An Efficient Graph-Based Algorithm for ESD Current Path Analysis

Chih-Hung Liu, Hung-Yi Liu, Chung-Wei Lin, Szu-Jui Chou, Yao-Wen Chang, *Member, IEEE*, Sy-Yen Kuo, Shih-Yi Yuan, and Yu-Wei Chen

***Abstract*—The electrostatic discharge (ESD) problem has become a challenging reliability issue in nanometer-circuit design. High voltages that resulted from ESD might cause high current densities in a small device and burn it out, so on-chip protection circuits for IC pads are required. To reduce the design cost, the protection circuit should be added only for the IC pads with an ESD current path, which causes the ESD current path analysis problem. In this paper, we first introduce the analysis problem for ESD protection in circuit design. We then model the circuit as a constraint graph, decompose the ESD connected components (ECCs) linked with the pads, and apply breadth-first search (BFS) to identify the ECCs in each constraint graph and, thus, the current paths. Experimental results show that our algorithm can very efficiently and economically detect all ESD paths. For example, our algorithm can detect all ESD paths in a circuit with more than 1.3 million vertices in 1.39 s and consume only 44-MB memory on a 3.0-GHz Intel Pentium 4 PC. To the best of our knowledge, our algorithm is the *first point tool* available to the public for the ESD analysis.**

***Index Terms*—Analysis, electrostatic discharge (ESD), graph search, network flow, reliability.**

I. Introduction

THE PHENOMENON of electrostatic discharge (ESD) exists everywhere in our daily life, such as a standing hair or an electric shock by a doorknob. It occurs when an electrostatic voltage develops and discharges as a current impulse. Although ESD only induces a little discomfort to humans, it can cause great damage in semiconductor fabrication. In a practical situation, ESD often occurs between two or more devices with different electrostatic potentials, and the current impulses generated by ESD may break circuits and burn devices out. For example, for a 0.13-μm CMOS device designed for operation at 1.2 V, the voltage drop across a 2-Ω power bus exceeds 20 V and burns out the ultrathin gate oxides [2]. As the process technology enters the nanometer era, device size has continued to shrink, and the breakdown voltage of the thin-oxide devices is usually less than 5 V, making the ESD damage occur easily and difficult to prevent [3]. As a result, the prevention of ESD becomes one of the major concerns for IC reliability.

There are many solutions to the ESD problem [2], examples of which are the use of antistatic coatings to prevent static charge generation in wafers, the use of shielded materials to prevent ESD resulted from human handling, and the implementation of protection circuits within the chip. For the three widely used ESD discharge models [4], [5], we can categorize them into two major classes.

1) Human body model/machine model (HBM/MM): An external voltage induces a discharge current through a pair of pads.
2) Charged device model (CDM): The charge is accumulated on the chip itself and then discharged through a single pad.

In HBM/MM, if there exists a current path with very low impedance between two pads, this path will suffer a large ESD current, and some devices on this path could be burnt out. As a result, to protect an IC chip from the HBM/MM ESD damage, a protection circuit for a pair of pads needs to: 1) provide a low impedance path between the two pads to safely discharge a large ESD current and 2) clamp the pad voltage between them to a sufficiently low level [6]. In practice, an industrial flow to protect the HBM/MM ESD damage is typically divided into two stages, i.e., design and verification, as shown in Fig. 1. For protection circuit design, we construct protection circuits for pairs of pads; for protection circuit verification, we estimate the protection strengths (i.e., ESD sensitivity levels) for the designed protection circuits and then modify the circuits with insufficient protection strengths. Here, a protection circuit could be a back-to-back diode with low impedance, and a protection strength estimation could be performed by some simulation-based methods [7].

However, if protection circuits are constructed for all pairs of n pads, we would need $n(n-1)/2$ protection circuits at the protection circuit design stage. For a modern system-on-chip

Manuscript received August 21, 2007; revised January 13, 2008. This work was supported in part by Incentia Design Systems and in part by the National Science Council of Taiwan under Grant NSC 96-2221-E-002-023, Grant NSC 96-2628-E-002-248-MY3, Grant NSC 96-2628-E-002-249-MY3, and Grant NSC 96-2221-E-002-245. This paper was recommended by Associate Editor D. Z. Pan.

C.-H. Liu is with the Graduate Institute of Electronics Engineering, National Taiwan University, Taipei 106, Taiwan, R.O.C.

H.-Y. Liu and C.-W. Lin were with the Graduate Institute of Electronics Engineering, National Taiwan University, Taipei 106, Taiwan, R.O.C. They are currently serving in the military in R.O.C.

S.-J. Chou is with the Implementation Group, Synopsys Taiwan Ltd., Taipei 110, Taiwan, R.O.C.

Y.-W. Chang is with the Graduate Institute of Electronics Engineering and the Department of Electrical Engineering, National Taiwan University, Taipei 106, Taiwan, R.O.C. (e-mail: ywchang@cc.ee.ntu.edu.tw).

S.-Y. Kuo is with the Department of Electrical Engineering, National Taiwan University, Taipei 106, Taiwan, R.O.C. (e-mail: sykuo@cc.ee.ntu.edu.tw).

S.-Y. Yuan is with the Department of Communications Engineering, Feng Chia University, Taichung 407, Taiwan, R.O.C.

Y.-W. Chen is with the Design and Development Department, Faraday Technology Corp., Hsinchu 300, Taiwan, R.O.C.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCAD.2008.925779

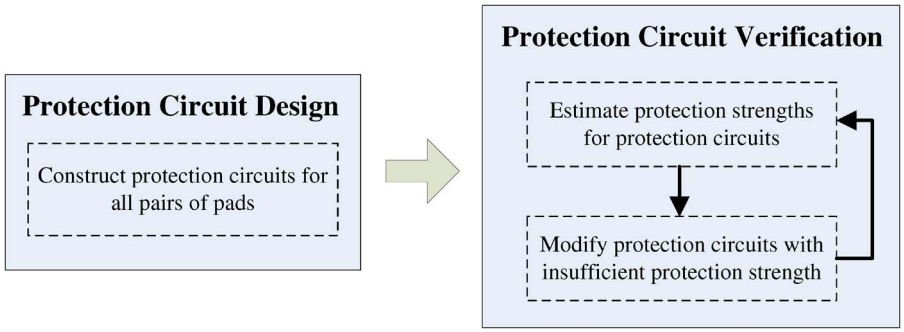


Fig. 1. Typical industrial flow for HBM/MM ESD protection.

design, it often consists of hundreds of pads in a circuit. Consequently, there may be hundreds of thousands of pad pairs to be considered for ESD protection, but most of the pad pairs might be safe from ESD damage. The large number of unnecessary protection circuits substantially increases the design and the protection circuit verification costs. In practice, a current path between two pads through at most one MOS transistor gate may suffer from HBM/MM ESD damage, whereas one through more than one MOS transistor gate could be considered as open. Hence, for practical considerations, we only need to protect a pair of pads between which there exists a current path through at most one MOS transistor gate (denoted as an ESD current path later in this paper). Furthermore, based on the real-world designs from Faraday Technology Inc., the number of ESD current paths is typically far under $n(n-1)/2$. Therefore, ESD current path analysis can be employed at the protection circuit design stage to avoid an overdesign with unnecessary protection circuits, and thus, the ESD circuitry cost can be substantially reduced. Furthermore, ESD current path analysis can also reduce the simulation time of the subsequent protection circuit verification.

As a relatively new design challenge, there is not much EDA work on ESD protection in the literature (although there are a number of publications on the circuit design for ESD protection [2], [3]). Very recently, Hayashi *et al.* [7] proposed a full-chip analysis method of the ESD protection network to analyze pad voltage and maximum gate voltage (could be considered as protection strength) for all pad pairs in HBM. Qian *et al.* [8] proposed a chip-level method to simulate CDM ESD events. In addition, Zhan *et al.* showed how to verify the protection circuit design, considering parasitic and devices at the post-layout stage [9], [10]. The aforementioned works [7]–[10] focus on the protection circuit verification (see the flow shown in Fig. 1).

In this paper, we introduce the ESD current path analysis problem for HBM/MM ESD protection at the protection circuit design stage. We first consider the case where the ESD voltage burns out at most one transistor, which is the typical case for current applications. To detect those pairs of pads with ESD current paths between them, we model the circuit as a constraint graph, decompose the ESD connected components (ECCs) linked with the pads, reduce the graph, and apply the breadth-first search (BFS) to identify the ECCs (i.e., ESD current paths) in each constraint graph. We also extend our algorithm to the case where an ESD voltage may be large enough to destroy more than one transistor. Experimental results show that our algorithm can very efficiently and economically detect all ESD paths. For example, our algorithm can detect all ESD paths in a circuit with more than 1.3 million vertices in 1.39 s and consume only 44 MB memory on a 3.0-GHz Intel Pentium 4 PC with 2-GB random access memory. In contrast, the well-known Johnson's all-pair shortest-path algorithm needs more than 415 s and 415-MB memory. It should be noted that, to the best of our knowledge, our algorithm is *the first point tool* available to the public for the ESD analysis (ESDA) at the protection circuit design stage.

The rest of this paper is organized as follows: Section II formulates the ESD current path analysis problem. Section III presents the algorithm to identify all ESD current paths between pads for the case where the ESD voltage burns out at most one transistor. Section IV extends the problem to the general case where the ESD voltage can destroy more than one transistor. Section V reports the experimental results. Finally, we conclude this paper in Section VI.

II. Problem Formulation

Given a netlist with the circuit hierarchy, we define a *circuit block*, a *pad*, and a *current path* as follows.

