Current Path Analysis for Electrostatic Discharge Protection*

Hung-Yi Liu†, Chung-Wei Lin†, Szu-Jui Chou†, Wei-Ting Tu†, Chih-Hung Liu†, Yao-Wen Chang†‡, and Sy-Yen Kuo†‡
Graduate Institute of Electronics Engineering, National Taiwan University, Taipei, Taiwan†
Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan‡

ABSTRACT

The electrostatic discharge (ESD) problem has become a challenging reliability issue in nanometer circuit design. High voltages 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 arises 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 constrained graph, decompose ESD connected components linked with the pads, and apply the breadth-first search (BFS) to identify the ESD connected components in each constrained graph and thus the current paths. Experimental results show that our algorithm can detect all ESD paths very efficiently and economically. To our best knowledge, our algorithm is the *first point tool* available to the public for the ESD analysis.

1. 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 human, it can cause severe 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 0.13um CMOS device designed for operation at 1.2V, the voltage drop across a 2Ω power bus exceeds 20V and burns out the ultra-thin gate oxides [1]. As the process technology enters the nanometer era, devices size has continued to shrink and the breakdown voltage of the thin-oxide devices is usually less than 5V, making the ESD damage occur easily and difficult to prevent [4]. As a result, the prevention of ESD becomes one of the major concerns for IC reliability.

*This work was partially supported by Incentia Design Systems, Inc. and NSC of Taiwan under Grant No's. NSC 94-2215-E-002-005, NSC 94-2215-E-002-030, and NSC 94-2752-E-002-008-PAE.

ICCAD'06 November 5–9, 2006, San Jose, CA

There are many solutions to the ESD problem [1], for example, by using antistatic coatings to prevent static charge generation in wafers, using shielded materials to prevent ESD resulted from human handling, and implementing protection circuits within the chip. For protection circuits, the purpose is to find the discharging path and protect the I/O pads of a chip. By applying a low-impedance path to drain the ESD current safely, we can prevent the relatively high ESD current from destroying the devices. If ESD occurs between any pair of I/O pads in a chip, we will need to protect all pairs of pads, requiring $n(n-1)/2$ protection circuits for an n-pad chip. However, there might be some pairs of pads without current paths between them, for which we need not to protect [1]. Therefore, it is desirable to detect the current paths and protect only the pads with current paths to save the cost for circuit design.

As a relatively new design challenge, there is not much EDA work on ESD protection in the literature (although there are numerous publications on the circuit design for ESD protection [1, 4]). Very recently, Zhan et al. showed how to verify the protection circuit design considering parasitic and devices at the post-layout stage [5, 6].

In this paper, we first introduce the current path analysis problem for ESD protection in circuit design. To detect those pairs of pads with current paths between them, we model the circuit as a constrained graph, decompose ESD connected components linked with the pads, reduce the graph, and apply the breadth-first search (BFS) to identify the ESD connected components in each constrained graph and thus the current paths. Experimental results show that our algorithm can detect all ESD paths very efficiently and economically. For example, our algorithm can detect all current paths in a circuit with more than 1.3 million vertices in 2.66 seconds and consume only 44 MB memory on a 1.6 GHz Intel Pentium 4 PC with 2 GB RAM. In contrast, the well-known Floyd-Warshall all-pairs shortest paths algorithm needs more than 2,007 seconds and 1,446 MB memory. It should be noted that to our best knowledge, our algorithm is *the first point tool* available to the public for the ESD analysis.

The rest of this paper is organized as follows. Section 2 formulates the ESD current path analysis problem. Section 3 presents the algorithm to identify all ESD current paths between pads. Section 4 reports the experimental results. Finally, we conclude this paper in Section 5.

