	MAX1472	8-SOT23	33	2	✓	x	x
	ICM175	8-SO	144	1	✓	x	x
4	MAX1473	28-TSSOP	63	2	✓	x	x
5	MAX2209	4-UCSP	18	1	✓	x	x
6	MAX7036	21-TQFN	45	1	✓	x	x
7	MAX7044	8-SOT23	33	2	✓	x	x
	ICM1755	8-SO	144	1	✓	x	x
8	MAX14921	80-TQFP	7	1	✓	x	x
	MAX6126	8-SO	144	1	✓	x	x
	MAX11163	10-UMAX	15	2	✓	x	x
	MAX9620	5-SC70	34	1	✓	x	x
9	DC1503	32-BGA	5	1	✓	x	x
10	DC2358	38-BGA	4	1	✓	x	x

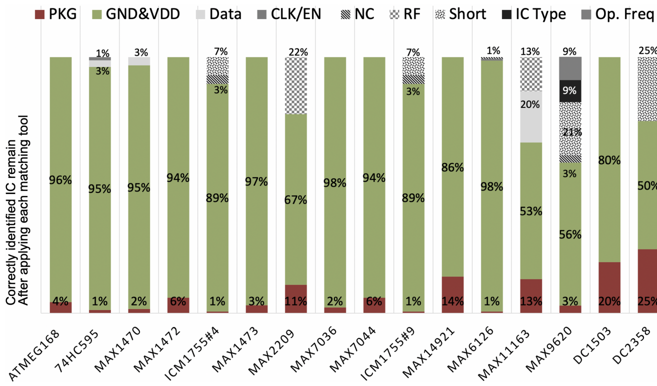


Fig. 8. Percentage of correctly identified ICs stacked after the application of each successive extraction tool, for each of the IC(s) on PCBs under test.

means the percentage of correct ICs with package matching is $N_f/N_{pkg} = 1/23 = 4.3\%$ (the maroon bar in Fig. 8); GND & VDD matching eliminate 22 other ICs (the green bar $(23-1)/23 = 95.7\%$ in Fig. 8), therefore the final percentage of correct ICs is 100%.

This PCB#1 has other 4 ICs in a 16-TSSOP package; for those 4 ICs, $N_{pkg} = 79$. After applying GND and VDD matching the remaining IC count, $N_{gnd\&vdd} = 4$, likewise: $N_{data} = 2$, $N_{clk/en} = 1$, after each of those matching steps was applied. In this case, there is also one correct IC out of 79 different ICs having the same package (as each of those four ICs is 74HC595). Therefore, the corresponding stack bar value from the ground would become $1/79, (79-4)/79, (4-2)/79, (2-1)/79$. Note that all the stacked bars' final results are 100% as finally no false+ or false- remain.

The computation time recorded for each step of our tool (excluding the $PackageLib$ and $SpecPinTables$ preparation time, which only has to be done once for any number of applications of MTBoM) is shown in Fig. 9. The significant time contribution that directly depends on the number of metals is the netlist construction time. The maximum time recorded was 143.2 seconds for the DC2358 PCB that consisted of 75 pins and 10594 traces on a PC running the Red Hat Linux 7.8 operating system.

VI. CONCLUSIONS

We have presented MTBoM. Our scheme can accurately extract the bill of materials by analyzing the metal layers. We demonstrate that MTBoM can detect all ICs on a set of 10 PCBs with no false positives or negatives. This scheme also identifies passive components, such as resistors, capacitors, and inductors.

REFERENCES

[1] N. Asadizanjani, M. Tehranipoor, and D. Forte, "Pcb reverse engineering using nondestructive x-ray tomography and advanced image processing," *IEEE Transactions on Components, Packaging and Manufacturing Technology*, vol. 7, no. 2, pp. 292–299, 2017.

[2] J. Grand, "Printed circuit board deconstruction techniques," in *8th USENIX Workshop on Offensive Technologies (WOOT 14)*, 2014.