Definition 1: A *circuit block* is a circuit or a subcircuit containing the following components: resistors, diodes, MOS transistors, and other circuit blocks. A *top-level circuit block* is the topmost block in the circuit hierarchy. A netlist has at least one circuit block and exactly one top-level circuit block.

It should be noted that it is sufficient to consider MOS transistors, diodes, and resistors for modern ESD protection. A capacitor can reduce ESD-induced voltage and thus even alleviate the ESD effect, and modern digital designs seldom consider inductors for on-chip ESD protection. Therefore, capacitors and inductors can be ignored in current practical applications for ESD protection. Furthermore, our modeling is general and even applicable to non-MOS circuits, such as bipolar junction transistors, which can similarly be modeled as MOSs. (See Section III-B1 for more detail.)

Definition 2: A *pad* is an I/O pin of a circuit block.

Definition 3: A *current path* is a path that can contain diodes and resistors and propagate between the source and the drain, the gate and the source, or the gate and the drain in a MOS transistor.

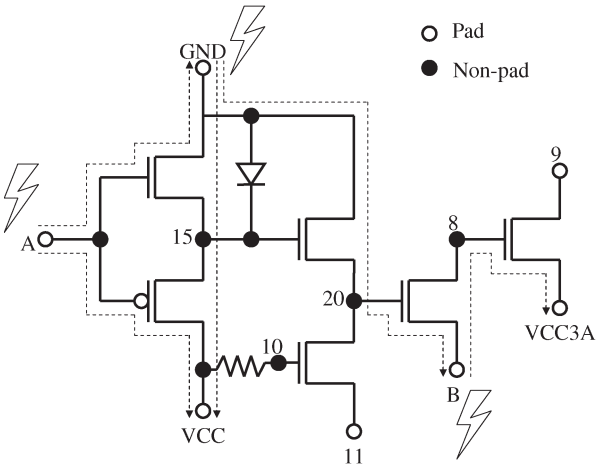


Fig. 2. Sample netlist and its ESD paths denoted by the dotted lines.

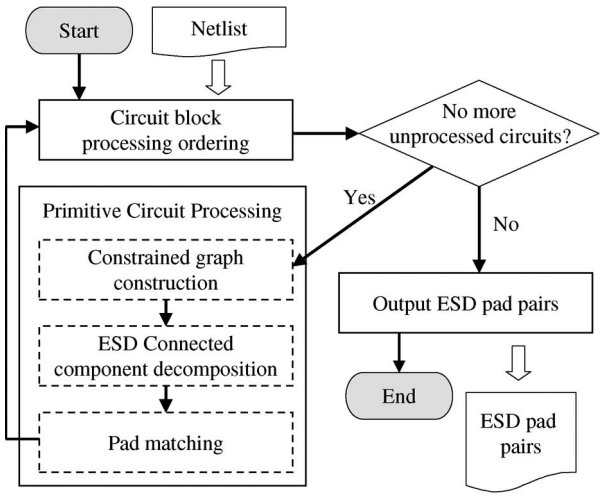


Fig. 3. Flowchart of the proposed approach.

As mentioned in Section I, devices on a current path through at most one MOS transistor gate could be burnt out under an HBM/MM ESD event. We define an ESD current path as follows:

Definition 4: An *ESD current path* is a current path that propagates between the gate and the source or the gate and the drain of a MOS transistor at most once.

Our objective is to detect all pairs of pads that need protection circuits for the practical consideration. Therefore, we do not consider an ESD current path between two pads with another pad on the path. This is because, if an ESD current path between two pads includes other pads, there could be redundant protection circuits. For instance, assuming that there is a pad p_2 in the ESD current path from p_1 to p_3, it is clear that there exist ESD current paths from p_1 to p_2 and from p_2 to p_3; that is, there are protection circuits for the pad pairs (p_1, p_2) and (p_2, p_3). As a result, an external voltage from p_1 to p_3 could be discharged through the protection circuits for (p_1, p_2) and (p_2, p_3), implying that the protection circuit for (p_1, p_3) is redundant. Considering practical circuit operations, we further define an *ESD path* as follows.

Definition 5: An *ESD path* is an ESD current path passing through no other intermediate pads, except the two terminal pads of the path.

Note that the ESD path is undirected since we can ignore the direction of current by using a symmetric back-to-back diode to protect an ESD path. With the aforementioned definitions, we can formulate the addressed problem given here.

- *The ESDA Problem*: Given a netlist with circuit hierarchy, find every pair of pads with an ESD path between them for ESD circuit protection.

For example, as shown in Fig. 2, an ESD current can propagate from A to VCC and GND through a MOS transistor. However, the current cannot propagate from A to B according to the definition that an ESD path includes at most one gate–source or gate–drain path. Therefore, there are, in total, five pairs of pads between which there is an ESD path: (A, VCC), (A, GND), (B, VCC3A), (B, GND), and (VCC, GND).

III. Algorithm

As shown in Fig. 3, a bottom–up approach is proposed to exploit the circuit hierarchy. For a hierarchical circuit, a high-level circuit block could embed low-level circuit blocks through circuit references. Since the low-level circuit blocks may be frequently referenced, it is inefficient to top–down (recursively) process each subcircuit block once it is referenced. Therefore, we shall apply a bottom–up circuit block processing ordering to speed up the processing (see Section III-A).

Definition 6: A *primitive circuit block* is defined as a circuit block that references no subcircuit block or the subcircuit blocks whose ESD paths have been detected.

For a primitive circuit block, detecting its ESD paths is different from the traditional reachability problem [11], due to the ESD path definition (see Definition 5). In this paper, we first model a primitive circuit block as a constraint graph (see Section III-B1). Given a constraint graph, the ECCs in the graph are decomposed (see Section III-B2). After that, the stage of pad matching detects all the ESD paths of the primitive circuit block (see Section III-B3). Finally, we prove the correctness of our algorithm.

A. Circuit Block Processing Ordering

Hierarchical circuit block references can be represented by a topology tree. These references form a processing order among circuit blocks. As shown in Fig. 4, where circuit block TOP references circuit blocks A and B, this example implies that A and B have to be processed, i.e., have their ESD paths detected, before TOP. Thus, when detecting the ESD paths of TOP, we have already known the ESD paths of the blocks referenced by TOP. Performing postorder traversal (POT) on the topology tree defines a *feasible processing order* for the circuit blocks. A processing order P_f is *feasible* if, for each circuit block b in P_f, all subcircuit blocks of b have been processed before b in P_f. For example, the POT order in Fig. 4 $\langle \mathrm{F}, \mathrm{F}, \mathrm{C}, \mathrm{D}, \mathrm{A}, \mathrm{F}, \mathrm{F}, \mathrm{C}, \mathrm{F}, \mathrm{E}, \mathrm{B}, \mathrm{TOP} \rangle$ is a feasible processing order, in which $\{\mathrm{F}\}$ is processed before C, $\{\mathrm{F}, \mathrm{C}, \mathrm{D}\}$ are processed before A, and so on.

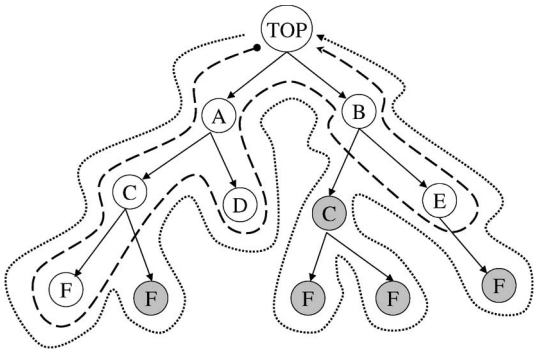


Fig. 4. Hierarchical-circuit topology tree example. The nodes and edges represent circuit blocks and circuit block references, respectively. In this example, TOP references A and B, A references C and D, and so on. The dotted line traverses the tree in a postorder, i.e., $\langle$F, F, C, D, A, F, F, C, F, E, B, TOP$\rangle$. The dashed line traverses the tree in a postorder with redundancy removal, i.e., $\langle$F, C, D, A, E, B, TOP$\rangle$, in which the gray nodes are redundant (already visited).

However, since a subcircuit block can repeatedly be referenced, it is not necessary to detect its ESD paths whenever it is referenced. As illustrated in Fig. 4, performing POT with redundancy removal on the topology tree defines an *irredundant processing order* for the circuit blocks. A processing order P_i is *irredundant* if P_i is feasible and each circuit block b appears in P_i exactly once. In the topology tree example, the processing order $\langle$F, C, D, A, E, B, TOP$\rangle$ is an irredundant order. To remove the redundancy during POT, we establish a hash table saving processed circuit blocks and their ESD paths that have been detected, for the redundancy check and for the ESD path lookup. Thus, we can efficiently detect all ESD paths bottom–up in the top-level circuit block.

B. Primitive Circuit Processing

A primitive circuit block references no subcircuit block or subcircuit blocks whose ESD paths have been detected (see Definition 6). The following three stages process a primitive circuit block:

1) constraint graph construction: modeling of the netlist describing the primitive circuit block as a constraint graph;
2) ECC decomposition (ECCD): detection of all ECCs linked with the pad vertices of the constraint graph;
3) pad matching: pairing of the pads bridging the ESD paths according to the connection of the detected ECCs.

1) Constraint Graph Construction: A *constraint graph* models the device interconnection in a primitive circuit block. Physical device terminals (connections) are modeled as graph vertices (edges). There are two types of vertices as follows:

1) pad vertex (p-vertex), which represents the pad of the primitive circuit block;
2) nonpad vertex (np-vertex), which represents all the device terminals, except the pads.