2. PROBLEM FORMULATION

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

DEFINITION 1. *A* circuit block *is a circuit or a sub-circuit 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.*

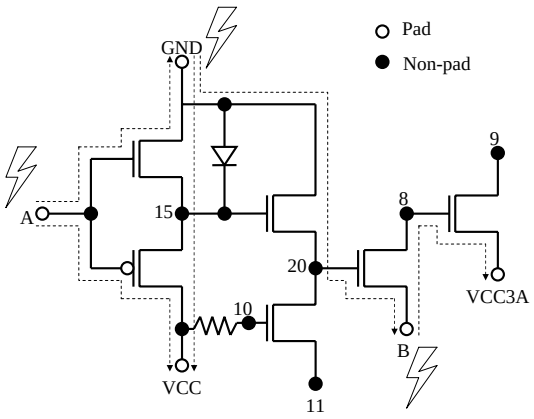

Figure 1: An example netlist and its ESD paths denoted by the dotted lines.

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. Further, our modeling is general and even applicable to non-MOS circuits, such as BJTs which can be similarly modeled as those for MOS's. (See Section 3.2.1 for more detail.)

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

Our objective is to detect all pairs of pads between which there is a current path. If a current path between two pads is detected, a back-to-back diode can be inserted between the two pads to protect the path. A current path can include any number of resistors and diodes and propagate between source and drain, from gate to source, and from gate to drain in a MOS transistor. Considering practical circuit operation, we further define an *ESD path* as follows:

DEFINITION 3. *An* ESD path *is a current path with the following two constraints:*

1. *An ESD path can propagate between gate-source or gate-drain at most once.* Practically, if an ESD voltage occurs, even a small voltage, the weakest current paths between two pads would be burned out and the chip would fail. Therefore, this constraint tries to detect the weakest current paths, i.e., the paths which propagate between gate-source or gate-drain at most once.
2. *An ESD path cannot pass through any other pads except the two terminal pads of this ESD path.* For instance, assuming that there is a pad, p_2, in the path from p_1 to p_3, the protection for the path from p_1 to p_2 and from p_2 to p_3 will ensure that the path from p_1 to p_3 is also protected. Hence, a current path containing any pad between two terminal pads is not regarded as an ESD 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 definition above, we can formulate our problem as follows:

PROBLEM 1. **ESD Analysis (ESDA):** *Given a netlist with circuit hierarchy, find every pair of pads with an ESD path between them for ESD circuit protection.*

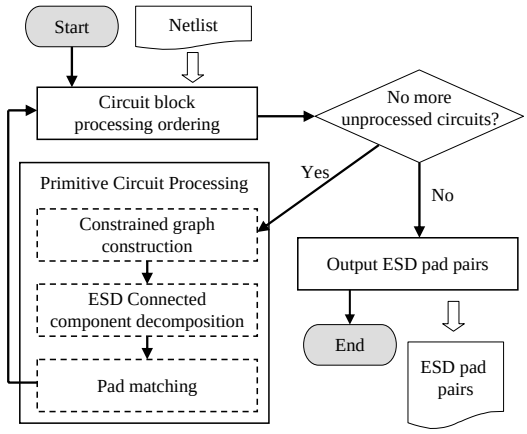

Figure 2: The flowchart of the proposed approach.

For example, as shown in Figure 1, 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 totally five pairs of pads between which there is an ESD path: (A, VCC), (A, GND), (B, VCC3A), (B, GND), and (VCC, GND).

3. ALGORITHM

As shown in Figure 2, 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 sub-circuit block once it is referenced. Therefore, we shall apply a bottom-up circuit block processing ordering to speed up the processing (see Section 3.1).

DEFINITION 4. *A* primitive circuit block *is defined as a circuit block which references no sub-circuit block, or the sub-circuit blocks whose ESD paths have been detected.*

For a primitive circuit block, detecting its ESD paths is different from the traditional reachability problem [3], due to the ESD path definition (see Definition 3). In this paper, we first model a primitive circuit block as a constrained graph (see Section 3.2.1). Given a constrained graph, the ESD connected components in the graph are decomposed (see Section 3.2.2). After that, the stage of pad matching detects all ESD paths of the primitive circuit block (see Section 3.2.3). Finally, we prove the correctness of our algorithm.