[3] S. Ghosh, A. Basak, and S. Bhunia, "How secure are printed circuit boards against trojan attacks?" *IEEE Design Test*, vol. 32, no. 2, pp. 7–16, 2015.

[4] S. Paley, T. Hoque, and S. Bhunia, "Active protection against pcb physical tampering," in *2016 17th International Symposium on Quality Electronic Design (ISQED)*, 2016, pp. 356–361.

[5] F. Domke, "Blackbox jtag reverse engineering," in *Corpus ID: 1567451*, 2009.

[6] H. G. Longbotham, Ping Yan, H. N. Kothari, and Jun Zhou, "Nondestructive reverse engineering of trace maps in multilayered pcbs," in *Conference Record AUTOTESTCON '95. 'Systems Readiness: Test Technology for the 21st Century'*, 1995, pp. 390–397.

[7] DigiKey, *DigiKey Product Search Page*, 2017, https://www.digikey.com.

[8] Texas_Instruments, *TI LM2676 Specification*, http://www.ti.com/lit/ds/symlink/lm2676.pdf.

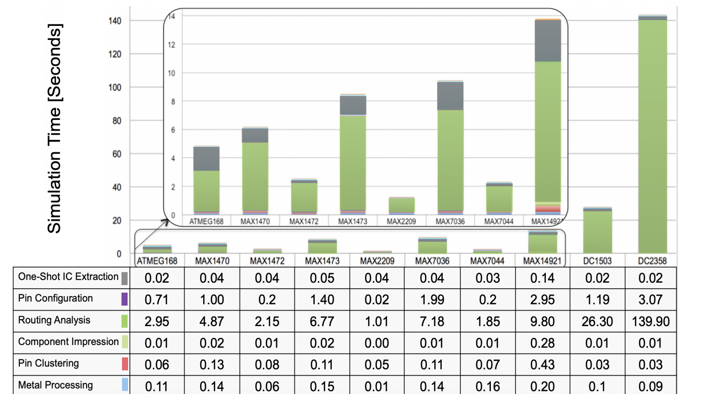
[9] Python, *Python Python 2.7.9 Release*, 2014, https://www.python.org/dev/peps/pep-0373/.

[10] python, *pdfminer, 2015*, https://buildmedia.readthedocs.org/media/pdf/pdfminer-docs/latest/pdfminer-docs.pdf.

[11] Madworm, *Madworm 8x8 RGB LED Matrix Controller*, 2015, https://github.com/madworm/.

[12] Maxim, *Maxim Gerber Files*, 2016, https://www.maximintegrated.com/en/design/tools/.

[13] Linear, *Linear Evaluation board Kits*, 2016, https://www.analog.com/en/design-center/evaluation-hardware-and-software/evaluation-boards-kits.html/.



	ATMEG168	MAX1470	MAX1472	MAX1473	MAX2209	MAX7036	MAX7044	MAX14921	DC1503	DC2358
One-Shot IC Extraction	0.02	0.04	0.04	0.05	0.04	0.04	0.03	0.14	0.02	0.02
Pin Configuration	0.71	1.00	0.2	1.40	0.02	1.99	0.2	2.95	1.19	3.07
Routing Analysis	2.95	4.87	2.15	6.77	1.01	7.18	1.85	9.80	26.30	139.90
Component Impression	0.01	0.02	0.01	0.02	0.00	0.01	0.01	0.28	0.01	0.01
Pin Clustering	0.06	0.13	0.08	0.11	0.05	0.11	0.07	0.43	0.03	0.03
Metal Processing	0.11	0.14	0.06	0.15	0.01	0.14	0.16	0.20	0.1	0.09

Fig. 9. Simulation time for each PCB.

Authorized licensed use limited to: Univ of Texas at Dallas. Downloaded on October 09,2022 at 04:05:02 UTC from IEEE Xplore. Restrictions apply.