Edges are also classified into two categories as follows:

1) constrained edge (c-edge), which can be involved in an ESD path at most once;
2) nonconstrained edge (nc-edge), which has no occurrence limit appearing in an ESD path.

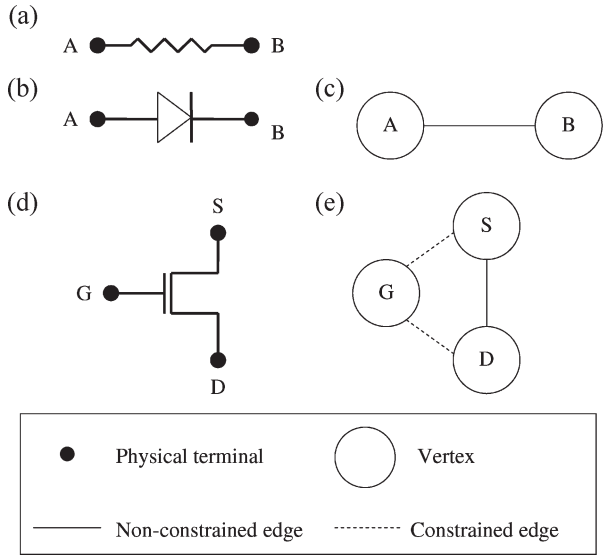


Fig. 5. (a) Resistor with its two terminals. (b) Diode with its two terminals. (c) Constraint graph model of a resistor/diode. (d) MOS transistor with its three terminals: gate, source, and drain (denoted by G, S, and D, respectively). (e) Constraint graph model of a MOS transistor.

Using these predefined vertices and edges, we construct the constraint graph according to three device types.

1) *Resistors and diodes*: Since a current can always propagate through resistors and diodes, two corresponding vertices of terminals of these devices are connected by an nc-edge, as shown in Fig. 5(a)–(c). In addition, a vertex is labeled as a p-vertex (np-vertex) if it is (is not) a pad of the primitive circuit block.
2) *MOS transistors*: An MOS transistor is referenced with three terminals: gate, source, and drain. To meet the first constraint of the ESD path definition (Definition 5), each gate–source and gate–drain path is modeled by a c-edge. On the other hand, we use an nc-edge to model a source–drain path. Fig. 5(d) and (e) shows the graph modeling for a MOS. In addition, a vertex is labeled as a p-vertex (np-vertex) if it is (is not) a pad of the primitive circuit block.
3) *Subcircuit references*: A primitive circuit block references blocks whose ESD paths have been detected (see Definition 6). Assume that there are two terminals t_1 and t_2 in a primitive circuit block and their corresponding vertices in the constraint graph are v_1 and v_2. If t_1 and t_2 are connected to two pads of a referenced block, i.e., p_1 and p_2, respectively, we can determine the connection types of v_1 and v_2 by looking up the connection types of p_1 and p_2 through the access to the hash table established in Section III-A. All cases are listed here.
   a) If p_1 and p_2 are connected by an ESD path involving no c-edge, add an nc-edge between v_1 and v_2.
   b) If p_1 and p_2 are connected by an ESD path involving one c-edge, add a c-edge between v_1 and v_2.
   c) If p_1 and p_2 are not connected by any ESD path, do nothing.

   Finally, label v_1 (and v_2) as a p-vertex if it is also a pad of the primitive circuit block; label it as an np-vertex, otherwise.

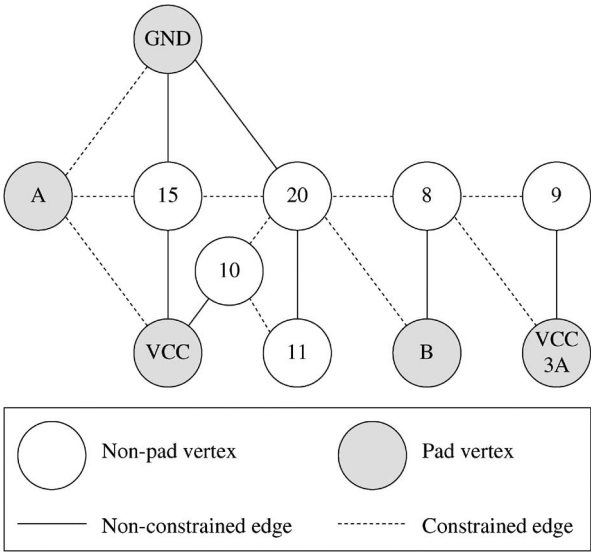


Fig. 6. Constraint graph constructed from the netlist in Fig. 2, using the construction methods presented in Section III-B1.

A constraint graph example is given in Fig. 6, which is constructed from the netlist shown in Fig. 2, using the construction methods presented in this section. From the constraint graph construction, we immediately have the following property.

Property 1: In the constraint graph constructed from a primitive circuit, an ESD path is represented as a path including at most one c-edge and any number of nc-edges.

2) ECCD: In this section, we propose an ECCD algorithm (see Fig. 7) that decomposes the ECCs linked with p-vertices from a constraint graph. Using the ECCD algorithm, we can reduce the solution space to increase the efficiency.

Definition 7: Let $G = (V, E)$ be a constraint graph, where V and E are the vertex set and the edge set, respectively. Let $V_{\rm np} \subseteq V$ be the np-vertex set and $E_c \subseteq E$ be the c-edge set. An *ECC* is a maximal vertex set $C \subseteq V_{\rm np}$, where $\forall v_i, v_j \in C$, $i \neq j$, there is a path between v_i and v_j, and this path involves no edge $e \in E_c$.

By Definition 7, an ECC of a constraint graph contains only np-vertices, and for each np-vertex in the ECC, there is always a path leading to every other np-vertices in the ECC by nc-edges. Hence, according to Property 1, an ECC can be considered as a vertex when finding ESD paths to reduce the solution space. It will be clear later in Theorem 1 that it is sufficient to only decompose ECCs that are linked with p-vertices to substantially improve the efficiency. As a result, our objective in this stage is to find three kinds of elements that relate the: 1) ECCs and 2) ESD paths between p-vertices in a constraint graph, which are given as follows:

1) all the ECCs linked with p-vertices in the constraint graph;
2) p-vertices linked to each ECC in 1) by c-edges or nc-edges;
3) precedent neighboring ECCs linked to each ECC in 1) by c-edges.

Among them, we define the precedent neighboring ECCs here.

Algorithm: $ECCD(C_p, H, G, S_{ECC})$
Input: C_p /* primitive circuit block ID */
H /* global hash table */
$G = (V, E)$ /* constrained graph of C_p */
Output: S_{ECC}
/* set of ESD connected components */
1 Clear S_{ECC}
2 Clear ESD connected component $C(id, V_p, S_{PN})$
/* id: ID of C */
/* V_p: set of p-vertex linked to C */
/* S_{PN}: set of precedent neighboring ESD */
/* connected components of C */
3 Clear queue Q
4 Component count $n = 0$
5 **for** each p-vertex $v_p \in V$
6 **for** each vertex v_i adjacent to v_p
7 **if** v_i is a p-vertex /* a trivial ESD path */
8 $H(C_p) \leftarrow H(C_p) \cup \{path(v_i, v_p)\}$
9 Go to line 6
/* to find an ESD connected component */
10 **if** v_i is not visited
11 n++ /* find a new component */
12 $C.id = v_i.componentID = n$
13 $v_i.visit = true$
14 Q.push(v_i)
15 **while** Q is not empty /* BFS starts */
16 $v_j = Q$.first()
17 **for** each v_k adjacent to v_j
18 **if** v_k is a p-vertex
19 $C.V_p \leftarrow C.V_p \cup \{(v_k, edge(v_j, v_k))\}$
20 **else if** v_k is visited
/* v_k here is an np-vertex */
21 **if** edge(v_j, v_k) is a c-edge
22 $C.S_{PN} \leftarrow C.S_{PN} \cup \{v_k.componentID\}$
23 **else if** $edge(v_j, v_k)$ is an nc-edge
/* v_k here is not visited */
24 $v_k.visit = true$
25 $v_k.componentID = n$
26 Q.push(v_k)
27 Q.pop() /* BFS ends */
28 $S_{ECC} \leftarrow S_{ECC} \cup \{C\}$
29 Clear C
30 Return S_{ECC}

Fig. 7. ECCD algorithm.

Definition 8: Assume that the ECCD algorithm decomposes ECCs in the following order: $P_{\rm ECC} = \langle {\rm ECC}_1, {\rm ECC}_2, \ldots, {\rm ECC}_n \rangle$, where ${\rm ECC}_i$ is an ECC, $1 \leq i \leq n$. The *precedent neighboring ECCs* of ${\rm ECC}_i$ is a set $PN^i = \{{\rm ECC}_j | {\rm ECC}_j \in P_{\rm ECC}, j < i,$ and ${\rm ECC}_j$ is linked to ${\rm ECC}_i$ by c-edges$\}$.

Using the constraint graph in Fig. 6 as an example, we explain the ECCD algorithm described in Fig. 7 to decompose the ECCs linked with p-vertices. After the initialization steps in lines 1–4 of the ECCD algorithm, we assume that the p-vertex enumeration order in line 5 is $\langle {\rm GND}, {\rm VCC3A}, {\rm B}, {\rm VCC}, {\rm A} \rangle$. Starting from the p-vertices, v_p is GND in the beginning. In line 6, we have an np-vertex v_i being vertex 15, and vertex 15 is not visited. Now, a new ECC ${\rm ECC}_1$ is ready to expand using the BFS (see lines 15–27). In lines 18 and 19, vertex 15 collects GND and VCC as the p-vertices linked to ${\rm ECC}_1$ using nc-edges and collects A using a c-edge. However, vertex 15 does not link to any np-vertex by nc-edges. Therefore, the first-run BFS finishes, resulting in the ${\rm ECC}_1$ listed in the first row of Table I. Now, the set of ECCs $S_{\rm ECC}$ has the first element ${\rm ECC}_1$ (line 28).