3.1 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 Figure 3, where circuit block TOP references circuit blocks A and B, this example implies that A and B have to be processed, i.e., be detected their ESD paths, before TOP. Thus, when detecting the ESD paths of TOP, we have already known the ESD paths of the blocks referenced by TOP. Performing post-order 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 sub-circuit blocks of b have been processed before b in P_f. For example, the POT order in Figure 3, <F, F, C, D, A, F, F, C, F, E, B, TOP>, is a feasible processing order, in which {F} is processed before C, {F, C, D} are processed before A, and so on.

However, since a sub-circuit block can be referenced repeatedly, it is not necessary to detect its ESD paths whenever it is

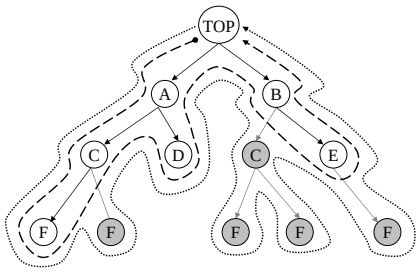

Figure 3: An example hierarchical-circuit topology tree. The nodes and edges represent circuit blocks and circuit block references, respectively. In this example, TOP references A and B, A references C and D, and so on. The dotted line traverses the tree in a post-order, <F, F, C, D, A, F, F, C, F, E, B, TOP>. The dashed line traverses the tree in a post-order with redundancy removal, <F, C, D, A, E, B, TOP>, in which the gray nodes are redundant (already visited).

referenced. As illustrated in Figure 3, 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, <F, C, D, A, E, B, TOP>, is an irredundant order. To remove the redundancy during POT, we establish a hash table saving processed circuit blocks and their ESD paths 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.

3.2 Primitive Circuit Processing

A primitive circuit block references no sub-circuit block or sub-circuit blocks whose ESD paths have been detected (Definition 4). Hence, we devise the following three stages to process a primitive circuit block: *Constrained Graph Construction* models the netlist describing the primitive circuit block as a constrained graph, *ESD Connected Component Decomposition* detects all ESD connected components linked with pad vertices of the constrained graph, and *Pad Matching* pairs the pads bridging the ESD paths according to the connection of the detected ESD connected components.

3.2.1 Constrained Graph Construction

A *constrained 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: pad vertex (p-vertex for short) represents the pad of the primitive circuit block; non-pad vertex (np-vertex) represents all the device terminals except the pads. Besides, edges are also classified into two categories: constrained edge (c-edge) can be involved in an ESD path at most once; non-constrained edge (nc-edge) has no occurrence limit appearing in an ESD path.

Using these pre-defined vertices and edges, we construct the constrained graph according to the device types as follows:

- **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 Figure 4 (a), (b), and (c). Besides, a vertex is labelled as a p-vertex (np-vertex) if it is (is not) a pad of the primitive circuit block.
- **MOS Transistors**: A MOS transistor is referenced with three terminals: gate, source, and drain. To meet the first constraint of the ESD path definition (Definition 3), 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. Figure 4 (d) and (e) show the graph modeling for a MOS. Also, a vertex is labelled as a p-vertex (np-vertex) if it is (is not) a pad of the primitive circuit block.

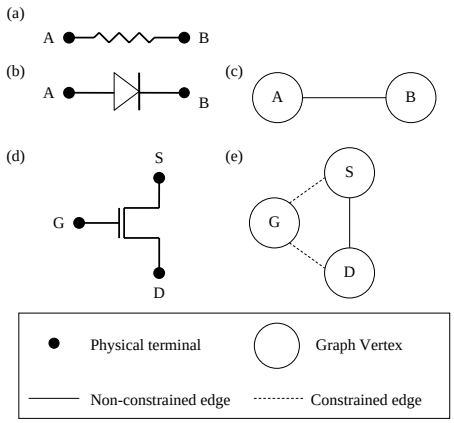

Figure 4: (a) A resistor with its two terminals. (b) A diode with its two terminals. (c) The constrained graph model of a resistor/diode. (d) A MOS transistor with its three terminals: gate, source, and drain (denoted by G, S, and D, respectively). (e) The constrained graph model of a MOS transistor.

- **Sub-Circuit References**: A primitive circuit block references blocks whose ESD paths have been detected (see Definition 4). Assume that there are two terminals, t_1 and t_2, in a primitive circuit block and their corresponding vertices in the constrained graph are v_1 and v_2. If t_1 and t_2 are connected to two pads of a referenced block, 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 3.1. All cases are listed below:

  1. 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.
  2. 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.
  3. 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.

An example constrained graph is given in Figure 5, which is constructed from the netlist shown in Figure 1, using the construction methods presented in this subsection.

3.2.2 ESD Connected Component Decomposition

In this section, we propose an ESD Connected Component Decomposition (ECCD) algorithm (see Figure 6) which decomposes ESD connected components linked with p-vertices from a constrained graph.

DEFINITION 5. *Let $G = (V, E)$ be a constrained graph, where V and E are the vertex set and the edge set, respectively. Let $V_{np} \subseteq V$ be the non-pad vertex set and $E_c \subseteq E$ be the constrained edge set. An* ESD connected component *is a vertex set $C \subseteq V_{np}$. $\forall v_i, v_j \in C$, $i \neq j$, there is a path between v_i and v_j and this path involves no edge $e \in E_c$.*

By the definition, an ESD connected component (ECC) of a constrained graph contains only np-vertices, and for each np-vertex in the ECC, there is always a path leading to every other

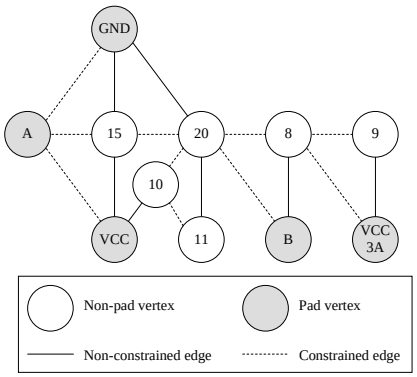

Figure 5: The constrained graph constructed from the netlist in Figure 1, using the construction methods presented in Section 3.2.1.

np-vertices in the ECC by nc-edges. Following the definition, our objective in this stage is to find in a constrained graph:

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

Among them, we define the precedent neighboring ESD connected components below:

DEFINITION 6. *Assume the ECCD algorithm decomposes ESD connected components in the following order* $P_{ECC} =< ECC_1, ECC_2, \ldots, ECC_n >$, *where* ECC_i *is an ESD connected component,* $1 \leq i \leq n$. *The* precedent neighboring ESD connected components *of* ECC_i *is a set* $S^i_{PN} = \{ECC_j \mid ECC_j \in P_{ECC}, j < i,$ *and* ECC_j *is linked to* ECC_i *by at least one constrained edge*$\}$.

Using the constrained graph in Figure 5 as an example, we explain the ECCD algorithm described in Figure 6 to decompose the ESD connected components 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 <GND, VCC3A, B, VCC, A>. Starting from the p-vertices, v_p is GND in the beginning. In line 6, we have an np-vertex v_i being the vertex 15 and vertex 15 is not visited. Now a new ESD connected component ECC_1 is ready to expand using the breadth-first search (BFS); see lines 15–27. In lines 18–19, vertex 15 collects GND and VCC as the p-vertices linked to 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 ECC_1 listed in the first row of Table 1. Now, the set of ESD connected components, S_{ECC}, has the first element, ECC_1 (line 28).