TABLE I
ECCs DECOMPOSED FROM THE CONSTRAINT GRAPH IN FIG. 6 USING THE ECCD ALGORITHM, ASSUMING THAT THE DECOMPOSITION ORDER IS $\langle \mathrm{ECC}_1, \mathrm{ECC}_2, \mathrm{ECC}_3, \mathrm{ECC}_4, \mathrm{ECC}_5 \rangle$

ESD Connected Component	Vertex	P-vertex Linked by NC-Edge	P-vertex Linked by C-Edge	Precedent Neighboring ESD Connected Component
ECC_1	15	GND, VCC	A	-
ECC_2	11, 20	GND	B	ECC_1
ECC_3	9	VCC3A	-	-
ECC_4	8	B	VCC3A	ECC_2, ECC_3
ECC_5	10	VCC	-	ECC_2

Back to line 6, v_i has another choice, i.e., vertex 20, expanding another ECC ECC_2. During the BFS of ECC_2, vertex 20 collects GND as the p-vertex linked to ECC_2 using an nc-edge and collects B using a c-edge. In addition, vertex 20 collects a precedent neighboring ECC ECC_1 by visiting vertex 15 (lines 20–22). On the other hand, vertex 20 also visits the np-vertex, i.e., vertex 11, trying to expand ECC_2 and to collect more p-vertices and more precedent neighboring ECCs. After the second BFS run finishes, the resulting ECC_2 is united to the set S_{ECC}. To this point, p-vertex GND cannot further expand.

Similar to the aforementioned steps, VCC3A, B, and VCC detect the ECCs ECC_3, ECC_4, and ECC_5, respectively. However, p-vertex A detects no ECC since all the np-vertices have been visited. Finally, the ECCD algorithm returns the set of the ECCs detected from the given constraint graph. In this example, $S_{\mathrm{ECC}} = \{\mathrm{ECC}_1, \mathrm{ECC}_2, \mathrm{ECC}_3, \mathrm{ECC}_4, \mathrm{ECC}_5\}$, and all the related information is listed in Table I.

Since all the np-vertices in the constraint graph $G = (V, E)$ are visited no more than once—an np-vertex surrounded by several levels of c-edges may never be visited—the ECCD algorithm performs all the needed BFSs in $O(|V| + |E|)$ time. Because a p-vertex p is added to the V_p of an ECC C only when p is linked to a *visited* np-vertex of C (see lines 18–19), it implies that, for each p-vertex, the number of times to add it to the V_ps of the ECCs is bounded by its degree. Thus, the ECCD collects p-vertices in $O(|E|)$ time for all V_ps during the BFSs. On the other hand, the number of collecting precedent neighboring ECCs is bounded by $O(|E|)$ during the BFSs since an ECC C_1 is added to the PN of ECC C_2 via c-edge (v_1, v_2) only when both v_1 and v_2 have been visited (see lines 20–22), where PN is the set of precedent neighboring ECCs. Thus, the ECCD collects precedent neighboring ECCs in $O(|E|)$ time for all visited np-vertices via c-edges. Overall, the time complexity of the ECCD algorithm is $O(|V| + |E|)$.

Note that the ECCD algorithm decomposes ECCs *linked with p-vertices only*, instead of all ECCs. This is why we check only the p-vertices in line 5 of the ECCD algorithm. To test the effect of the p-vertex identification on the efficiency of the ECCD algorithm, we also implemented a version of the ECCD algorithm without p-vertex identification (Algorithm ECCD-WPI), which decomposes all ECCs from a constraint graph, for comparative study (see Section V). A sample comparison between ECCD and ECCD-WPI is shown in Fig. 8, for which ECCD results in a much smaller problem size.

Theorem 1: The ESD paths detected by ECCD (which decomposes ECCs linked with p-vertices only) are identical to those detected by ECCD-WPI (which decomposes all ECCs).

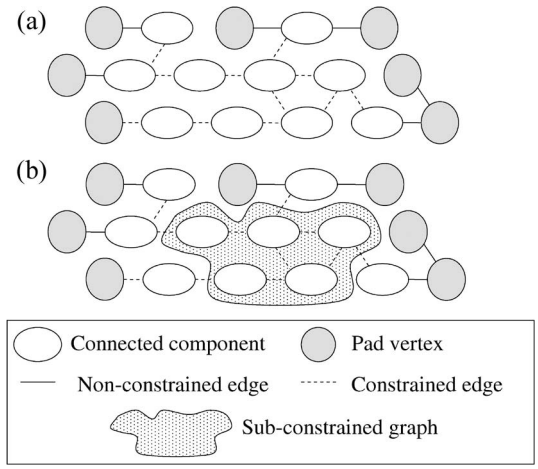


Fig. 8. Example after the connected component decomposition stage with (a) the ECCD-WPI algorithm and (b) the ECCD algorithm. The dotted region in (b) is a subconstraint graph where ECCD does not decompose the ECCs inside, whereas ECCD-WPI decomposes all the ECCs from a constraint graph.

Proof: For an ESD path between two pads, there are four possible cases of the two pads.

1) The two pads are linked to the same ECC via nc-edges, as shown in Fig. 9(a).
2) The two pads are linked to two different ECCs via nc-edges, and the two components are linked via a c-edge, as shown in Fig. 9(b).
3) One of the pads is linked to an ECC via an nc-edge, and the other pad is linked to the component via a c-edge, as shown in Fig. 9(c).
4) The two pads are directly linked to each other, as shown in Fig. 9(d).

For all connected components shown in Fig. 9, they are all linked with p-vertices. Hence, only the connected components linked with p-vertices need to be decomposed. For this reason, ECCD can indeed handle these four cases, and the ESD paths detected by ECCD are the same as those detected by ECCD-WPI. ■

3) Pad Matching: After the ECCD stage, four feasible connection types between two pads are constructed as those mentioned in the proof of Theorem 1. Among them, the type-4 ESD path is detected in the ECCD algorithm (lines 7 and 8). Therefore, in this stage, we do not need to match the pads of this type. In addition, the type-1 path has higher priority than those of all the other connection types. That is, a path P_1 involving no c-edge is given a higher priority than a path P_2 involving one c-edge; thus, P_1 can conduct more ESD paths in higher level circuit blocks than P_2.

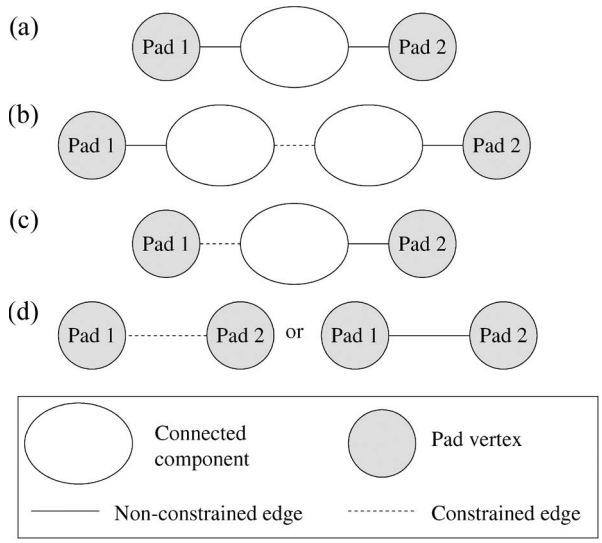


Fig. 9. Four feasible connection types of an ESD path.

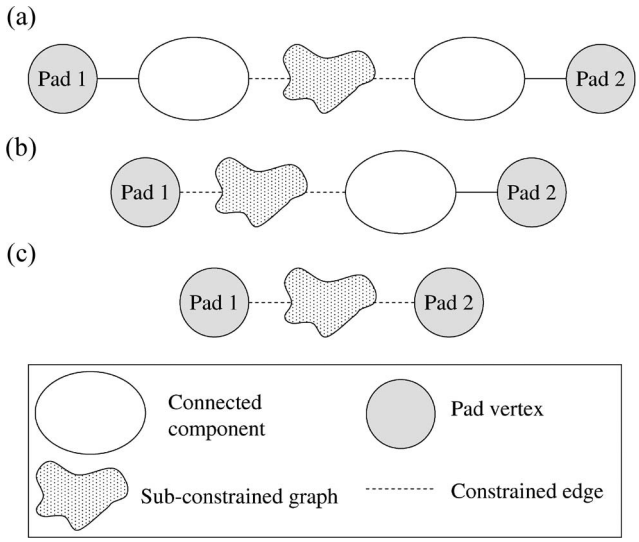


Fig. 10. Three connection types for the proof of Theorem 2.

Given the ECCs returned from the ECCD algorithm, we can match the pads according to the aforementioned feasible connection types. For example, in Fig. 6 and Table I, we have detected the ECCs using the ECCD algorithm. For each ECC, we match its p-vertices according to the type-1 connection, i.e., (GND, VCC), and the type-3 connection, i.e., (A, GND), (A, VCC), (GND, B), and (B, VCC3A). In addition, for each ECC and its precedent neighboring ECCs, we match their p-vertices according to the type-2 connection: (GND, VCC), (GND, B), and (B, VCC3A). There are also type-4 connections detected in the ECCD algorithm: (A, GND) and (A, VCC). Considering the path priority, the resulting matching is of type 1, i.e., (GND, VCC), type 2, i.e., (GND, B) and (B, VCC3A), and type 3, i.e., (A, GND) and (A, VCC).