Back to line 6, v_i has another choice, vertex 20, expanding another ESD connected component, 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 ESD connected component, ECC_1, through visiting vertex 15 (lines 20–22). On the other hand, vertex 20 also visits the np-vertex, vertex 11, trying to expand ECC_2 and to collect more p-vertices and more precedent neighboring ESD connected components. After the second BFS run finishes, the resulting ECC_2 is united to the set, S_{ECC}. To this point, the p-vertex, GND, cannot further expand.

Similar to the aforementioned steps, VCC3A, B, and VCC detect the ESD connected components, ECC_3, ECC_4, and ECC_5, respectively. However, the p-vertex, A, detects no ESD connected component, since all the np-vertices have been visited. Finally, the ECCD algorithm returns the set of the ESD connected components detected from the given constrained graph. In this example,

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}

Figure 6: The ESD Connected Component Decomposition (ECCD) algorithm.

$S_{ECC} = \{ECC_1, ECC_2,$
$ECC_3, ECC_4, ECC_5\}$, and all the related information is listed in Table 1.

Since all the np-vertices in the constrained 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 BFS's in $O(|V|$
$+ |E|)$ time. During one of the BFS's, the ECCD collects p-vertices in $O(p)$ time for each visited p-vertex (see lines 18–19), where p is the average number of the visited p-vertices of an ESD connected component, and collects precedent neighboring ESD connected components in $O(n_{PN})$ time for each visited np-vertex via c-edges (see lines 20–22), where n_{PN} is the average number of precedent neighboring ESD connected components of an ESD connected component. Let V_p be the visited p-vertex set and V_{npc} be the set of np-vertices visited by c-edges. The time-complexity of the ECCD algorithm is $O(p\,|V_p| + n_{PN}\,|V_{npc}| + |V \setminus V_p \setminus V_{npc}| + |E|)$. From the fact that $p \leq |V|$, $|V_p| \leq |V|$, $n_{PN} \leq |V|$, $|V_{npc}| \leq |V|$, and $|E| = O(|V|^2)$, the time complexity of the ECCD algorithm is $O(|V|^2)$.

Note that the ECCD algorithm decomposes ESD connected

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

Table 1: The ESD connected components decomposed from the constrained graph in Figure 5 using the ECCD algorithm, assuming the decomposition order is $< ECC_1, ECC_2, ECC_3, ECC_4, ECC_5 >$.

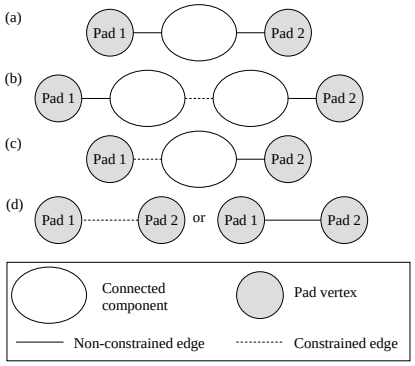

Figure 7: The four feasible connection types of an ESD path.

components *linked with p-vertices only*, instead of all ESD connected components. This is why we check only p-vertices in line 5 in 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 ESD connected components from a constrained graph, for comparative study (see Section 4).

THEOREM 1. *The ESD paths detected by ECCD (which decomposes ESD connected components linked with p-vertices only) are identical to those detected by ECCD-WPI (which decomposes all ESD connected components).*

Proof: For an ESD path between two pads, there are four possible combinations of the two pads as follows:

1. The two pads are linked to the same ESD connected component via nc-edges, as shown in Figure 7(a).
2. The two pads are linked to two different ESD connected components via nc-edges, and the two components are linked via a c-edge, as shown in Figure 7(b).
3. One of the pads is linked to an ESD connected component via an nc-edge, and the other pad is linked to the component via a c-edge, as shown in Figure 7(c).
4. The two pads are directly linked to each other, as shown in Figure 7(d).

For all connected components shown in Figure 7, 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. □

3.2.3 Pad Matching

After the ESD connected component decomposition 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 7–8). Therefore, in this stage, we do not need to match the pads of this type. Besides, the type-1 path has a higher priority than 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, for that P_1 can conduct more ESD paths in a higher-level circuit blocks than P_2.