Finally, these paths are saved in the hash table for the subcircuit references mentioned in Section III-B1. In the aforementioned example, (GND, VCC) is saved as the first case, whereas the others are saved as the second case.

Theorem 2: If two pads are not linked according to one of the four feasible types in the pad matching, there must be no ESD path between the pads.

Proof: If there is a path between two pads involving more than one nc-edge and no c-edge, the two pads must be connected to the same ECC, and the path is of the first feasible connection type in pad matching. Otherwise, there are only three remaining connection types in which the path must involve at least two c-edges. The remaining types are described here.

1) The two pads are linked to two different ECCs, and the two ECCs are linked by more than one c-edge, as shown in Fig. 10(a).
2) Only one pad is linked to an ECC, and the other pad is linked to the ECC by more than one c-edge, as shown in Fig. 10(b).
3) Neither of the two pads is linked to an ECC, and they are linked to each other by more than one c-edge, as shown in Fig. 10(c).

These types guarantee that the path between the two pads must involve at least two c-edges, violating the first constraint of the ESD path definition (see Definition 5). For this reason, there must be no ESD path between the pads. ■

IV. Extensions

In the preceding section, our algorithm handles the case where the ESD voltage burns out at most one transistor, which is the typical case for current applications. Considering the fast shrinking of CMOS devices (e.g., wires and oxide getting thinner), an ESD voltage could be large enough to destroy more than one MOS transistor in the future. Accordingly, we shall also explore this general ESD detection problem for which we have a different definition of an ESD path as follows.

Definition 9: An *N-ESD path* is a current path with two constraints.

1) An N-ESD path can propagate between gate–source or gate–drain at most N times.
2) An N-ESD path cannot pass through any other pads, except the two terminal pads of the path.

The extended problem is formulated as follows.

General ESDA (GESDA) problem: Given a netlist with circuit hierarchy and an integer N, find every pair of pads with an N-ESD path between them for circuit protection.

It should be noted that N is typically a very small constant in practice since the ESD voltage could not burn out an arbitrary number of MOS transistors. The GESDA problem can be solved by our primitive circuit processing in Section III-B with minor modifications; we give the modifications here.

A. Constraint Graph Construction for GESDA

Because an N-ESD path can propagate between gate–source or gate–drain at most N times, we assign weights to the edges of the constraint graph, instead of classifying the edges as c-edges and nc-edges. For resistors, diodes, and MOS transistors,

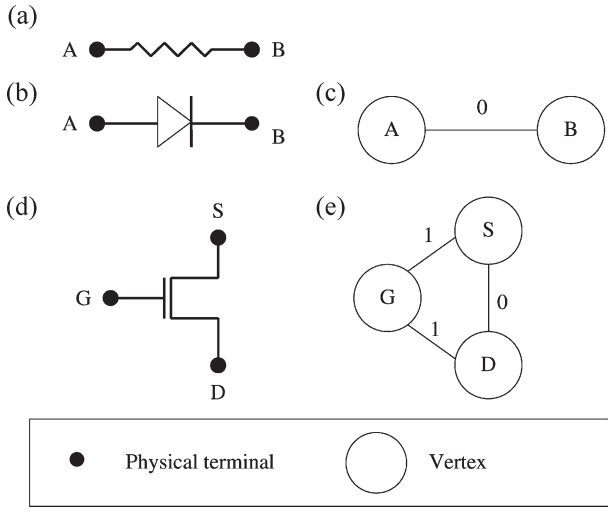


Fig. 11. (a) Resistor with its two terminals. (b) Diode with its two terminals. (c) Constraint graph model of a resistor/diode. (d) MOS transistor with its three terminals: gate, source, and drain (denoted by G, S, and D, respectively). (e) Constraint graph model of a MOS transistor.

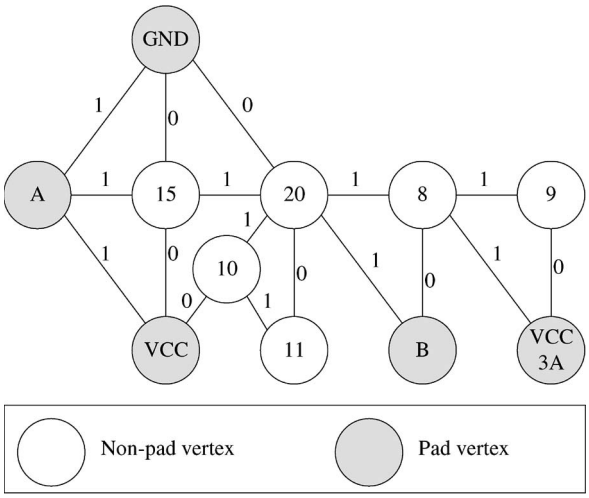


Fig. 12. Constraint graph constructed from the netlist in Fig. 2, using the construction methods presented in Section IV-A.

```
Algorithm: GECCD(C_p, H, G, G_r)
Input: C_p /* primitive circuit block ID */
       H /* global hash table */
       G = (V, E) /* constrained graph of C_p */
Output: G_r = (V_r, E_r)
       /* the reduced constraint graph: GECCD-G*/
/* representing an edge e as (v1,v2,length) */
1 V_r ← ∅
2 E_r ← ∅
3 for each np-vertex v_n ∈ V
4    v_n.GECC ←null
5    v_n.depth ← ∞
6 Q ← ∅
/* Q is a priority-queue with key being depth */
7 for each p-vertex v_p ∈ V
8    V_r ← V_r ∪ v_p
9    for each edge (v_p,v_i,l) in E
10      if v_i is a p-vertex and l ≤ N
11         H(C_p) ← H(C_p) ∪ {path(v_p, v_i, l)}
12      else if v_i is a np-vertex and l ≤ ⌈N/2⌉
13         v_i.depth ← MIN(v_i.depth, l)
14         Q ← Q ∪ {v_i}
15 while Q is nonempty
16   v_n ← EXTRACT-MIN(Q)
17   if v_n.GECC is not null
18      jump to line 15
19   C ← {v_n} /* use to decompose a GECC*/
20   create an np-vertex v_g /* represent a GECC */
21   V_r ← V_r ∪ v_g
22   v_g.depth ← v_n.depth
23   while C is nonempty
24      v_n ← Extract(C)
25      v_n.GECC ← v_g
26      for each edge (v_n,v_i,l) in E
27         if v_i is a p-vertex
28            if l + v_g.depth ≤ N
29               E_r ← E_r ∪ (v_g, v_i, l)
30         else if v_i.GECC is null
31            if l = 0
32               C ← C ∪ v_i
33            else if l + v_g.depth ≤ ⌈N/2⌉
34               v_i.depth ← MIN(v_i.depth, l + v_g.depth)
35               Q ← Q ∪ v_i
36         else if v_g ≠ v_i.GECC
37            v'_g ← v_i.GECC
38            if v'_g.depth + l + v_g.depth ≤ N
39               E_r ← E_r ∪ (v_g, v'_g, l)
40 Return G_r
```

Fig. 13. GECCD algorithm.

we model the corresponding c-edges and nc-edges of these vertices as the new edges with weights being 1 and 0, respectively. See Fig. 11 for an example. In addition, the constraint graph construction for a primitive circuit should be modified.

1) If p_1 and p_2 are connected by an N-ESD path involving no c-edge, add an edge between v_1 and v_2, and assign weight zero to this edge.
2) If p_1 and p_2 are connected by an N-ESD path involving M c-edges, where $M \leq N$, add an edge between v_1 and v_2, and assign weight M to this edge.
3) If p_1 and p_2 are not connected by any N-ESD path, do nothing.

Fig. 12 shows the new constraint graph for the netlist shown in Fig. 2. By the weight assignment, we immediately have the following property.

Property 2: In the constraint graph constructed from a primitive circuit, an N-ESD path is represented as a path with its weight being at most N.

Therefore, the GESDA problem is finding every pair of p-vertices with its path weight at most N in the new constraint graph.

B. General ECCD

In this section, we propose a general ECCD (GECCD) algorithm (see Fig. 13), which decomposes general ECCs (GECCs) from a constraint graph to construct a reduced constraint graph (denoted as GECCD-G). The GECCD algorithm only decomposes the GECC with a path leading to a p-vertex with weight being at most $\lceil N/2 \rceil$. As a result, the solution space can substantially be reduced. We first modify Definition 7 as Definition 10.

Definition 10: Let $G = (V, E)$ be a constraint graph described in Section IV-A, where V and E are the vertex set and the edge set, respectively. Let $V_{\text{np}} \subseteq V$ be the np-vertex set. A *GECC* is a maximal vertex set $C \subseteq V_{\text{np}}$, where $\forall v_i, v_j \in C$,

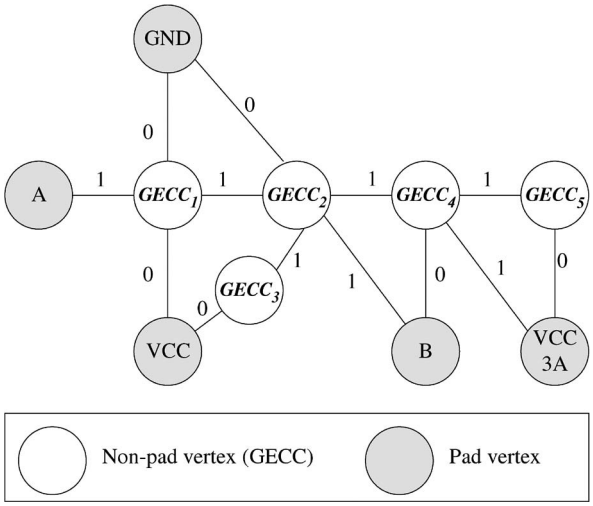


Fig. 14. Reduced constraint graph GECCD-G constructed from Fig. 12 by the GECCD algorithm.

$i \neq j$, and there is a path between v_i and v_j with weight being zero.

By Definition 10, a GECC contains only np-vertices, and for each np-vertex in the GECC, there is always a path with zero weight leading to every other np-vertices in the GECC. Hence, according to Property 2, a GECC can be considered as a vertex when finding N-ESD paths to reduce the solution space. Furthermore, it will be clear in Theorem 3 that it is sufficient to only decompose the GECC with a path leading to a p-vertex with weight being at most $\lceil N/2 \rceil$. As a result, the reduced constraint graph GECCD-G constructed by the GECCD algorithm helps to solve the GESDA problem.

The GECCD algorithm mainly applies a BFS, and some details are described here.

1) Lines 7–14 initialize the BFS and generate trivial N-ESD paths.
2) Lines 15–39 process the BFS to decompose the GECC with a path leading to a p-vertex with weight being at most $\lceil N/2 \rceil$.
3) Lines 27–29 and lines 36–39 generate the edges of GECCD-G.
4) Lines 31 and 32 include vertices of this GECC.
5) Lines 33–35 add vertices in subsequent steps of the BFS.

Using the constraint graph in Fig. 12 as an example, we assume $N = 2$ (hence, $\lceil N/2 \rceil = 1$) and explain the GECCD algorithm described in Fig. 13. After the initialization steps in lines 1–6 of the GECCD algorithm, we assume that the p-vertex enumeration order in line 7 is $\langle \text{GND}, \text{VCC3A}, \text{B}, \text{VCC}, \text{A} \rangle$. In the iteration (lines 8–14) for GND, $H(C_p)$ contains a trivial N-ESD path (GND, A, 1) in line 11, and Q includes vertices 15 and 20 with zero depths in lines 13–14. Similarly, after the iterations (lines 8–14) for VCC3A, B, VCC, and A, $H(C_p) = \{(\text{GND}, \text{A}, 1), (\text{VCC}, \text{A}, 1)\}$, and $Q = \{(15, 0), (20, 0), (10, 0), (8, 0), (9, 0)\}$. After that, we begin the BFS in lines 15–39 by extracting vertex 15 from Q (line 16). First, an np-vertex is constructed (line 20) and denoted as GECC_1 shown in Fig. 14. Since there is no np-vertex linked to vertex 15 with a zero-weight edge in Fig. 12, GECC_1 only

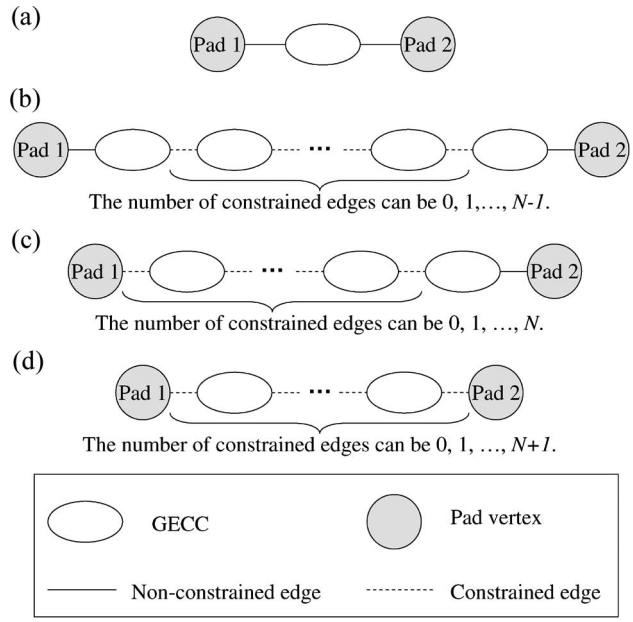


Fig. 15. All cases of pad matching for the GESDA problem.

Algorithm: $GESDA - PM(C_p, H, G_r)$
Input: C_p /* The primitive circuit block ID */
H /* The global hash table */
$G_r = (V_r, E_r)$
/* GECCD-G of C_p */
Output: H /* The global hash table */
1 Clear DA
/* DA is an array which has $N+1$ elements. */
/* Each element of DA is a double-link-list. */
2 **for** each p-vertex $v_p \in V_r$
3 Clear DA
4 **for** each vertex $v_i \in V_r - v_p$
5 $W[v_i] \leftarrow \infty$
6 **for** each edge $(v_p, v_n, l) \in E_r$
7 **if** $l \leq W[v_n]$ /* $l \leq N$ due to GECCD */
8 $W[v_n] \leftarrow l$
9 move v_n to $DA[l]$
10 **for** i = 0 to N
11 **for** each np-vertex $v_n \in DA[i]$
12 **for** each edge $(v_n, v_k, l) \in E_r$
13 **if** v_k is a p-vertex
14 $W[v_k] \leftarrow MIN(W[v_k], i + l)$
15 **else if** $i + l < W[v_k]$ and $i + l \leq N$
16 $W[v_k] \leftarrow i + l$
17 move v_k to $DA[i + l]$
18 **for** each p-vertex $v_i \in V_r - v_p$
19 **if** $W[v_i] \leq N$
20 $H(C_p) \leftarrow H(C_p) \cup path(v_p, v_i, W[v_i])$
21 Return H

Fig. 16. GESDA-PM algorithm.

contains vertex 15 (line 32 is not executed in this iteration). In addition, E_r contains $(\text{GECC}_1, \text{GND}, 0)$, $(\text{GECC}_1, \text{VCC}, 0)$, and $(\text{GECC}_1, A, 1)$ (line 29) at this iteration, as shown in Fig. 14. Then, we extract vertex 20 from Q (line 16) and begin the next iteration of the BFS. Since vertex 11 is linked to vertex 20 with a zero-weight edge, vertex 11 is also included to decompose GECC_2 (line 32). Furthermore, since $\text{GECC}_1.depth = 0$, $\text{GECC}_2.depth = 0$, and there is an edge connecting vertices 15 and 20 with weight being 1, E_r includes $(\text{GECC}_2, \text{GECC}_1, 1)$ (line 39), as shown in Fig. 14. In addition,

TABLE II
PARAMETERS OF THE 11 INDUSTRIAL CIRCUITS. THE NUMBER OF VERTICES REPRESENTS THE TOTAL VERTICES OF THE FLATTENED CIRCUIT IF WE EXPAND ALL SUBCIRCUITS IN THE NETLIST

Circuit	Number of						ESD Path % (A/B)
	Pads	Circuit Blocks	Vertices	References of Circuit Blocks	ESD Paths (A)	All Paths (B)	
Industry 1	17	1	109	1	48	136	35.29
Industry 2	261	73	4060	917	601	33930	1.77
Industry 3	542	317	17129	43424	1234	146611	0.84
Industry 4	15	436	56267	56635	39	105	37.14
Industry 5	26	604	63026	59675	65	325	20.00
Industry 6	28	263	102603	829859	53	378	14.02
Industry 7	28	295	156306	797381	53	378	14.02
Industry 8	69	840	424202	1675863	95	2346	4.05
Industry 9	69	1411	735271	3394388	95	2346	4.05
Industry 10	69	2193	1016038	5036407	102	2346	4.35
Industry 11	69	2601	1339677	5176763	136	2346	5.63

E_r also includes $(\text{GECC}_2, \text{GND}, 0)$ and $(\text{GECC}_2, \text{B}, 1)$ (line 29) in this iteration of the BFS. By repeating the preceding process until Q is empty, the reduced constraint graph GECCD-G is constructed, as shown in Fig. 14.

It is trivial that an N-ESD path either (1) directly connects two pad vertices or (2) goes through at least one GECC. Since the first case must be found by lines 7–14 of the GECCD algorithm, we just consider the second case here.

Theorem 3: If an N-ESD path goes through a GECC C, C must be decomposed by the GECCD algorithm.

Proof: We prove by contradiction, i.e., we assume that C cannot be decomposed by the GECCD algorithm. Without loss of generality, we assume that the N-ESD path is (p_1, p_2). Since the GECCD algorithm must decompose every GECC with a path leading to a p-vertex whose weight is at most $\lceil N/2 \rceil$, every path connecting C to a p-vertex must have a weight larger than $\lceil N/2 \rceil$. Hence, both the distances of (p_1, C) and (p_2, C) are larger than $\lceil N/2 \rceil$; this implies that the distance of (p_1, p_2) is larger than N, contradicting that (p_1, p_2) is an N-ESD path. Hence, if an N-ESD path goes through a GECC C, C must be decomposed by the GECCD algorithm. ■

By Theorem 3 and the GECCD algorithm, the N-ESD paths of the second case must be maintained in GECCD-G. As a result, the solution space can substantially be reduced to improve the efficiency.

C. Pad Matching for GESDA

As shown in Fig. 15, for an N-ESD path, there can be at most N c-edges in the path between two pads. There are more cases for pad matching, compared with the original ECCD. Thus, the method described in Section III-B3 is not sufficient for GECCD. We thus present a new method here.