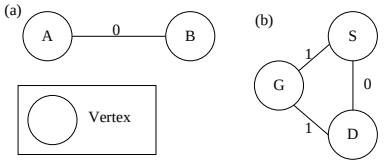

Figure 8: Graph connections used in the FW algorithm. (a) a resistor or a diode. (b) a MOS transistor (the gate, source and drain are denoted by G, S and D, respectively).

Given the ESD connected components returned from the ECCD algorithm, we can match the pads according to the feasible connection types. For example, in Figure 5 and Table 1, we have detected the ESD connected components using the ECCD algorithm. Now for each ESD connected component, we match its p-vertices according to the type-1 connection: (GND, VCC), and the type-3 connection: (A, GND), (A, VCC), (GND, B), and (B, VCC3A). In addition, for each ESD connected component and its precedent neighboring ESD connected components, we match their p-vertices according to the type-2 connection: (GND, VCC), (GND, B) and (B, VCC3A). There are also type-4 connections having been detected in the ECCD algorithm: (A, GND) and (A, VCC). Considering the path priority, the resulting matching is type-1: (GND, VCC), type-2: (GND, B) and (B, VCC3A), and type-3: (A, GND) and (A, VCC).

Finally, these paths are saved in the hash table for the subcircuit references mentioned in Section 3.2.1. In the above example, (GND, VCC) is saved as the first reference case, while the others are saved as the second one.

4. EXPERIMENTAL RESULTS

We implemented our algorithms, ECCD and ECCD-WPI, in the C/C++ language on a 1.6 Ghz Intel Pentium 4 PC with 2 GB memory under Linux 2.6 operating system. There are ten industrial circuits from a leading design service company as shown in Table 2. As shown in the last three columns, the number of ESD paths 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 circuit overheads by identifying the potential ESD paths and protecting only those paths.

We compare our algorithms with one based on the Floyd-Warshall all-pairs shortest paths algorithm [2]. The comparative algorithm, called FW, transfers the circuit to a weighted undirected graph by the following rules:

1. A resistor or a diode is transferred into an edge with weight, 0, as shown in Figure 8 (a).
2. A MOS transistor is transferred to three edges as shown in Figure 8 (b). The source-drain edge is with weight, 0; the gate-source or gate-drain edge is with weight, 1.

Circuit	Number of						ESD Path % (A/B)
	Pads	Circuit Blocks	Vertices	References of Circuit Blocks	ESD Paths (A)	All Paths (B)	
Ind. 1	261	73	4060	917	601	33930	1.77
Ind. 2	542	317	17129	43424	1234	146611	0.84
Ind. 3	15	436	56267	56635	39	105	37.14
Ind. 4	26	604	63026	59675	65	325	20.00
Ind. 5	28	263	102603	829859	55	378	14.55
Ind. 6	28	295	156306	797381	53	378	14.02
Ind. 7	69	840	424202	1675863	95	2346	4.05
Ind. 8	69	1411	735271	3394388	95	2346	4.05
Ind. 9	69	2193	1016038	5036407	102	2346	4.35
Ind. 10	69	2601	1339677	5176763	136	2346	5.63

Table 2: The parameters of the ten industrial circuits. The number of vertices represents the total vertices of the flattened circuit if we expand all sub-circuits in the netlist.

Circuit	FW		ECCD-WPI			ECCD		
	CPU Time (s)	Memory (MB)	CPU Time (s)	Memory (MB)	NDECC	CPU Time (s)	Memory (MB)	NDECC
Ind. 1	8.88	62	3.02	1	3164	0.03	1	916
Ind. 2	9.37	63	3.61	1	9178	0.04	1	2368
Ind. 3	13.39	72	4.87	1	28894	0.07	1	11528
Ind. 4	14.58	86	5.59	1	33446	0.09	1	13291
Ind. 5	93.11	646	67.46	47	15042	1.10	26	1601
Ind. 6	96.36	638	67.55	46	15175	1.06	26	1633
Ind. 7	164.20	699	80.56	53	42547	1.43	26	6095
Ind. 8	333.21	805	102.15	66	76176	2.03	25	10189
Ind. 9	1559.44	937	125.30	78	130432	2.61	28	24461
Ind. 10	8550.29	1097	133.11	80	221389	2.78	44	68515

Table 3: Comparison of the three algorithms. NDECC is the abbreviation of "number of decomposed ESD connected components."

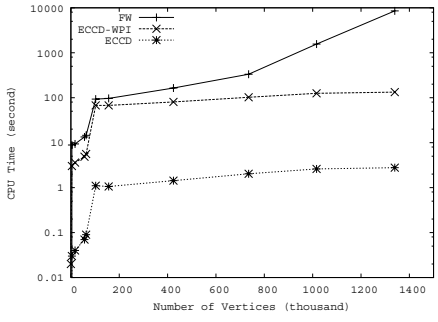

Figure 9: The number of vertices versus CPU time of the three algorithms.

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 is done by the Floyd-Warshall all-pairs shortest paths algorithm.

The three algorithms (ECCD, ECCD-WPI, and FW) detect the same ESD paths for all test circuits. Table 3 shows the CPU time and memory usages of the three algorithms. In Figure 9, 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 3 and Figure 9, ECCD is about 50X faster than ECCD-WPI, and ECCD-WPI is significantly faster than FW (the larger the circuit, the bigger the runtime difference). For example, when the number of vertices in the circuit reaches 1,339,677 (Ind. 10), ECCD still completes the computation in 2.78 seconds, while the ECCD-WPI requires more than 133 seconds and FW needs more than 8,550 seconds. Besides, ECCD is much more economical in memory usage than ECCD-WPI and FW.

It is also shown in Table 3 that the number of decomposed ESD connected components of ECCD is much smaller than that of ECCD-WPI. Taking the largest circuit (Ind. 10) for example, the ECCD decomposes 68,515 ESD connected components while the ECCD-WPI decomposes 221,389 ones. Therefore, our ECCD algorithm is very efficient and economical. This difference in the number of decomposed ESD connected components 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 work can indeed reduce the design cost.

5. CONCLUSIONS

We have proposed *the first* ESD detection algorithm in circuit design to efficiently detect all pads in danger of an ESD violation. Experimental results have shown that our algorithm is very efficient and economical. Along with the back-to-back diode protection, a circuit is more well-protected from the ESD threat and more reliable.

6. ACKNOWLEDGEMENTS

We thank Mr. Chih-Yang Peng and Dr. Yu-Wei Chen of Faraday Technology Corporation for their patient and valuable consultation on the industrial ESD practices.

7. REFERENCES

[1] A. Amerasekera and C. Duvvury, *ESD in Silicon Integrated Circuits*, 2^{nd} edition, John Wiley & Sons, 2002.

[2] T. Cormen, C. Leiserson, R. Rivest, and C. Stein, *Introduction to Algorithms*, 2^{nd} edition, The MIT Press, 2001.

[3] E. Nuutila, "Efficient Transitive Closure Computation in Large Digraphs," Ph. D. dissertation, *Mathematics and Computing in Engineering Series*, no. 74, Helsinki, 1995.

[4] B. P. Wong, A. Mittal, Y. Cao, and G. Starr, *Nano-CMOS Circuit and Physical Design*, John Wiley & Sons, 2005.

[5] 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 Tran. Computer-Aided Design*, vol. 23, no. 10, pp. 1421–1428, 2004.

[6] R. Zhan, H. Xie, H. Feng, and A. Wang, "ESDZapper: A New Layout-level Verification Tool for Finding Critical Discharging Path Under ESD Stress," *Proc. ASPDAC*, pp. 79–82, 2005.