As mentioned in Section IV-B, the N-ESD paths of the first case have been found by the GECCD algorithm, and the N-ESD paths of the second case are maintained in GECCD-G. Since there is no negative edge, we can apply Johnson's algorithm on GECCD-G to find all the shortest paths between all p-vertices. For each p-vertices pair, if the distance of a shortest path between the two p-vertices is smaller than or equal to $|N|$, there exists an N-ESD path between them. However, it is well known that operations of a Fibonacci heap are very time consuming; hence, Johnson's algorithm may be inefficient

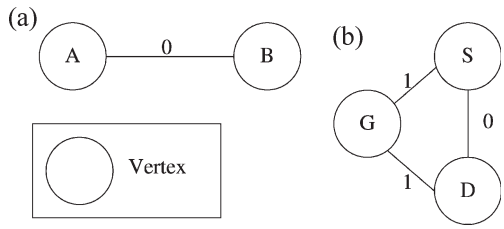


Fig. 17. Graph connections of a resistor, a diode, and a MOS transistor in the Johnson algorithm. (a) Graph representing a resistor or a diode. (b) Graph representing a MOS transistor. (The gate, source, and drain are denoted by G, S and D, respectively.)

in practice. Fortunately, since a path with weight more than N must not be an N-ESD path and N is a very small constant in practice, we can replace a Fibonacci heap with an array whose length is $N + 1$ to record np-vertices. With such an array, we develop the GESDA path-matching (GESDA-PM) algorithm to find N-ESD paths of the second case, as summarized in Fig. 16.

Some details about the GESDA-PM algorithm are described here.

1) Lines 3–17 compute the shortest paths from v_p leading to all other p-vertices.
2) Lines 6–9 initialize DA.
3) Lines 10–17 sequentially propagate vertices in DA.
4) By Definition 9, an N-ESD path cannot pass through any other pads, except the two terminal pads of the path. Hence, lines 13 and 14 do not put v_k to DA, but lines 15–17 do.
5) Lines 18 and 19 include N-ESD paths in $H(C_p)$.

V. EXPERIMENTAL RESULT

We implemented our algorithms ECCD and ECCD-WPI in the C/C++ language on a 3.0-GHz Intel Pentium 4 PC with 2-GB memory under Linux 2.6 operating system. There are 11 industrial circuits from a leading design service company, as shown in Table II, where the number of vertices represents the total vertices of the flattened circuit if we expand all subcircuits in the netlist. As shown in the last three columns, the number of ESD paths that are needed to be protected may just be a small portion of all pad pairs; the percentage of ESD paths ranges from 0.8% to 37%. The data reveal that we can save significant

TABLE III
COMPARISON OF THE THREE ALGORITHMS FOR THE ESDA PROBLEM. "DECC" IS THE ABBREVIATION OF "DECOMPOSED ECCS"

Circuit	Johnson		ECCD-WPI			ECCD		
	CPU Time (s)	Memory (MB)	CPU Time (s)	Memory (MB)	# of DECC	CPU Time (s)	Memory (MB)	# of DECC
Industry 1	0.00	2	0.00	< 1	68	0.00	< 1	58
Industry 2	0.39	4	1.52	1	3164	0.01	1	916
Industry 3	2.10	10	2.65	1	9178	0.02	1	2368
Industry 4	15.03	27	3.12	1	28894	0.03	1	11528
Industry 5	17.17	27	3.62	1	33446	0.03	1	13291
Industry 6	15.69	85	42.16	47	15042	0.60	26	1601
Industry 7	13.37	80	42.75	46	15175	0.58	26	1633
Industry 8	31.06	152	53.70	53	42547	0.74	26	6095
Industry 9	61.68	268	66.21	66	76176	1.03	25	10189
Industry 10	125.52	384	80.32	78	130432	1.28	28	24461
Industry 11	415.48	415	84.19	80	221389	1.39	44	68515

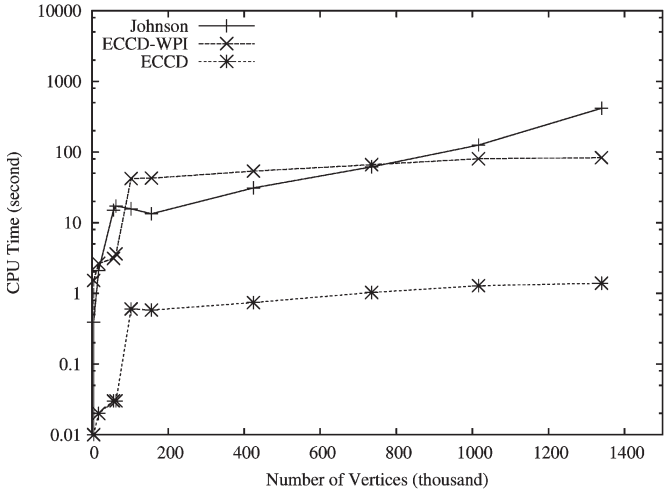


Fig. 18. Number of vertices versus CPU time of the three algorithms.

circuit overheads by identifying the potential ESD paths and protecting only those paths.

We compare our algorithms with one based on the Johnson's all-pair shortest-path algorithm [12]. The comparative algorithm, which is called Johnson, transfers the circuit to a weighted undirected graph by two rules.

1) A resistor or a diode is transferred into an edge with a weight of 0, as shown in Fig. 17(a).
2) A MOS transistor is transferred to three edges, as shown in Fig. 17(b). The source–drain edge has a weight of 0; the gate–source or gate–drain edge has a weight of 1.

Based on this model, the ESD path detection problem is reduced to a graph-search problem to examine if there is a path with its total weight being 0 or 1 between two pads. This can be done with the all-pair shortest-path algorithms. Since the constraint graph is typically a sparse graph, we select Johnson's all-pair shortest-path algorithm, instead of the Floyd–Warshall one. Furthermore, since the shortest paths between np-vertices must not be ESD paths, we only perform Dijkstra's shortest-path algorithm on p-vertices in Johnson's algorithm to make a fair comparison.

The three algorithms (ECCD, ECCD-WPI, and Johnson) detect the same ESD paths for all test circuits. Table III shows the CPU time and memory usage of the three algorithms. In Fig. 18, the CPU time (*in logarithmic scale*) is plotted as a function of the number of vertices for each of the three algorithms. As shown in Table III and Fig. 18, ECCD is about 50 times faster than ECCD-WPI, and ECCD-WPI is significantly faster than Johnson (the larger the circuit, the bigger the runtime difference). For example, when the number of vertices in the circuit reaches 1 339 677 (Industry 11), ECCD still completes the computation in 1.39 s, whereas ECCD-WPI requires more than 84 s and Johnson needs more than 415 s. In addition, ECCD is much more economical in memory usage than ECCD-WPI and Johnson.

It is also shown in Table III that the number of DECCs of ECCD is much smaller than that of ECCD-WPI. Taking the largest circuit Industry 11, for example, ECCD decomposes 68 515 ECCs, whereas ECCD-WPI decomposes 221 389 ones. Therefore, our ECCD algorithm is very efficient and economical. This difference in the number of DECCs also indicates that the connected relations between vertices are often sparse in a real circuit. It also implies that the ESD protection for all pads is really extravagant, and the ESD path detection in this paper can indeed reduce the design cost.

We also implemented our extension algorithm to solve the GESDA problem and called it GECCD. Since we typically only need to consider 1-ESD paths for practical applications, we shall only compute 2-ESD paths of those circuits here. Table IV shows the percentage of reduced vertices, the percentage of 2-ESD path, and the CPU times of the two algorithms Johnson and GECCD. As shown in Table IV, GECCD is much faster than Johnson. For instance, when the number of vertices in the circuit reaches 1.3 million (Industry 11), GECCD still completes the computation in 2.89 s, whereas Johnson requires more than 535 s. Furthermore, when an external voltage burns out more than one MOS transistor, the CMOS devices should shrink to a lower level. At that time, the design density of a circuit would be much larger, and the percentages of the reduced vertices and 2-ESD paths would drop. Hence, the runtime improvement of GECCD can significantly be increased. It also reveals that the N-ESD path detection in this paper could reduce the design cost.

VI. CONCLUSION

We have proposed *the first* ESD detection algorithm to detect all pads in danger of an ESD violation at the design stage. Experimental results have shown that our algorithm can very efficiently and economically detect all ESD paths. Our work

TABLE IV
COMPARISON OF THE TWO ALGORITHMS FOR THE GESDA PROBLEM

Circuit	Number of Paths		2-ESD Path	Number of Vertex		Reduced Vertices	CPU Time (sec)	
	2-ESD (A)	All (B)	(A/B) %	GECCD-G (C)	Constraint Graph (D)	(C/D) %	Johnson	GECCD
Industry 1	60	136	44.12	74	109	67.89	0.00	0.00
Industry 2	804	33930	2.37	1583	4060	38.99	0.45	0.01
Industry 3	1648	146611	1.12	5633	17129	32.89	2.65	0.03
Industry 4	45	105	42.86	19276	56267	34.26	19.61	0.13
Industry 5	70	325	21.54	22545	63026	35.77	21.60	0.14
Industry 6	53	378	14.02	14905	102603	14.53	15.85	0.65
Industry 7	53	378	14.02	15038	156306	9.62	13.50	0.64
Industry 8	111	2346	4.73	41762	424202	9.84	31.99	0.92
Industry 9	111	2346	4.73	75147	735271	10.22	63.21	1.35
Industry 10	118	2346	5.03	118664	1016038	11.68	132.70	1.86
Industry 11	152	2346	6.48	174041	1339677	12.99	535.35	2.89

shows that the ESD protection for all pads is really extravagant, and our ESD path detection algorithm can indeed reduce the design cost.

ACKNOWLEDGMENT

The authors would like to thank C.-Y. Peng and Y.-W. Lin of Faraday Technology, Inc., for their valuable help and comments on this paper.

REFERENCES

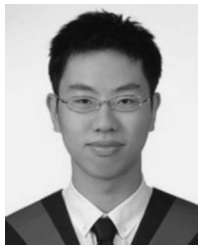
[1] H.-Y. Liu, C.-W. Lin, S.-J. Chou, W.-T. Tu, Y.-W. Chang, and S.-Y. Kuo, "Current path analysis for electrostatic discharge protection," in *Proc. ICCAD*, 2006, pp. 510–515.
[2] A. Amerasekera and C. Duvvury, *ESD in Silicon Integrated Circuits*, 2nd ed. New York: Wiley, 2002.
[3] B. P. Wong, A. Mittal, Y. Cao, and G. Starr, *Nano-CMOS Circuit and Physical Design*. New York: Wiley, 2005.
[4] *ESD Association Standard*, ANSI/ESD S20.20, 1999.
[5] IEC 61340-5-1 Ed. 1.0 b, 2007.
[6] A. Wang, *On-Chip ESD Protection for Integrated Circuits*. Norwell, MA: Kluwer, 2002.
[7] S. Hayashi, F. Minami, and M. Yamada, "Full-chip analysis method of ESD protection network," in *Proc. ISQED*, 2004, pp. 439–444.
[8] H. Qian, J. N. Kozhaya, S. R. Nassif, and S. S. Sapatnekar, "A chip-level electrostatic discharge simulation strategy," in *Proc. ICCAD*, 2004, pp. 315–318.
[9] R. Zhan, H. Feng, Q. Wu, H. Xie, X. Guan, G. Chen, and A. Wang, "ESDInspector: A new layout-level ESD protection circuitry design verification tool using a smart-parametric checking mechanism," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 23, no. 10, pp. 1421–1428, Oct. 2004.
[10] R. Zhan, H. Xie, H. Feng, and A. Wang, "ESDZapper: A new layout-level verification tool for finding critical discharging path under ESD stress," in *Proc. ASPDAC*, 2005, pp. 79–82.
[11] E. Nuutila, "Efficient transitive closure computation in large digraphs," Ph.D. dissertation, Helsinki Univ. Technol., Helsinki, Finland, 1995. Mathematics and Computing in Engineering Series, no. 74.
[12] T. Cormen, C. Leiserson, R. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed. Cambridge, MA: MIT Press, 2001.

Chih-Hung Liu received the B.S. degree in computer science and information engineering from National Taiwan University, Taipei, Taiwan, R.O.C., in 2005. He is currently working toward the Ph.D. degree in the Graduate Institute of Electronics Engineering, National Taiwan University.

His research interests include graph algorithm, computation geometry, combinatorial optimization, and their applications to EDA, with emphasis on physical design. He is currently studying the Steiner minimal tree problem in IC routing and has a related paper published in ISPD 2008.

Hung-Yi Liu received the B.S. degree in computer science and information engineering from National Chiao Tung University, Hsinchu, Taiwan, R.O.C., in 2005 and the M.S. degree in electronics engineering from National Taiwan University, Taipei, Taiwan, in 2007.

He is currently serving his military duty in the R.O.C. army as a Second Lieutenant. He has coauthored four papers published in DAC and ICCAD since 2006. His research interests include algorithm design/analysis, combinatorial optimization, and their application to EDA, with emphasis on low-power design.



Chung-Wei Lin received the B.S. degree in computer science and information engineering and the M.S. degree in electronics engineering from National Taiwan University, Taipei, Taiwan, R.O.C., in 2005 and 2007, respectively.

He is currently serving in the military. His research interests include routing-related topics and design for manufacturability/reliability.

Mr. Lin is an Honorary Member of the Phi Tau Phi Scholastic Honor Society (R.O.C.). He was the recipient of the Presidential Award from National Taiwan University for six semesters during his college years.

Szu-Jui Chou received the B.S. degree in electrical engineering and the M.S. degree in electronics engineering from National Taiwan University, Taipei, Taiwan, R.O.C., in 2005 and 2007, respectively.

She is currently an R&D Engineer with the Implementation Group, Synopsys Taiwan Ltd., Taipei. Her research interests include CMP-aware routing and dummy metal filling.

Yao-Wen Chang (S'94–A'96–M'96) received the B.S. degree from National Taiwan University, Taipei, Taiwan, R.O.C., in 1988 and the M.S. and Ph.D. degrees from the University of Texas, Austin, in 1993 and 1996, respectively, all in computer science.

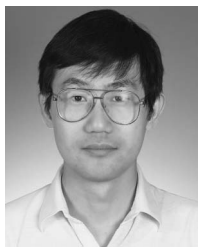
He is currently a Professor with the Department of Electrical Engineering and the Graduate Institute of Electronics Engineering, National Taiwan University. He is also currently a Visiting Professor with Waseda University, Kitakyushu, Japan. He was with the IBM T. J. Watson Research Center, Yorktown Heights, NY, in the summer of 1994, and the faculty of National Chiao Tung University, Hsinchu, Taiwan, from 1996 to 2001. His current research interests include VLSI physical design, design for manufacturability/reliability, and design automation for biochips. He has coauthored one book on routing and more than 130 ACM/IEEE conference proceeding/journal papers and has been working closely with the industry on projects in these areas.

Dr. Chang is a member of the IEEE Circuits and Systems Society, ACM, and ACM/SIGDA. He is an Editor for the *Journal of Information Science and Engineering*. He is currently an Associate Editor for the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS. He currently serves on the ICCAD Executive Committee, ACM/SIGDA Physical Design Technical Committee, and the organizing committees of ISPD and FPT. He has served on the technical program committees of ASP-DAC (Topic Chair), DAC, DATE, FPT (Program Co-Chair), GLSVLSI, ICCAD, ICCD, IECON (Topic Chair), ISPD, SOCC (Topic Chair), TENCON, VLSI-DAT (Topic Co-Chair), etc. He is currently an independent board director of Genesys Logic, Inc. and a member of the board of governors of Taiwan IC Design Society. He was a winner of the 2006 ACM ISPD Placement Contest and the 2008 ACM ISPD Global Routing Contest; Best Paper Awards at ICCD-95 and the 2007 VLSI Design/CAD Symposium; and 11 Best Paper Award Nominations from DAC (four times), ICCAD (twice), ISPD (twice), ACM TODAES, ASP-DAC, and ICCD. He has also been the recipient of many awards for research performance, such as the 2007 Distinguished Research Award, the inaugural 2005 First-Class Principal Investigator Award, the 2004 Dr. Wu Ta You Memorial Award from the National Science Council of Taiwan, and the 2004 MXIC Young Chair Professorship from the MXIC Corp.; and for excellent teaching from National Taiwan University (four times) and National Chiao Tung University.

Sy-Yen Kuo received the B.S. degree in electrical engineering from National Taiwan University, Taipei, Taiwan, R.O.C., in 1979, the M.S. degree in electrical and computer engineering from the University of California, Santa Barbara, in 1982, and the Ph.D. degree in computer science from the University of Illinois, Urbana, in 1987.

He is currently a Chair Professor and the Dean of the College of Electrical and Computer Engineering, National Taiwan University of Science and Technology, Taipei. He is also a Distinguished Professor with the Department of Electrical Engineering, National Taiwan University, where he is currently taking a leave of absence and was the Chairman at the same department from 2001 to 2004. He spent his sabbatical years as a Visiting Professor with Department of the Computer Science and Engineering, Chinese University of Hong Kong, from 2004 to 2005, and as a Visiting Researcher with AT&T Labs-Research, Florham Park, NJ, from 1999 to 2000. He was the Chairman of the Department of Computer Science and Information Engineering, National Dong Hwa University, Hualien, Taiwan, from 1995 to 1998; a faculty member with the Department of Electrical and Computer Engineering, University of Arizona, Tucson, from 1988 to 1991; and an Engineer with Fairchild Semiconductor, San Jose, CA, and Silvar-Lisco, Palo Alto, CA, from 1982 to 1984. In 1989, he was a Summer Faculty Fellow with the Jet Propulsion Laboratory, California Institute of Technology, Pasadena. He has published more than 290 papers in journals and conference proceedings. He is the holder of several patents. His current research interests include dependable systems and networks, software reliability engineering, mobile computing, and reliable sensor networks.

Dr. Kuo is a Research Fellow of the National Science Council in Taiwan. He was the recipient of the Distinguished Research Award (1997–2005) from the National Science Council in Taiwan, the Best Paper Award at the 1996 International Symposium on Software Reliability Engineering, the Best Paper Award in the simulation and test category at the 1986 IEEE/ACM Design Automation Conference, the National Science Foundation's Research Initiation Award in 1989, and the IEEE/ACM Design Automation Scholarship in 1990 and 1991.



Shih-Yi Yuan received the M.S. and Ph.D. degrees in electrical engineering from National Taiwan University, Taipei, Taiwan, R.O.C., in 1993 and 1997, respectively.

In 2003, he joined the Department of Communications Engineering, Feng Chia University, Taichung, Taiwan, as an Assistant Professor. He is also a member of the ICEMC Center, Feng Chia University. His research interests include driver design, ICEMC model building, and CAD tool design for EMC estimations.

Yu-Wei Chen received the B.S. degree in computer and information science from National Chiao Tung University, Hsinchu, Taiwan, R.O.C., and the Ph.D. degree in computer science from the University of Oklahoma, Norman.

Since 2001, he has been with the Design and Development Department, Faraday Technology Corp., Hsinchu, leading a team that researches and develops in-house EDA tools and flows. Prior to joining Faraday, he was a Member of Technical Staff with Cadence, where he worked on various physical verification tools. His current research interests include test challenges of circuit level and physical design in nanometer technologies, automation of design and implementation for multiple power domain IC designs, and quality in electronic design.